UNIVERSIDADE FEDERAL DE SANTA CATARINA

ENGENHARIA ELÉTRICA

EXTRAÇÃO AUTOMÁTICA DE LINHAS DE COSTA DA REGIÃO DA LAGOA DA CONCEIÇÃO EM FLORIANÓPOLIS

Alexandre Ferreira de Carvalho Filho – Matrícula 17200317

João José Medeiros de Figueiredo – Matrícula 21103475

EEL7815 – Projeto Nível 2 em Controle e Processamento de Sinais II

Turma: 08202

Professor: Joceli Mayer

FLORIANÓPOLIS-SC

Dez/2022

SUMÁRIO

1. **INTRODUÇÃO**

Já estamos vivendo a emergência climática, percebemos isso pelo aumento dos eventos extremos, como ciclones bomba, chuvas ou secas intensas, ondas de calor, aumento do nível do mar com alagamento ou erosão. A previsão feita por especialistas climáticos afirma que chegaremos ao aumento da temperatura global de 1,5 à 2 ºC em 2050. Um estudo feito pelo Laboratório de Clima e Meteorologia da UFSC afirma, com dados do painel do IPCC a partir da plataforma climate central, que o nível do mar nas áreas marginais da Lagoa da Conceição e do canal do Barra em Florianópolis, irá aumentar e alagar boa parte da região, tendo essa previsão em mente, necessitamos de diversos estudos para acompanhar a erosão ou alagamento das áreas da Lagoa da Conceição, para acompanhar e documentar o processo, a fim de melhorar o plano diretor da cidade de Florianópolis e ajudar os governantes e a população a trilhar o melhor caminho possível para preservar o ecossistema do local.

Por isso, o objetivo deste projeto é construir uma ferramenta computacional capaz de localizar e extrair as linhas de costa da Região da Lagoa de Conceição em Florianópolis-SC, do banco de imagens geo-espaciais do INPE, para fins de pesquisas de georreferenciamento multi-temporal sobre a erosão e o assoreamento da área, haja em vista, a aceleração do aumento do nível dos mares.

2. MÉTODOS E MATERIAIS

A metodologia adotada nesse projeto, pode ser dividida nos seguintes passos

a) Aquisição das Imagens de satélites (LandSat-5 e 7, CBERS-4A)
b) Destacamento da água utilizando a técnica MNDWI
c) Recorte das imagens nas coordenadas da Lagoa da conceição utilizanndo a biblioteca gdal
d) Binarização utilizando a biblioteca OpenCV
e) Floodfill utilizando a biblioteca OpenCV

f) Canny utilizando a biblioteca OpenCV
g) Conversão de .tiff para .png
h) Coloração das Imagens para melhor visualização utilizando a biblioteca OpenCV
i) Fundo Transparente utilizando a biblioteca OpenCV
j) Mesclagem das imagens utilizando a biblioteca OpenCV

Para implementar essas funcionalidades, as seguintes versões do python e das bibliotecas foram utilizadas:

- python - 3.9.13
- gdal - 3.4.3
- matplotlib - 3.5.3
- numpy - 1.21.6
- skymage - 0.19.3
- Pillow - 9.3.0

As bibliotecas (scikit-image, matplotlib, numpy, Pillow, GDAL) foram instaladas utilizando o comando pip install. Ressalta-se que para GDAL, foi baixado a o pacote com a versão mais recente, no site de Christoph Gohlke[7]

3.2 AQUISIÇÃO DAS IMAGENS DE SATÉLITES

As imagens de satélite utilizadas no trabalho, foram adquiridas no site do INPE [6], para fins de teste foram utilizadas imagens de 3 satélites diferentes Landsat 5, Landsat 7 e CBERS-4A. No entanto, devido a necessidade de se buscar imagens com maior série histórica e da qualidade das mesmas (Ruídos, nuvens e etc), optou-se pelas seguintes 6 cenas (220/79) dos landsats 5 e 7.

ANO	SATÉLITE	DATA	LUA
1986	LANDSAT 5	30/set	MINGUANTE
1991	LANDSAT 5	13/out	NOVA
1994	LANDSAT 5	18/jul	CRESCENTE
1999	LANDSAT 7	25/fev	CHEIA
2005	LANDSAT 5	21/nov	CHEIA
2010	LANDSAT 5	04/fev	CHEIA

Figura 1 *Imagens utilizadas*

3.2 DESTACAMENTO DA ÁGUA UTILIZANDO A TÉCNICA MNDWI

Como se observou em outro estudo [1], a técnica MNDWI *(Modified Normalized Difference Water Index)* pode ser utilizada a fim de destacar a água em uma imagem de satélite, para isso se utiliza de duas bandas de frequência das câmeras dos satélites a verde e a SWIR-1 (Infravermelho médio), respectivamente a banda 2 e 5 do landsat 5 e 7.

Sensor	Bandas Espectrais	Resolução Espectral	Resolução Espacial	Resolução Temporal	Área Imageada	Resolução Radiométrica
TM (Thematic Mapper)	(B1) AZUL	0.45 - 0.52 µm	30 m	16 dias	185 km	8 bits
	(B2) VERDE	0.50 - 0.60 µm				
	(B3) VERMELHO	0.63 - 069 µm				
	(B4) INFRAVERMELHO PRÓXIMO	0.76 - 0.90 µm				
	(B5) INFRAVERMELHO MÉDIO	1.55 - 1.75 µm				
	(B6) INFRAVERMELHO TERMAL	10.4 - 12.5 µm	120 m			
	(B7) INFRAVERMELHO MÉDIO	2.08 - 2.35 µm	30 m			

Figura 2 Bandas de Frequência do Landsat 5 e 7

A técnica consiste em uma conta algébrica das matrizes das duas bandas, onde o seu resultado estará entre os limites de -1 e 1, indicando para valores acima de 0, possivelmente água e abaixo terra. O cálculo do MNDWI é o seguinte:

$$MNDWI = \frac{Green - Swir\ 1}{Green + Swir\ 1}$$

Para aplicar essa técnica foi utilizado o seguinte código implementado em python, utilizando a biblioteca gdal:

```
from osgeo import gdal

band_green="LANDSAT_7_ETMXS_19990825_220_079_L2_BAND2"
band_swir="LANDSAT_7_ETMXS_19990825_220_079_L2_BAND5"
end_band_green='D:\OneDrive\Organizar\UFSC\Documentos\Georreferenciamento\La
ndsat 7\19990825\LANDSAT_7_ETMXS_19990825_220_079_L2_BAND2'
end_band_swir='D:\OneDrive\Organizar\UFSC\Documentos\Georreferenciamento\Lan
dsat 7\19990825\LANDSAT_7_ETMXS_19990825_220_079_L2_BAND5'
MNDWI_19990825='D:\OneDrive\Organizar\UFSC\Documentos\Georreferenciamento\La
ndsat 7\19990825\MNDWI_19990825.tif'

processing.run("qgis:rastercalculator",
               {'EXPRESSION':'(band_green - band_swir) /
(band_green  +  band_swir)',
                'LAYERS':[end_band_green,end_band_swir],
                'CELLSIZE':0,'EXTENT':None,
                'CRS':QgsCoordinateReferenceSystem('EPSG:31982'),
                'OUTPUT':MNDWI_19990825})
```

3.3 RECORTE DAS IMAGENS NAS COORDENADAS GEOGRÁFICA DA LAGOA DA CONCEIÇÃO

Para obtenção dos recortes das imagens, utilizamos em sua maior parte a biblioteca GDAL, e utilizando um arquivo criado manualmente no software QGIS 3.28.1 no formato .shp no qual nomeamos de *“Coord_lagoa_tudo.shp”*, conseguimos padronizar todos os recortes das imagens com o recorte que achamos mais adequado.

Para fins de melhor visualização dos efeitos lunares, decidimos recortar parte do mar na encosta da Lagoa, para perceber o nível das marés e podermos tirar melhores comparativos.

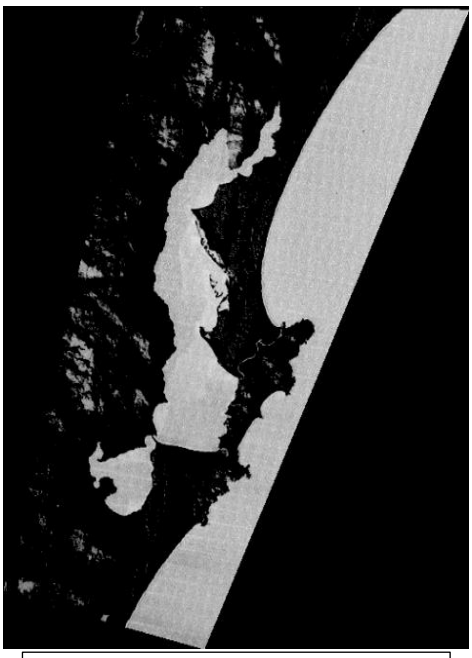
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from osgeo import gdal

#abrindo dataset para recorte
ndwi_23062020 = gdal.Open(r"C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\ndwi_23062020.tif")
ndwi_01022022 = gdal.Open(r"C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\ndwi_01022022.tif")

#### recortar
#### garantir que o dataset-rasterizado e o shapefile(.shp) tenham a mesma projeção.

datasetClip = gdal.Warp(r"C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\recorte_NDWI_23062020.tif",
                        ndwi_23062020,
                        cutlineDSName = r"C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\Coord_lagoa_tudo.shp",
                    cropToCutline = True, dstNodata = np.nan)

datasetClip = gdal.Warp(r"C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\recorte_NDWI_01022022.tif",
                        ndwi_01022022,
                        cutlineDSName = r"C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\Coord_lagoa_tudo.shp",
                    cropToCutline = True, dstNodata = np.nan)
```

```
#recortes prontos
recorte_20200623 = r"C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\recorte_NDWI_01022020.tif"
recorte_20200623 = r"C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\recorte_NDWI_23062020.tif"

img_nova = cv2.imread(recorte_20200623, -1)

plt.imshow(img_nova, cmap="gray", vmin=-1, vmax=1)
plt.show()
```

3.4 BINARIZAÇÃO EM OPENCV

A binarização consiste em uma técnica de processamento em que se divide a intensidade das imagens em duas grandes classes, no caso das imagens recortadas dos MNDWI, água e terra. Para isso se define um threshold, ou seja, um limiar que se deve considerar para fronteira entre as duas classes.

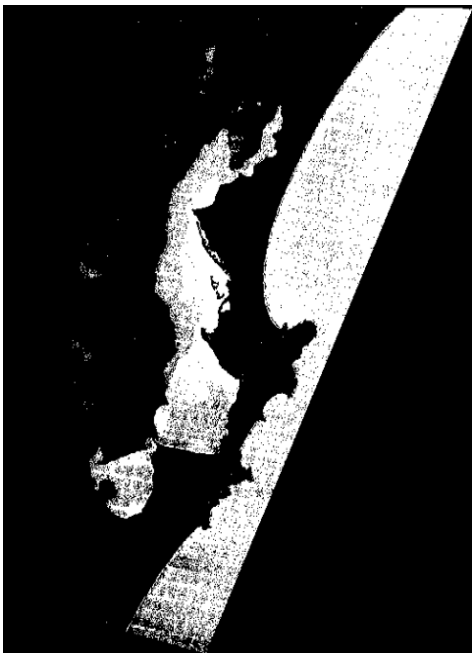
Durante o desenvolvimento do código, adotou-se em um primeiro momento o valor de 0 para o threshold. No entanto, nas análises das imagens, percebeu-se um problema na detecção dos morros dos macacos, que é o triangulo com vegetação na imagem do norte da lagoa da conceição.



(a)Sem binarização

(b)thres=0.2

(c)thres=0.3



(d)thres=0.4

Figura 3 Threshold aplicado a imagem de 1994

Figura 4 Morro dos Macacos

Este morro em algumas fotos, com as de 1994 e 2005, não apresentavam a detecção do morro. Por isso se fez necessário alterar manualmente o Threshold, para cada imagem. Esses testes podem ser observados, nos seguintes 4 quadros da imagem de 1994, onde percebe-se o efeito da binarização de *"a"* para *"b".* Além disso, observando o aumento do threshold feito para detectar o morro dos macacos, que em certo momento a imagem começa a perder a qualidade, como em *"d",* deste modo, se optou por utilizar a imagem em *"b"* mesmo que ela não tenha identificado com precisão dos morro dos macacos, ela apresenta menos ruídos que *"c"*, observe que esta última apresenta mais pontos pretos no sul da lagoa. Este processo, se repetiu para os 6 casos, obtendo os seguintes resultado

ANO	THRESHOLD
1986	0.2
1991	0.1
1994	0.2
1999	0.2
2005	0
2010	0.2

O código implementado, foi o seguinte

```
img = cv2.imread(rec, -1)

print(f'dtype: {img.dtype}, shape: {img.shape}, min: {np.min(img)}, max:
{np.max(img)}')

plt.imshow(img, cmap="gray", vmin=-1, vmax=1)
plt.show()

ret, thresh = cv2.threshold(img, 0.3, 1, cv2.THRESH_BINARY)
plt.imshow(thresh)
plt.show()
```

3.5 FLOODFILL EM OPENCV

Como pode se observar nas imagens anteriores, ainda se apresenta grandes manchas brancas que não fazem parte da lagoa, logo para destacar apenas as curvas da lagoa, foi utilizado a técnica *flood fill* que a partir de pontos pré-definidos dentro de objetos com fronteira fechadas, consegue “pinta-los” como pode ser observado na imagem abaixo, onde foram escolhidos 3 pontos, um na lagoa norte, outra na lagoa ao sul e último no mar.

Para o isso o seguinte código foi implementado, utilizando o Opencv e a biblioteca Floodfill.

```
start_pt_meio=(200,350) # lagoa de cima
start_pt_baixo=(130,420) # lagoa de baixo
start_pt_mar=(170,600) # mar

h, w = thresh.shape[:2]
mask = np.zeros((h+2, w+2), np.uint8)
im_floodfill = thresh.copy()

cv2.floodFill(im_floodfill, mask, start_pt_meio, 255, flags=4)
cv2.floodFill(im_floodfill, mask, start_pt_baixo, 255, flags=4)
cv2.floodFill(im_floodfill, mask, start_pt_mar, 255, flags=4)
plt.imshow(im_floodfill)
plt.show()
```

A partir deste código foi possível obter as seguintes imagens:

(a)Sem Floodfill | (b)Com Floodfill

Figura 5 Aplicação do FloodFill - 1994

3.6 EXTRAÇÃO DAS CURVAS DE COSTA – CANNY NO OPENCV

A partir de todos os processamentos anteriores, é possível utilizar o algoritmo *Canny*, para detectar a costa da lagoa e da praia. O código e o resultado desse passo podem ser observados abaixo:

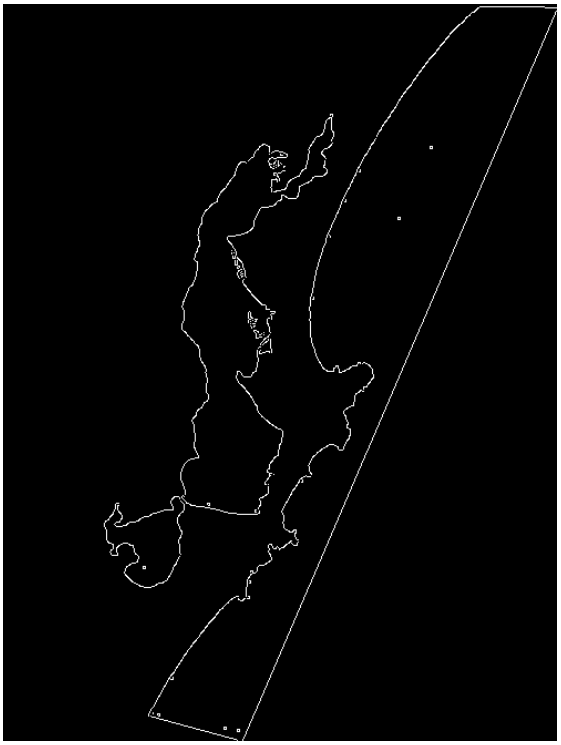


Figura 6 Aplicação do Canny

```
r = 255 # Upper threshold
aperture_size = 3 # Aperture size
L2Gradient = False # Boolean

im_floodfill_1 = np.uint8(im_floodfill)

# Applying the Canny Edge filter
# with Aperture Size and L2Gradient
edge = cv2.Canny(im_floodfill_1, t_lower, t_upper,
                 apertureSize = aperture_size,
                 L2gradient = L2Gradient )
```

3.7 CONVERSÃO DE TIFF.PARA PNG

Foi utilizado a biblioteca Pillow e essa simples função para a conversão.

```
from PIL import Image
im = Image.open('imagem.tiff')
im.save('imagem.png')
```

3.8 COLORAÇÃO DAS IMAGENS PARA MELHOR VISUALIZAÇÃO

Para a coloração foi utilziado uma pequena função definida como cor(): onde fazia a transformação do formato GRAY para RGB, criava-se uma máscara definida por um range de branco no qual as fronteiras da costa da lagoa estavam ‘pintadas’, e depois do programa identificar a encosta, apenas aplicamos a cor em RGB que definimos.

```
import cv2
import numpy as np
from skimage import io

def cor(imagem,cor,nome_do_arquivo):
    imagem = cv2.cvtColor(imagem,cv2.COLOR_GRAY2BGR)
    mask = cv2.inRange(imagem, lower_white, upper_white)
    imagem[mask > 0] = cor
    cv2.imwrite(nome_do_arquivo,imagem)

img1          =          io.imread(r'C:\Users\joaof\OneDrive\Área          de
Trabalho\imagens\1986.png')
img2          =          io.imread(r'C:\Users\joaof\OneDrive\Área          de
Trabalho\imagens\1991.png')
img3          =          io.imread(r'C:\Users\joaof\OneDrive\Área          de
Trabalho\imagens\1994.png')
img4          =          io.imread(r'C:\Users\joaof\OneDrive\Área          de
Trabalho\imagens\1999.png')
img5          =          io.imread(r'C:\Users\joaof\OneDrive\Área          de
Trabalho\imagens\2005.png')
img6          =          io.imread(r'C:\Users\joaof\OneDrive\Área          de
Trabalho\imagens\2010.png')
```

```
#de cores quentes para frias
amarelo = (0,255,255)
laranja = (0,165,255)
vermelho = (0,0,255)
violeta = (238,130,238)
azul = (255,0,0)
verde = (0,128,0)
branco = (255,255,255)
verdepale = (152,251,152)

lower_white = np.array([255,255,255])
upper_white = np.array([255,255,255])

# Define lower and uppper limits of what we call "white"
# color_dict_HSV = {'black': [[180, 255, 30], [0, 0, 0]],
#                'white': [[180, 18, 255], [0, 0, 231]],
#                'red1': [[180, 255, 255], [159, 50, 70]],
#                'red2': [[9, 255, 255], [0, 50, 70]],
#                'green': [[89, 255, 255], [36, 50, 70]],
#                'blue': [[128, 255, 255], [90, 50, 70]],
#                'yellow': [[35, 255, 255], [25, 50, 70]],
#                'purple': [[158, 255, 255], [129, 50, 70]],
#                'orange': [[24, 255, 255], [10, 50, 70]],
#                'gray': [[180, 18, 230], [0, 0, 40]]}

cor(img1,amarelo,'1986_amarelo.png')
cor(img2,laranja,'1991_laranja.png')
cor(img3,branco,'1994_vermelho.png')
cor(img4,azul,'1999_azul.png')
cor(img5,verde,'2005_verde.png')
cor(img6,branco,'2010_branco.png')
```

3.9 FUNDO TRANSPARENTE PARA SOBREPOR AS IMAGENS

Para obtenção do fundo transparente utilizamos a função definida como blackbackgroundtotransparent(): no qual utilizava-se da função treshhold utilizando

o THRESH_BINARY, separava-se em duas partes, e excluía o preto, essa função junta os espectros R,G e B junto com o espectro Alpha definido pelo threshhold.

```
import cv2
from skimage import io

# Import the image
img1_colorida = r'C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\cores\1986_amarelo.png'
img2_colorida = r'C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\cores\1991_laranja.png'
img3_colorida = r'C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\cores\1994_vermelho.png'
img4_colorida = r'C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\cores\1999_azul.png'
img5_colorida = r'C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\cores\2005_verde.png'
img6_colorida = r'C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\cores\2010_branco.png'

def blackbackgroundtotransparent(imagem,nome_do_arquivo):
    src = io.imread(imagem)
    tmp = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
    _,alpha = cv2.threshold(tmp,0,255,cv2.THRESH_BINARY)
    b, g, r = cv2.split(src)
    rgba = [b,g,r, alpha]
    dst = cv2.merge(rgba,4)
    # Writing and saving to a new image
    cv2.imwrite(nome_do_arquivo, dst)

blackbackgroundtotransparent(img1_colorida,'1986_amarelo_transparente.png')
blackbackgroundtotransparent(img2_colorida,'1991_laranja_transparente.png')
blackbackgroundtotransparent(img3_colorida,'1994_vermelho_transparente.png')
```

```
blackbackgroundtotransparent(img4_colorida,'1999_azul_transparente.png')
blackbackgroundtotransparent(img5_colorida,'2005_verde_transparente.png')
blackbackgroundtotransparent(img6_colorida,'2010_branco_transparente.png')
```

3.10 MESCLAGEM DAS IMAGENS

Para sobreposição das imagens apenas utilizamos o sinal de adição, poderíamos ter utilizado a função cv2.add() que retornaria a mesma imagem.

```
import cv2

from skimage import io



img1_colorida = io.imread(r'C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\cores\1986_amarelo.png')

img2_colorida = io.imread(r'C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\cores\1991_laranja.png')

img3_colorida = io.imread(r'C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\cores\1994_vermelho.png')

img4_colorida = io.imread(r'C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\cores\1999_azul.png')

img5_colorida = io.imread(r'C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\cores\2005_verde.png')

img6_colorida = io.imread(r'C:\Users\joaof\OneDrive\Área de Trabalho\22.2\Projeto Nível 2 em Controle e Processamento de Sinais II\projeto_lagoa\imagem de comparação\cores\2010_branco.png')
```

```
imagens_juntas = img1_colorida + img2_colorida + img3_colorida + img4_colorida
+ img5_colorida + img6_colorida



cv2.imwrite('mesclar.png',imagens_juntas)
```

3.CONCLUSÃO

Sobrepondo todas as 6 imagens de 1986,1991,1994,1999,2005 e 2010 conseguimos obter a seguinte imagem abaixo.

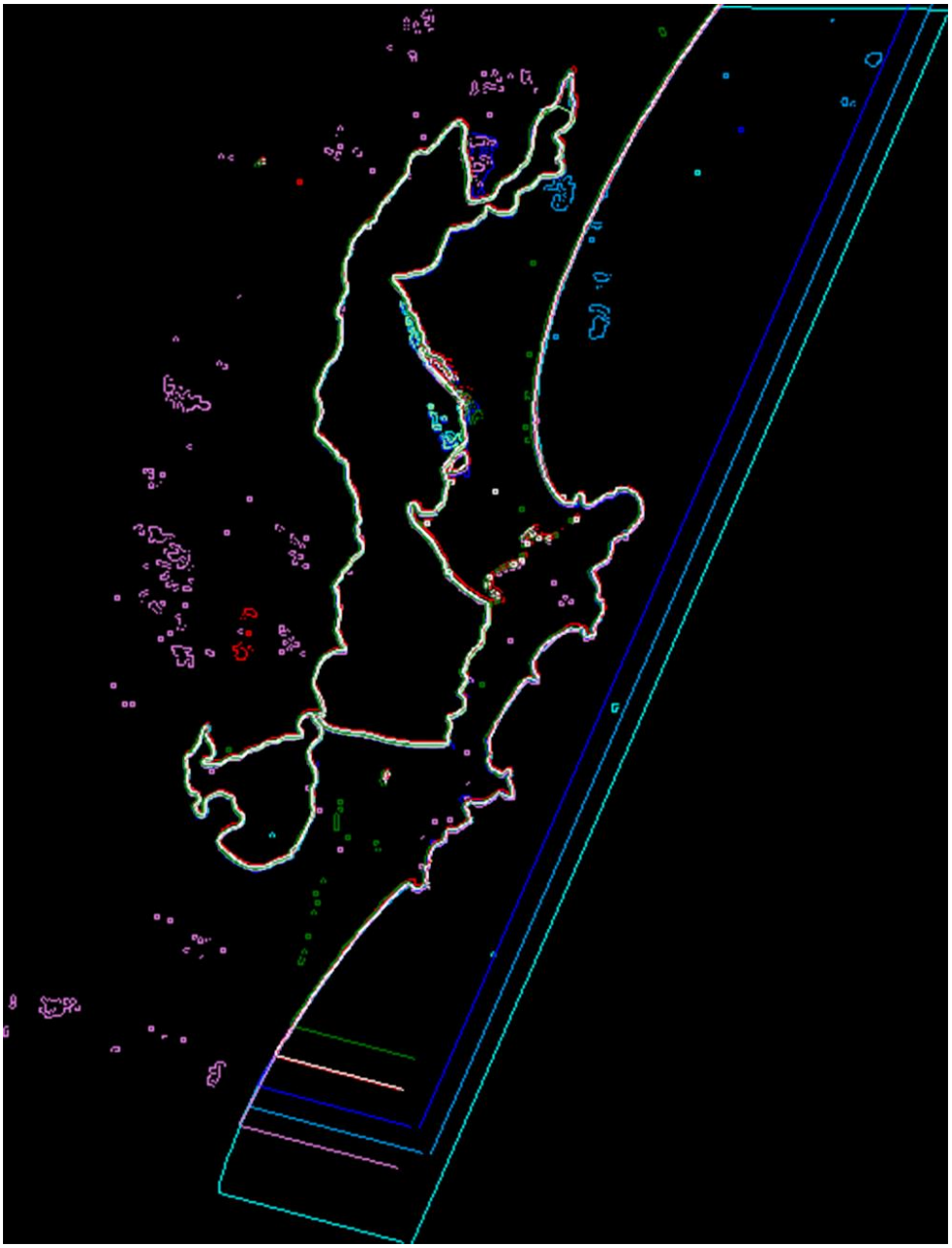


Figura 7 Imagens sobrepostas

Ainda, foi testado também imagens retiradas do satélite CBERS-4A, em branco uma foto de 01/02/2020 e em vermelho uma foto de 23/06/2022. Ainda sem a aplicação da função floodfill. As duas estão sob influência de Lua Nova. Essa escolha se dá para mostrar os efeitos do Floodfill sobre a qualidade das curvas que foram obtidas.

Figura 8 Imagens do CBERS-4A (2020-2022), sem flodfill

Como pode ser observado na figura 7, a imagem ficou “poluída”, por isso a equipe optou por fazer a análise das curvas da costa, comparando dois-a-dois as imagens, conforme a seguir.

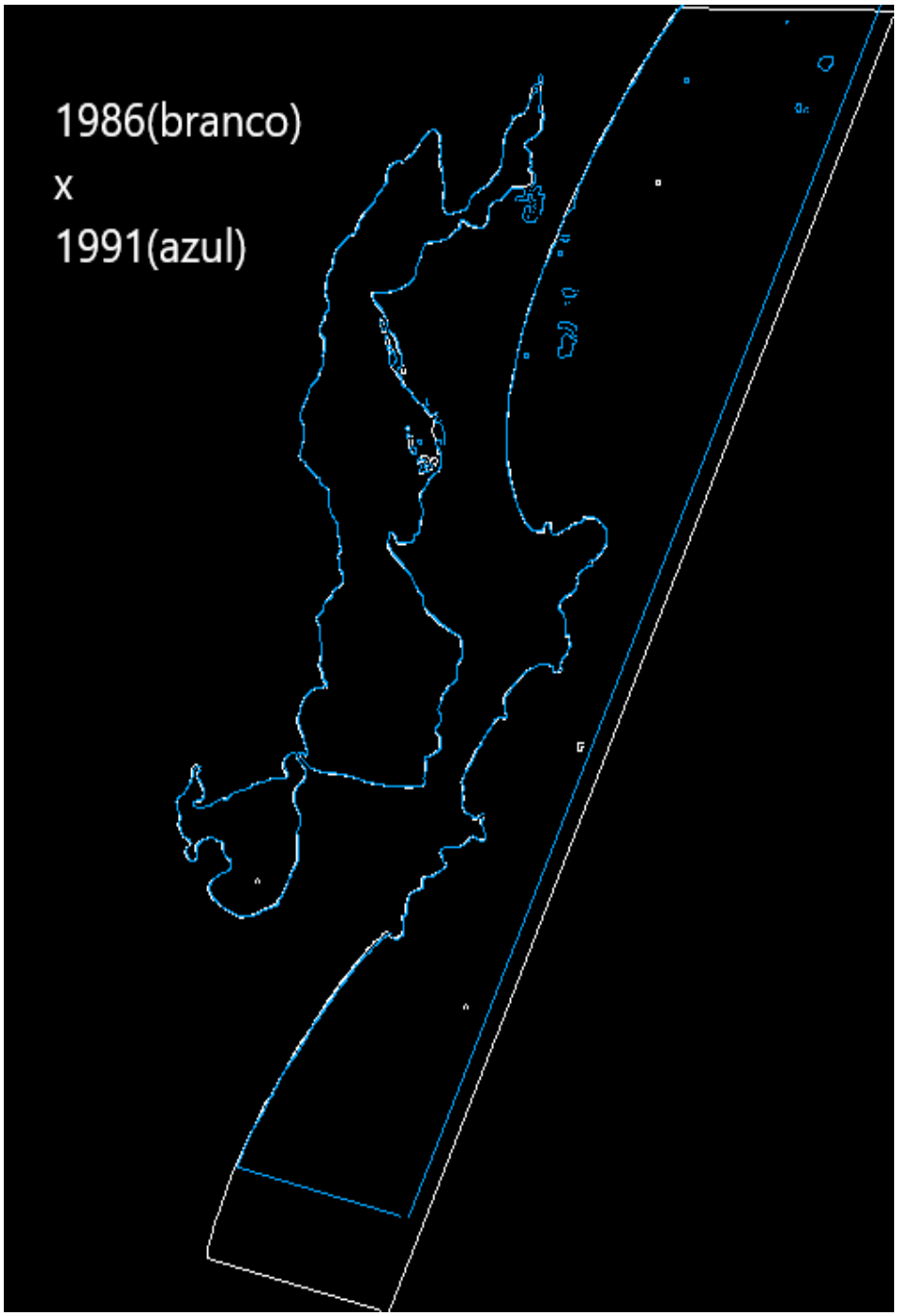


Figura 9 Comparação entre 1986 e 1991

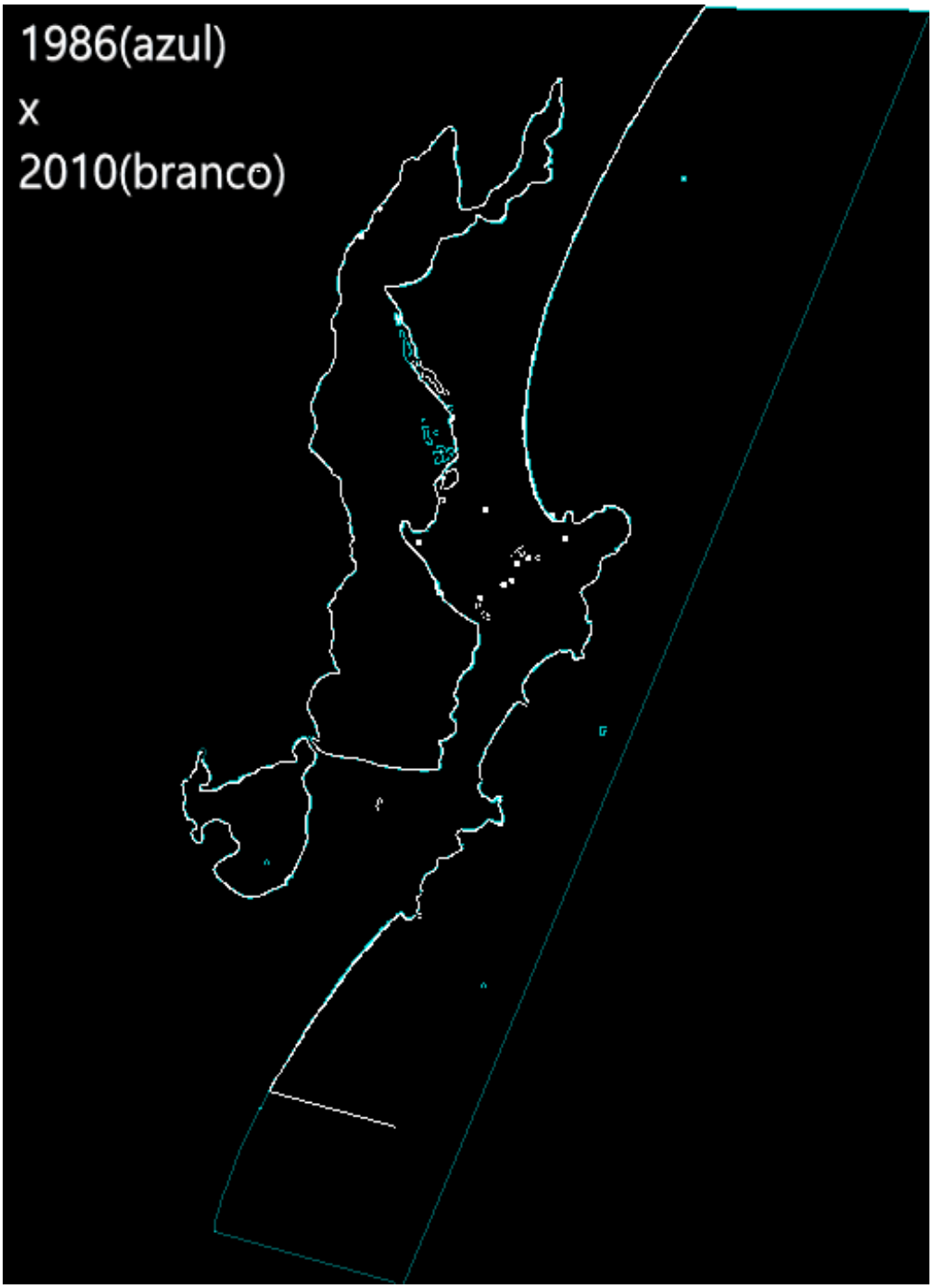


Figura 10 Comparação entre 1986 e 2010

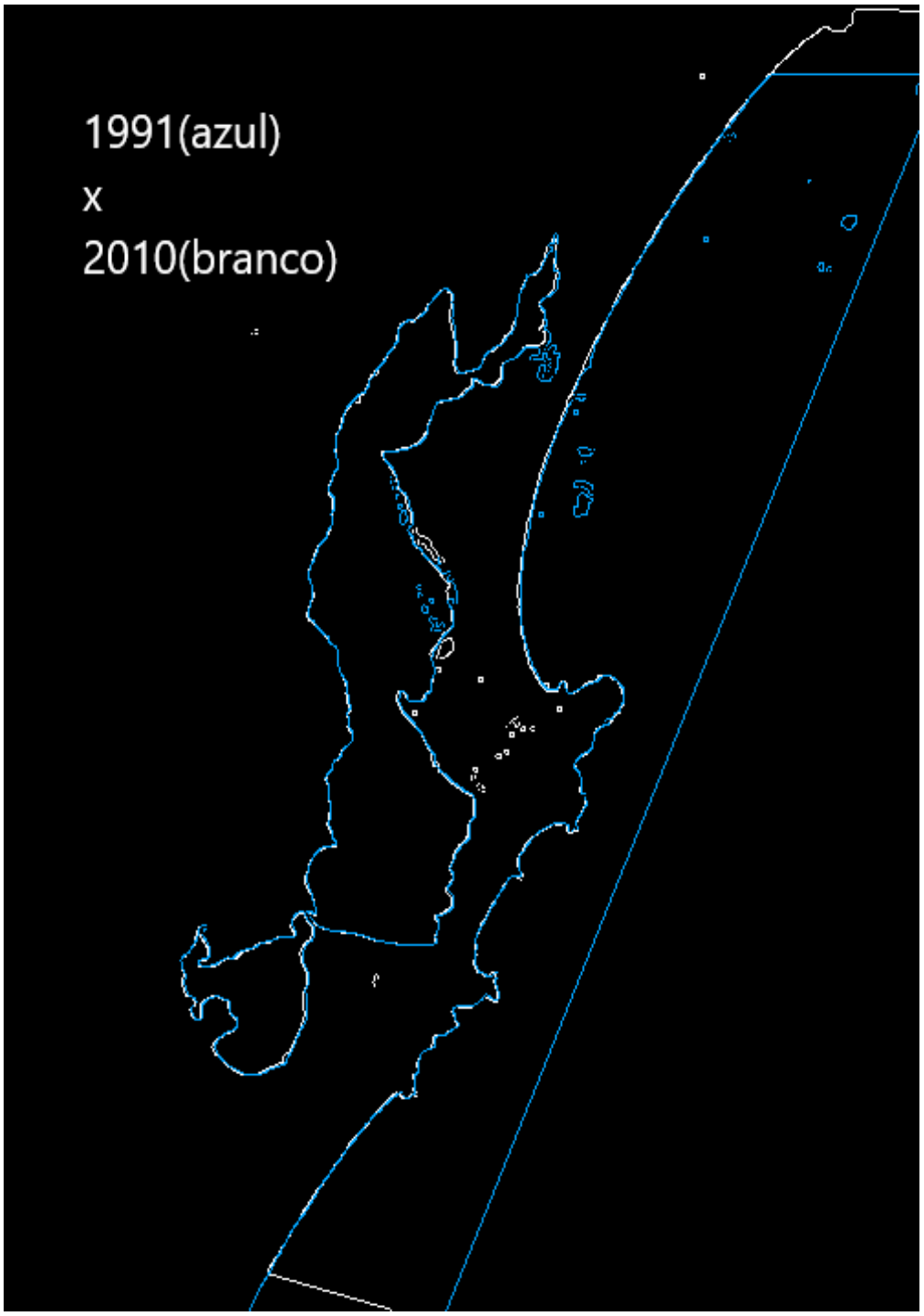


Figura 11 Comparação entre 1991 e 2010

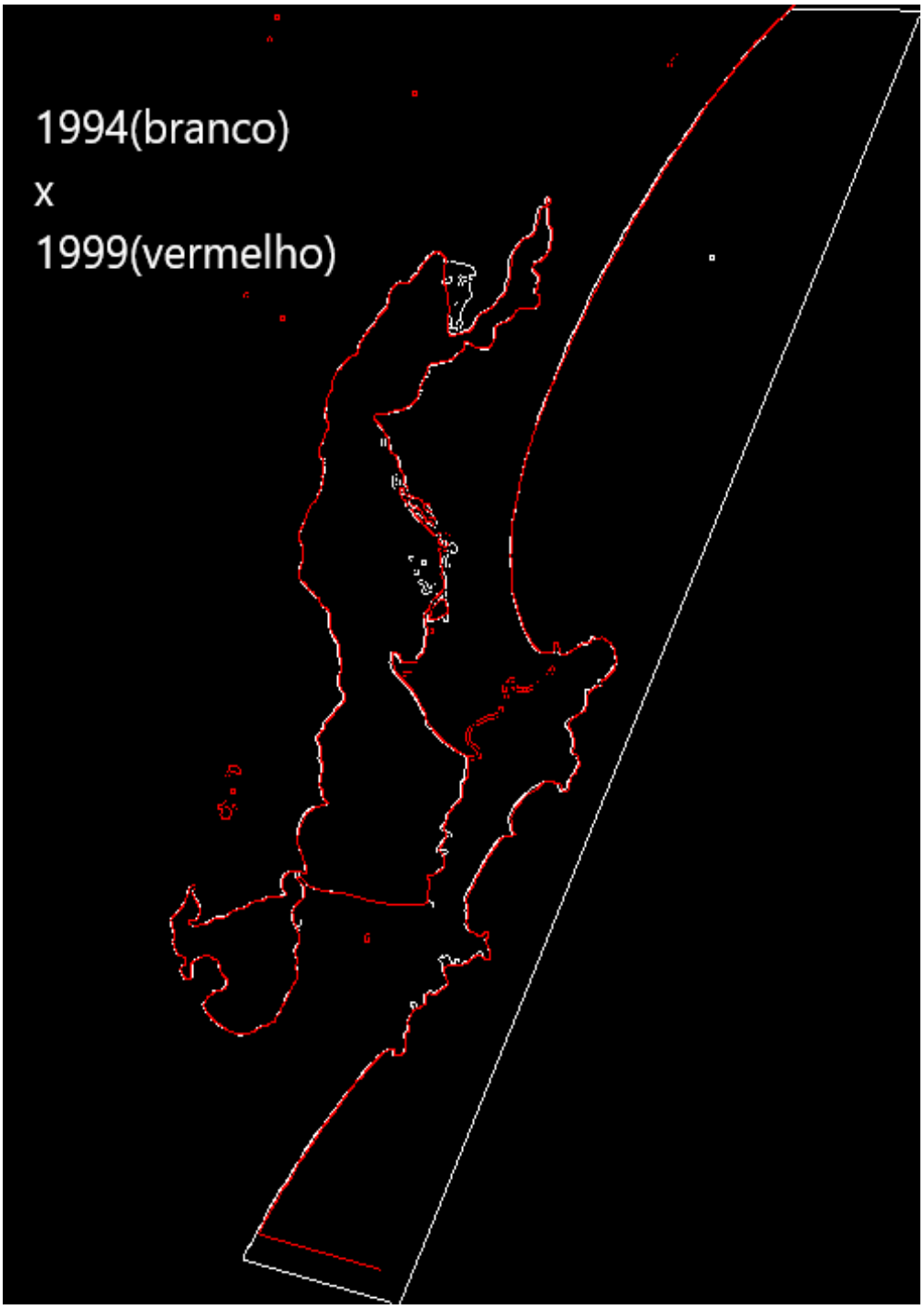


Figura 12 Comparação entre 1994 e 1999

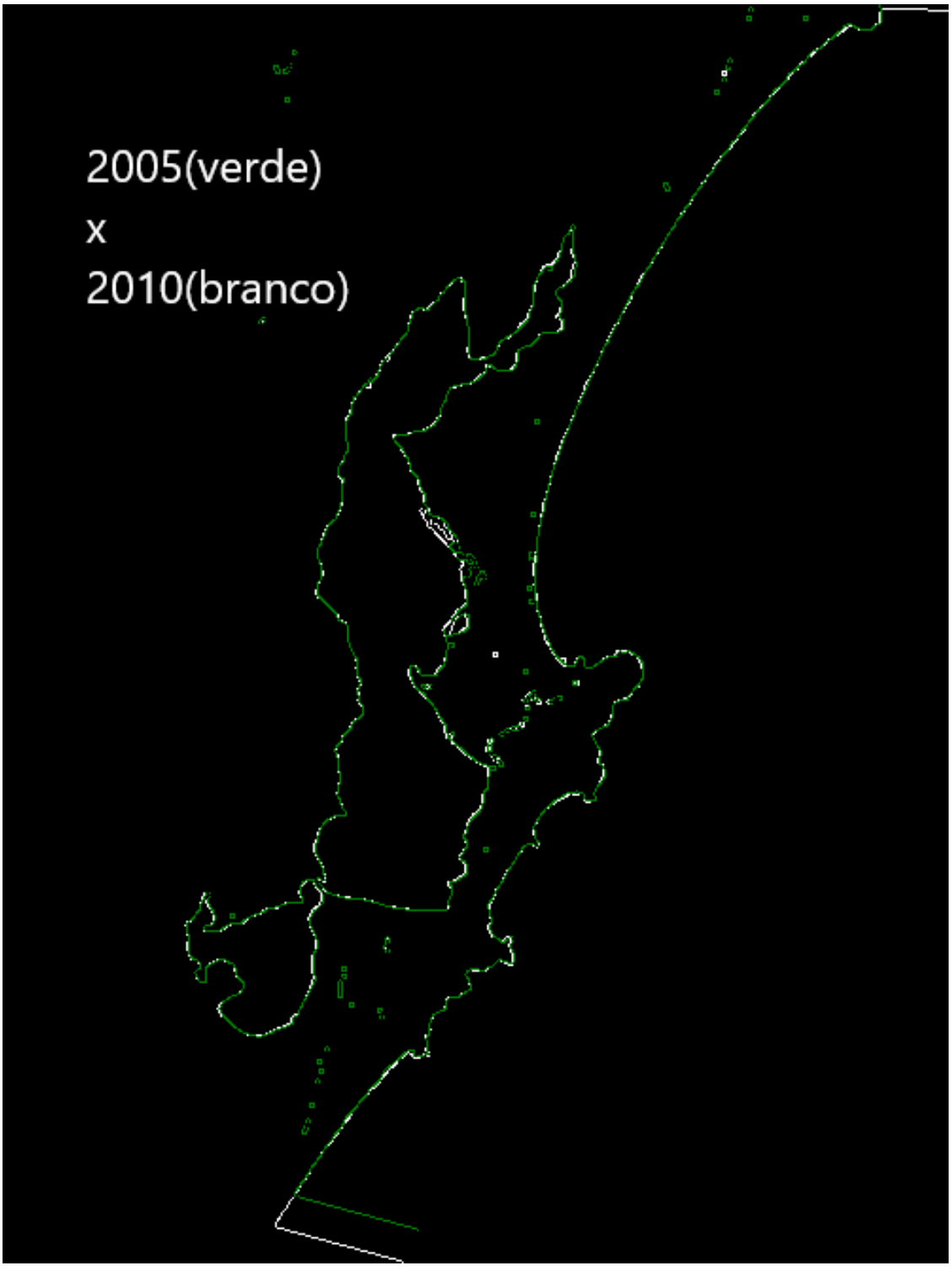


Figura 13 Comparação entre 2005 e 2010

Com a análise final das imagens comparativas, conseguimos perceber que não há aumento significativo tanto na erosão quanto no alagamento da Lagoa da Conceição, durante o período estudado (1986-2010), porém há lugares como as ilhas e atois, próximos dos bancos de areia na região da Barra da Lagoa (Figura 14), que segundo as imagens históricas, apresentam grande variabilidade no formato de suas costas, no entanto os bancos de areia submersos não foram encontrados.

Figura 14 Bancos de areia da Lago - Barra da Lagoa

7 REFERÊNCIAS

[1] OBSERVATÓRIO de Segurança e Saúde no Trabalho. [S. l.], mai 2022. [1] Moreira, A. ; Cynthia, R. ; Michael, V. APLICAÇÃO DE TÉCNICAS DE SENSORIAMENTO REMOTO NA ANÁLISE MULTITEMPORAL DA LINHA DE COSTA REGIÃO DE CAPUÍ/CE, ENTRE 1984 E 2013.. Disponível em: https://repositorio.ufc.br/bitstream/riufc/64398/1/2014_art_amdamasceno.pdf. Acesso em: 6 de outubro de 2022.

[2] Cláudio Ângelo da Silva Neto , Cynthia Romariz Duarte , Michael Vandesteen Silva Souto , Eduardo Viana Freires, Willamys Rangel Nunes de Sousa , Maykon Targino da Silva. Caracterização dos setores erosivos e deposicionais da linha de costa de Icapuí (CE) com base em produtos de sensoriamento remoto e técnicas de geoprocessamento .Disponível em: https://repositorio.ufc.br/bitstream/riufc/64962/1/2020_art_casilvaneto.pdf. Acesso em: 6 de outubro de 2022.

[3] Thaís Ferreira Conceição; Miguel da Guia Albuquerque; Jean Marcel de Almeida Espinoza. CARACTERIZAÇÃO DA LINHA DE COSTA ESTUARINA DA ÁREA URBANA DO MUNICÍPIO DE RIO GRANDE-RS. Disponível em: https://ocs.ige.unicamp.br/ojs/sbgfa/article/view/2167. Acesso em: 6 de outubro de 2022.

[4] LEONLENE DE SOUSA AGUIAR. RISCO POR INUNDAÇÃO COSTEIRA NA FOZ ESTUARINA DO RIO APODIMOSSORÓ/RN: APLICAÇÕES DE GEOTECNOLOGIAS E SIMULAÇÕES DE MUDANÇAS CLIMÁTICAS. Disponível em: https://repositorio.ufrn.br/bitstream/123456789/26917/1/Riscoinunda%c3%a7%c3%a3ocosteira_Aguiar_2018.pdf.Acesso em: 6 de outubro de 2022.

[5] Willamys Rangel Nunes de Sousa ; Michael Vandesteen Silva Souto; Stefanny Soares Matos; Cláudio Ângelo da Silva Neto; Cynthia Romariz Duarte. Extração automática de linhas de costa aplicada ao monitoramento de processos de erosão costeira .Disponível em: http://marte2.sid.inpe.br/ibi/8JMKD3MGP6W34M/3PSMCSC. Acesso em: 6 de outubro de 2022.

[6] Projetos do INPE. Disponível em: http://www.dpi.inpe.br/DPI/projetos. Acesso em: 6 de outubro de 2022.

[7] ARCHIVED: Unofficial Windows Binaries for Python Extension Packages. [S. l.], 26 jun. 2022. Disponível em: https://www.lfd.uci.edu/~gohlke/pythonlibs/#gdal. Acesso em: 23 nov. 2022.