1-exploratory-data-analysis

May 12, 2024

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
!pip install imagecodecs
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting imagecodecs
  Downloading
imagecodecs-2021.11.20-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(31.0 MB)
     |                  | 31.0 MB 1.2 MB/s
Requirement already satisfied: numpy>=1.16.5 in
/usr/local/lib/python3.7/dist-packages (from imagecodecs) (1.21.6)
Installing collected packages: imagecodecs
Successfully installed imagecodecs-2021.11.20
```

[ ]:
```
import cv2
import datetime
import gc
import glob
import math
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import skimage.morphology
import sys
import imagecodecs
import json
import tifffile as tiff
from matplotlib import colors
from matplotlib import pyplot as plt
from matplotlib.lines import Line2D
from matplotlib_venn import venn2_unweighted
```

[ ]:
```
plot_full_image = True

# Number of glomeruli to display for each image
num_glom_display = 5

# Number of glomberuli to save as tiff files.
num_glom_save = 5

glob_scale = 0.25
base_path = '/content/drive/MyDrive/kidneysegmentation/'
#Directory Contents
print("Directory Contents")
print('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%' )
print('\n'.join(os.listdir(base_path)))
print('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%' )
train_images = sorted(glob.glob(os.path.join(base_path, 'train/*.tiff')))
print(f'Number of training images: {len(train_images)}')
print('\n'.join(train_images))
print('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%' )

#Test Images
test_images = sorted(glob.glob(os.path.join(base_path, 'test/*.tiff')))
print(f'Number of test images: {len(test_images)}')
print('\n'.join(test_images))

df_train = pd.read_csv(os.path.join(base_path, 'train.csv'))
df_info = pd.read_csv(os.path.join(base_path,'HuBMAP-20-dataset_information.
  ↪csv'))
```

```
Directory Contents
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sample_submission.csv
train.csv
.ipynb_checkpoints
train
test
HuBMAP-20-dataset_information.csv
submission.csv
test.csv
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Number of training images: 15
/content/drive/MyDrive/kidneysegmentation/train/0486052bb.tiff
/content/drive/MyDrive/kidneysegmentation/train/095bf7a1f.tiff
/content/drive/MyDrive/kidneysegmentation/train/1e2425f28.tiff
/content/drive/MyDrive/kidneysegmentation/train/26dc41664.tiff
/content/drive/MyDrive/kidneysegmentation/train/2f6ecfcdf.tiff
/content/drive/MyDrive/kidneysegmentation/train/4ef6695ce.tiff
```

```
/content/drive/MyDrive/kidneysegmentation/train/54f2eec69.tiff
/content/drive/MyDrive/kidneysegmentation/train/8242609fa.tiff
/content/drive/MyDrive/kidneysegmentation/train/aaa6a05cc.tiff
/content/drive/MyDrive/kidneysegmentation/train/afa5e8098.tiff
/content/drive/MyDrive/kidneysegmentation/train/b2dc8411c.tiff
/content/drive/MyDrive/kidneysegmentation/train/b9a3865fc.tiff
/content/drive/MyDrive/kidneysegmentation/train/c68fe75ea.tiff
/content/drive/MyDrive/kidneysegmentation/train/cb2d976f4.tiff
/content/drive/MyDrive/kidneysegmentation/train/e79de561c.tiff
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Number of test images: 5
/content/drive/MyDrive/kidneysegmentation/test/2ec3f1bb9.tiff
/content/drive/MyDrive/kidneysegmentation/test/3589adb90.tiff
/content/drive/MyDrive/kidneysegmentation/test/57512b7f1.tiff
/content/drive/MyDrive/kidneysegmentation/test/aa05346ff.tiff
/content/drive/MyDrive/kidneysegmentation/test/d488c759a.tiff
```

[ ]:
```
import cv2
import glob
import json
import numpy as np
import os
import pandas as pd
import tifffile as tiff
from matplotlib import colors
from matplotlib import pyplot as plt
from matplotlib.lines import Line2D
from matplotlib_venn import venn2_unweighted
```

[ ]:
```
train_images = list(map(lambda x: os.path.basename(x), train_images))
test_images = list(map(lambda x: os.path.basename(x), test_images))
```

[ ]:
```
pd.set_option('display.max_colwidth', None)
dataset_information = pd.read_csv(base_path + 'HuBMAP-20-dataset_information.
 ↪csv')
dataset_information['pixels_total'] = dataset_information.width_pixels *␣
 ↪dataset_information.height_pixels
```

[ ]:
```
train_dataset_information =␣
 ↪dataset_information[dataset_information['image_file'].isin(train_images)].
 ↪reset_index(drop=True)
test_dataset_information =␣
 ↪dataset_information[dataset_information['image_file'].isin(test_images)].
 ↪reset_index(drop=True)
```

0.1 Training Dataset

Image resolution table and bar chart representation

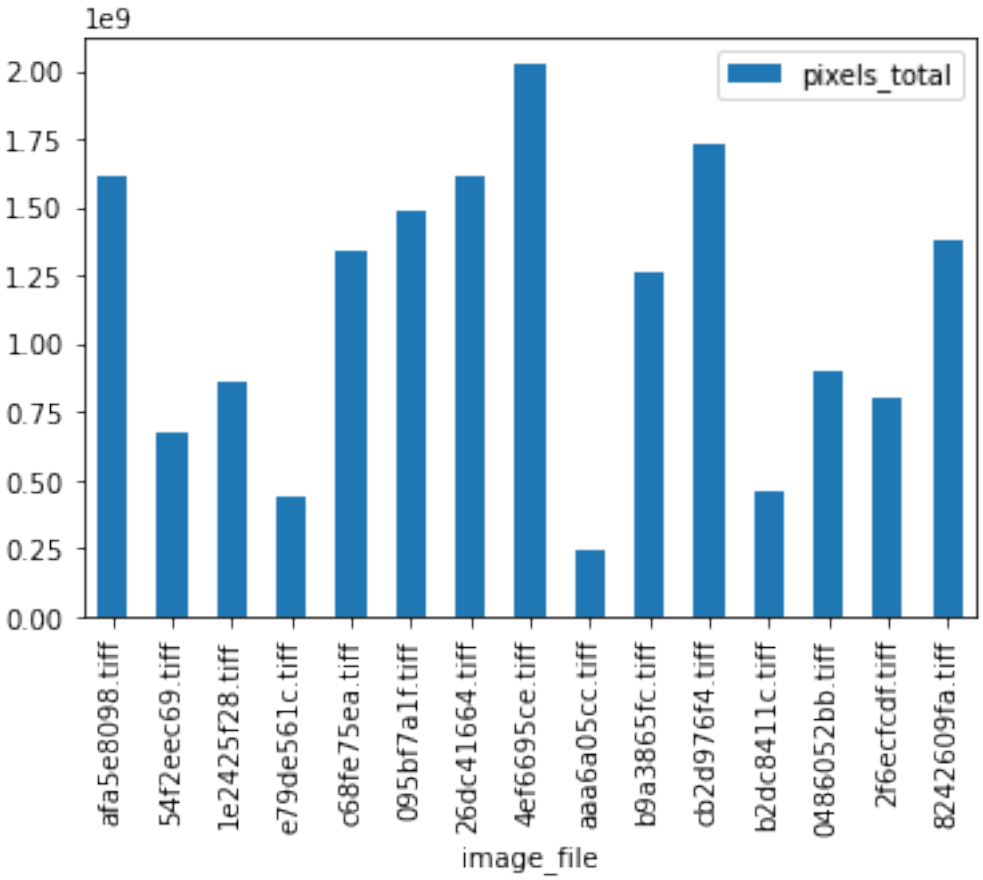
[ ]:
```
train_dataset_information.sort_values('pixels_total',␣
↪ascending=False)[['image_file', 'width_pixels','height_pixels']].
↪reset_index(drop=True)
```

[ ]:

	image_file	width_pixels	height_pixels
0	4ef6695ce.tiff	50680	39960
1	cb2d976f4.tiff	49548	34940
2	26dc41664.tiff	42360	38160
3	afa5e8098.tiff	43780	36800
4	095bf7a1f.tiff	39000	38160
5	8242609fa.tiff	44066	31299
6	c68fe75ea.tiff	49780	26840
7	b9a3865fc.tiff	40429	31295
8	0486052bb.tiff	34937	25784
9	1e2425f28.tiff	32220	26780
10	2f6ecfcdf.tiff	25794	31278
11	54f2eec69.tiff	22240	30440
12	b2dc8411c.tiff	31262	14844
13	e79de561c.tiff	27020	16180
14	aaa6a05cc.tiff	13013	18484

[ ]:
```
train_dataset_information.plot.bar(x='image_file', y='pixels_total', rot=90)
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7efe299fc510>
```

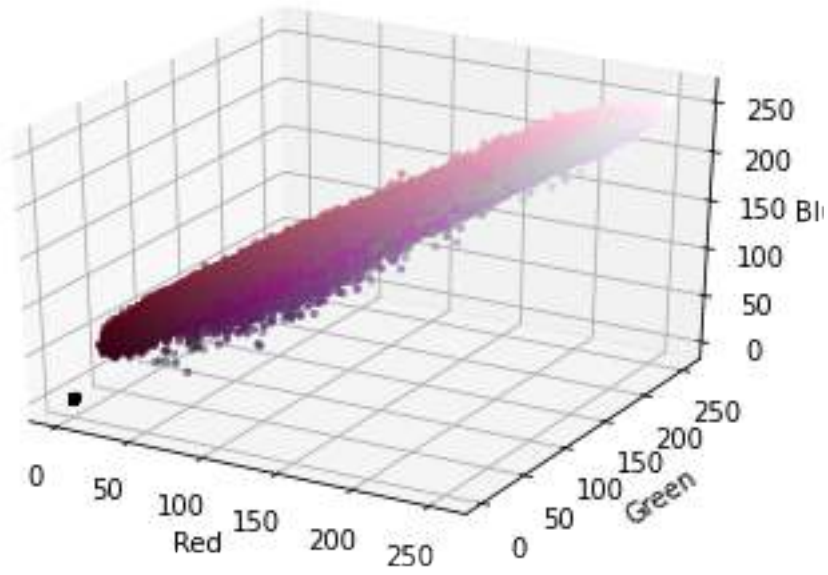


RGB and HSV color spaces of aaa6a05cc.tiff

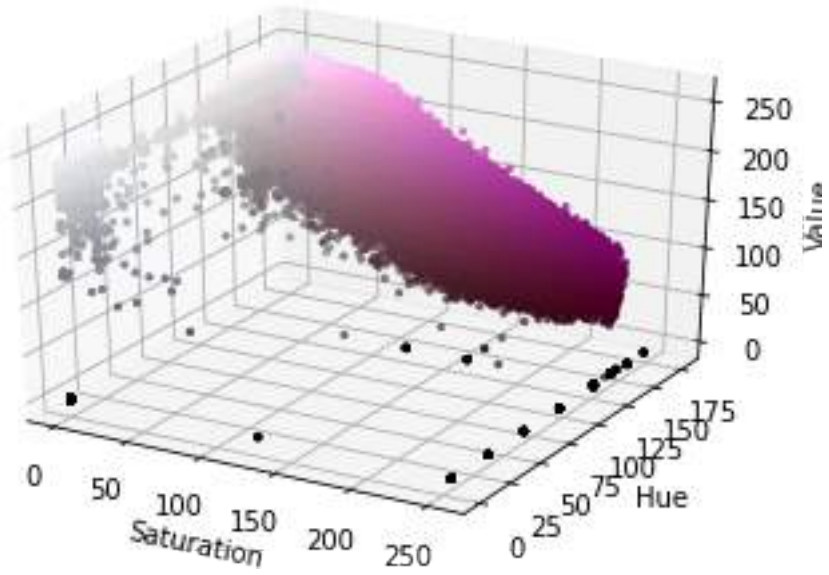
[ ]:
```
# open and resize image
image = cv2.imread(base_path + 'train/aaa6a05cc.tiff')
image_resize = cv2.resize(image,(image.shape[1]//10,image.shape[0]//10),␣
 ↪interpolation = cv2.INTER_CUBIC)
```

[ ]:
```
# calculate colors
pixel_colors = image_resize.reshape((np.shape(image_resize)[0]*np.
 ↪shape(image_resize)[1], 3))
norm = colors.Normalize(vmin=-1.,vmax=1.)
norm.autoscale(pixel_colors)
pixel_colors = norm(pixel_colors).tolist()

# split channels
b, g, r = cv2.split(image_resize)

# scatter plot
fig = plt.figure()
```

```
axis = fig.add_subplot(1, 1, 1, projection='3d')
axis.scatter(r.flatten(), g.flatten(), b.flatten(), facecolors=pixel_colors,␣
 ↪marker='.')
axis.set_xlabel('Red')
axis.set_ylabel('Green')
axis.set_zlabel('Blue')
plt.show()
```



[ ]:
```
# convert to hsv
hsv_image = cv2.cvtColor(image_resize, cv2.COLOR_BGR2HSV)
h, s, v = cv2.split(hsv_image)

# scatter plot
fig = plt.figure()
axis = fig.add_subplot(1, 1, 1, projection='3d')
axis.scatter(s.flatten(), h.flatten(), v.flatten(), facecolors=pixel_colors,␣
 ↪marker='.')
axis.set_xlabel('Saturation')
axis.set_ylabel('Hue')
axis.set_zlabel('Value')
plt.show()
```



Glomeruli Count

[ ]:
```
train_glom_seg_files =␣
 ↪train_dataset_information['glomerulus_segmentation_file'].to_list()
train_glomeruli_dict = {}

for file_name in train_glom_seg_files:
    file_id = file_name[:9]
    with open(f'/content/drive/MyDrive/kidneysegmentation/train/{file_name}')␣
 ↪as json_file:
        data = json.load(json_file)
        train_glomeruli_dict[file_id] = 0
        for entry in data:
            if entry['type'] == 'Feature' and␣
 ↪entry['properties']['classification']['name'] == 'glomerulus':
                    train_glomeruli_dict[file_id] += 1
            else:
                raise Exception(f"Unexpected json format: {entry['type']},␣
 ↪{entry['properties']['classification']['name']}")

train_nr_glom = pd.DataFrame(list(train_glomeruli_dict.items()),␣
 ↪columns=['file_id', 'nr_glomeruli'])
train_nr_glom.sort_values('nr_glomeruli', ascending=False).
 ↪reset_index(drop=True)
```

[ ]:

	file_id	nr_glomeruli
0	8242609fa	586

```
1   b9a3865fc          469
2   4ef6695ce          439
3   095bf7a1f          350
4   cb2d976f4          319
5   26dc41664          245
6   afa5e8098          235
7   e79de561c          180
8   1e2425f28          178
9   2f6ecfcdf          160
10  54f2eec69          139
11  b2dc8411c          138
12  0486052bb          130
13  c68fe75ea          118
14  aaa6a05cc           99
```

[ ]:
```
#total count
train_nr_glom['nr_glomeruli'].sum()
```

[ ]:
```
3785
```

Glomeruli Size Distribution

[ ]:
```
train_glom_seg_files =␣
 ↪train_dataset_information['glomerulus_segmentation_file'].to_list()
train_glomeruli_polys_dict = {}

for file_name in train_glom_seg_files:
    with open(f'/content/drive/MyDrive/kidneysegmentation/train/{file_name}')␣
 ↪as json_file:
        data = json.load(json_file)
        train_glomeruli_polys_dict[file_name] = []
        for entry in data:
            if entry['type'] == 'Feature' and␣
 ↪entry['properties']['classification']['name'] == 'glomerulus':
                geom = np.array(entry['geometry']['coordinates']).astype(np.
 ↪float32)
                x,y,w,h = cv2.boundingRect(geom.squeeze(axis=0))
                train_glomeruli_polys_dict[file_name].append((h,w)) # height,␣
 ↪width!
            else:
                raise Exception(f"Unexpected json format: {entry['type']},␣
 ↪{entry['properties']['classification']['name']}")

train_res_glom = pd.DataFrame(list(train_glomeruli_polys_dict.items()),␣
 ↪columns=['glomerulus_segmentation_file', 'glomeruli_height_width'])
train_res_glom = train_res_glom.explode('glomeruli_height_width')
```

```
train_res_glom['height'], train_res_glom['width'] =␣
 ↪zip(*train_res_glom['glomeruli_height_width'])
train_res_glom = train_res_glom.drop(columns=['glomeruli_height_width'])
train_res_glom.describe()
```

[ ]:

	height	width
count	3785.000000	3785.000000
mean	322.601057	325.266579
std	84.198179	86.372246
min	94.000000	103.000000
25%	262.000000	263.000000
50%	318.000000	319.000000
75%	383.000000	385.000000
max	626.000000	659.000000

Anatomical Structure

[ ]:
```
def list_structures(seg_files, folder):
    train_anatomical_dict = {}
    folder_path = os.path.join(base_path, folder)

    for file_name in seg_files:
        file_id = file_name[:9]
        with open(os.path.join(folder_path, file_name)) as json_file:
            data = json.load(json_file)
            train_anatomical_dict[file_id] = []
            for entry in data:
                if entry['type'] == 'Feature':
                    train_anatomical_dict[file_id].
 ↪append(entry['properties']['classification']['name'])
                else:
                    raise Exception(f"Unexpected json format: {entry['type']},␣
 ↪{entry['properties']['classification']['name']}")

    return pd.DataFrame(list(train_anatomical_dict.items()),␣
 ↪columns=['file_id', 'anatomical_structure'])
```

[ ]:
```
train_anatomical_seg_files =␣
 ↪train_dataset_information['anatomical_structures_segmention_file'].to_list()
train_anatomical = list_structures(train_anatomical_seg_files, 'train')
train_anatomical
```

[ ]:

	file_id	anatomical_structure
0	afa5e8098	[Outer Medulla, Outer Stripe, Cortex, Inner medulla]
1	54f2eec69	[Medulla, Cortex]
2	1e2425f28	[Cortex, Medulla]
3	e79de561c	[Cortex]

```
4   c68fe75ea                                          [Cortex, Medulla]
5   095bf7a1f                                                  [Cortex]
6   26dc41664  [Cortex, Outer Stripe, Inner medulla, Outer Medulla]
7   4ef6695ce                                          [Medulla, Cortex]
8   aaa6a05cc                                          [Medulla, Cortex]
9   b9a3865fc                                          [Cortex, Medulla]
10  cb2d976f4                                [Cortex, Cortex, Medulla]
11  b2dc8411c                                [Cortex, Cortex, Medulla]
12  0486052bb                                          [Medulla, Cortex]
13  2f6ecfcdf                                          [Medulla, Cortex]
14  8242609fa                                          [Medulla, Cortex]
```

[ ]:
```
#listing all unique anatomical structures
pd.DataFrame(train_anatomical.
 ↪explode('anatomical_structure')['anatomical_structure'].unique(),␣
 ↪columns=['unique_structures'])
```

[ ]:
```
  unique_structures
0     Outer Medulla
1      Outer Stripe
2            Cortex
3     Inner medulla
4           Medulla
```

Mask Visualization

[ ]:
```
def make_grid(shape, window=256, min_overlap=32):
    """
    function to generate a grid layout for sliding window
    :param shape: height and width of the image
    :param window: size of the window
    :param min_overlap: minimal window overlap
    :return: array of window coordinates (x1,x2,y1,y2)
    """
    x, y = shape
    nx = x // (window - min_overlap) + 1
    x1 = np.linspace(0, x, num=nx, endpoint=False, dtype=np.int64)
    x1[-1] = x - window
    x2 = (x1 + window).clip(0, x)
    ny = y // (window - min_overlap) + 1
    y1 = np.linspace(0, y, num=ny, endpoint=False, dtype=np.int64)
    y1[-1] = y - window
    y2 = (y1 + window).clip(0, y)
    slices = np.zeros((nx,ny, 4), dtype=np.int64)

    for i in range(nx):
        for j in range(ny):
```

```
            slices[i,j] = x1[i], x2[i], y1[j], y2[j]
    return slices.reshape(nx*ny,4)
```

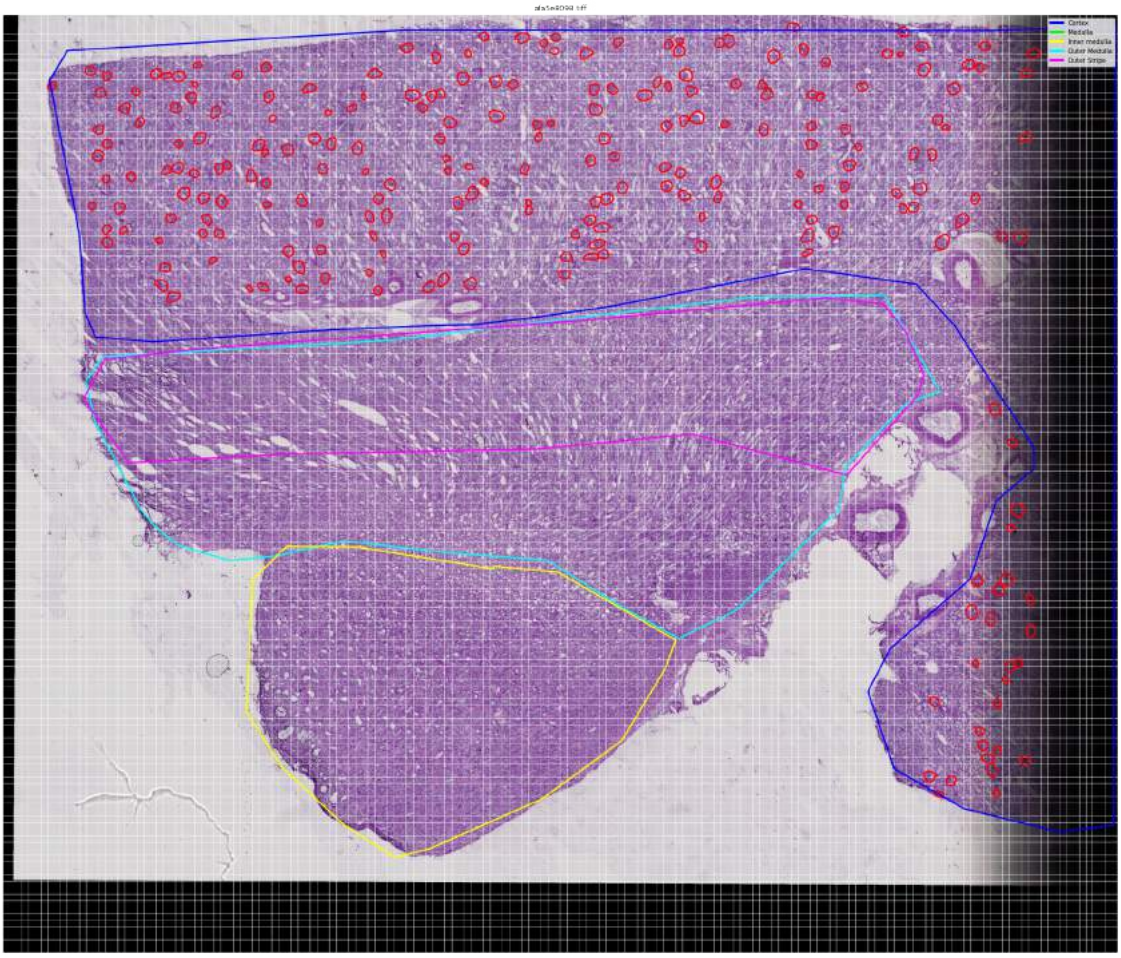
[ ]:
```
def plot_masks(dataset_information, folder, frame_size, frame_overlap,␣
↪plot_glom):
    folder_path = os.path.join(base_path, folder)

    for i in range(len(dataset_information)):

        # create new figure
        plt.figure(figsize=(32, 30))
        obj_line_thickness = 60

        # find metadata row for json file
        image_metadata = dataset_information.iloc[i]

        # open image file
        image = tiff.imread(os.path.join(folder_path,␣
↪image_metadata['image_file']))
        print(image_metadata['image_file'])

        # reshape image if necessary
        if len(image.shape) == 5:
            image = image.squeeze()
        if image.shape[0] == 3:
            image = image.transpose(1, 2, 0)

        # create a copy of the image
        image = image.copy()

        # draw sliding window boxes
        frame_grid = make_grid((image.shape[1], image.shape[0]), frame_size,␣
↪frame_overlap)

        for frame in frame_grid:
            x1, y1 = frame[0], frame[2]
            x2, y2 = frame[1], frame[3]
            image = cv2.rectangle(image, (x1, y1), (x2, y2),␣
↪color=(255,255,255), thickness=16)

        # draw glomeruli polygons
        if plot_glom:
            # open glomeruli json file
            read_glom_seg_file = open(os.path.join(folder_path,␣
↪image_metadata['glomerulus_segmentation_file']), 'r')
            glom_seg_data = json.load(read_glom_seg_file)
```

```
            for k in range(len(glom_seg_data)):
                glom_poly = np.
↪array(glom_seg_data[k]['geometry']['coordinates']).astype(np.int32) # get␣
↪coordinates of glomeruli
                cv2.polylines(image, glom_poly, True,(255,0,0),␣
↪thickness=obj_line_thickness)

        # open anatomical json file
        read_anatomical_seg_file = open(os.path.join(folder_path,␣
↪image_metadata['anatomical_structures_segmention_file']), 'r')
        anatomical_seg_data = json.load(read_anatomical_seg_file)

        # scan anatomical json file and draw lines
        for n in range(len(anatomical_seg_data)):
            obj_name =␣
↪anatomical_seg_data[n]['properties']['classification']['name']
            obj_coords = anatomical_seg_data[n]['geometry']['coordinates']

            if (obj_name == 'Cortex'): # draw line around cortex
                cv2.polylines(image, np.expand_dims(np.array(obj_coords[0]).
↪astype(np.int32), axis=0), True, (0,0,255), thickness=obj_line_thickness)
            elif (obj_name == 'Medulla'): # draw line around medulla
                cv2.polylines(image, np.array(obj_coords).astype(np.int32),␣
↪True, (0,255,0), thickness=obj_line_thickness)
            elif (obj_name == 'Inner medulla'): # draw line around inner medulla
                cv2.polylines(image, np.array(obj_coords).astype(np.int32),␣
↪True, (255,255,0), thickness=obj_line_thickness)
            elif (obj_name == 'Outer Medulla'): # draw line around outer medulla
                cv2.polylines(image, np.array(obj_coords).astype(np.int32),␣
↪True, (0,255,255), thickness=obj_line_thickness)
            elif (obj_name == 'Outer Stripe'): # draw line around outer stripe
                cv2.polylines(image, np.array(obj_coords).astype(np.int32),␣
↪True, (255,0,255), thickness=obj_line_thickness)
            else:
                raise Exception(f'Unknown anatomical object: {obj_name}')

        # down-scale the image
        image_resize = cv2.resize(image,(image.shape[1]//10,image.shape[0]//
↪10), interpolation = cv2.INTER_CUBIC)

        # add legend and view the image
        custom_lines = [Line2D([0], [0], color=(0.,0.,1.), lw=4),
                        Line2D([0], [0], color=(0.,1.,0.), lw=4),
                        Line2D([0], [0], color=(1.,1.,0.), lw=4),
                        Line2D([0], [0], color=(0.,1.,1.), lw=4),
                        Line2D([0], [0], color=(1.,0.,1.), lw=4),]
```

```
        op_file = "output_" + image_metadata['image_file'] + ".tiff"
        plt.legend(custom_lines, ['Cortex', 'Medulla', 'Inner medulla', 'Outer␣
↪Medulla', 'Outer Stripe'])
        plt.axis('off')
        plt.title(image_metadata['image_file'])
        plt.imshow(image_resize)
        plt.show()
        plt.savefig(op_file)
```

[ ]:
```
plot_masks(train_dataset_information, 'train', 1024, 256, True)
```

```
afa5e8098.tiff
```

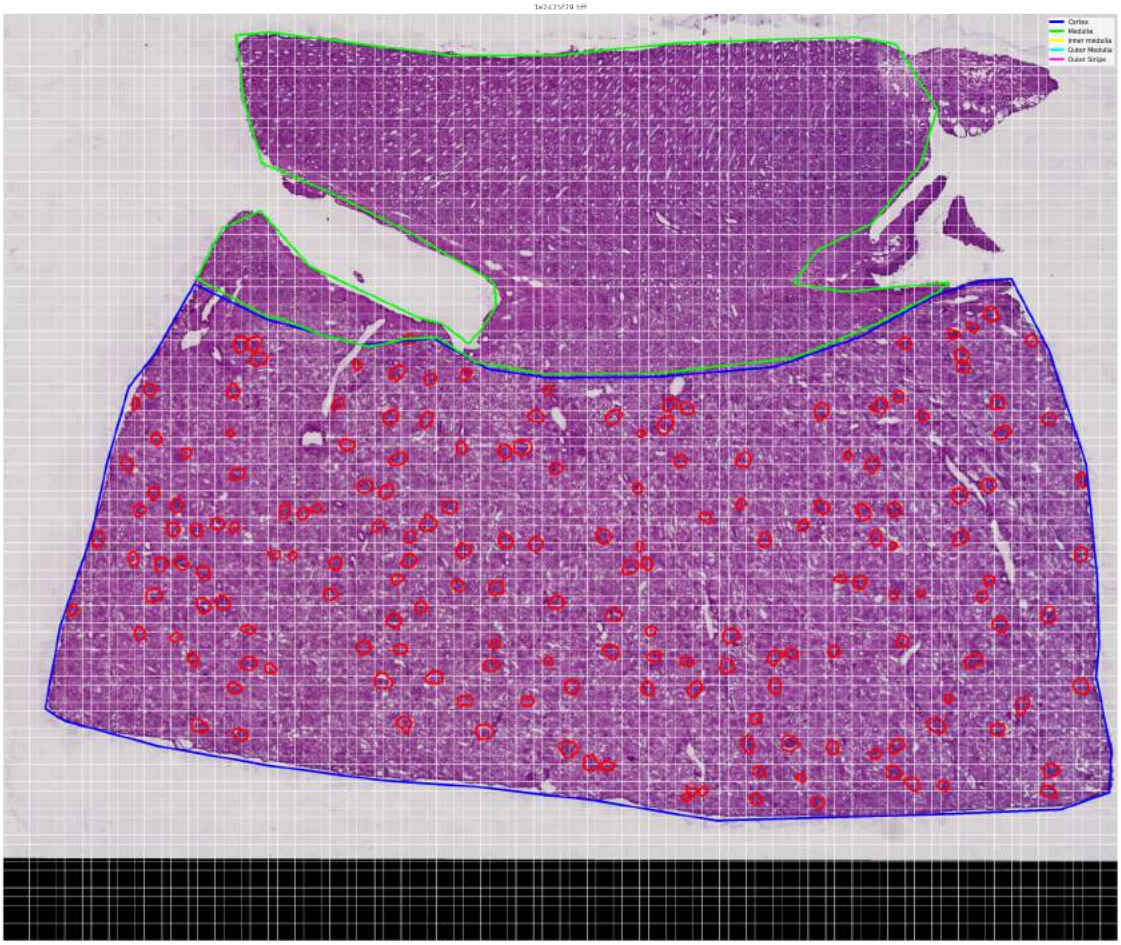


```
54f2eec69.tiff
```

```
<Figure size 432x288 with 0 Axes>
```

1e2425f28.tiff

<Figure size 432x288 with 0 Axes>

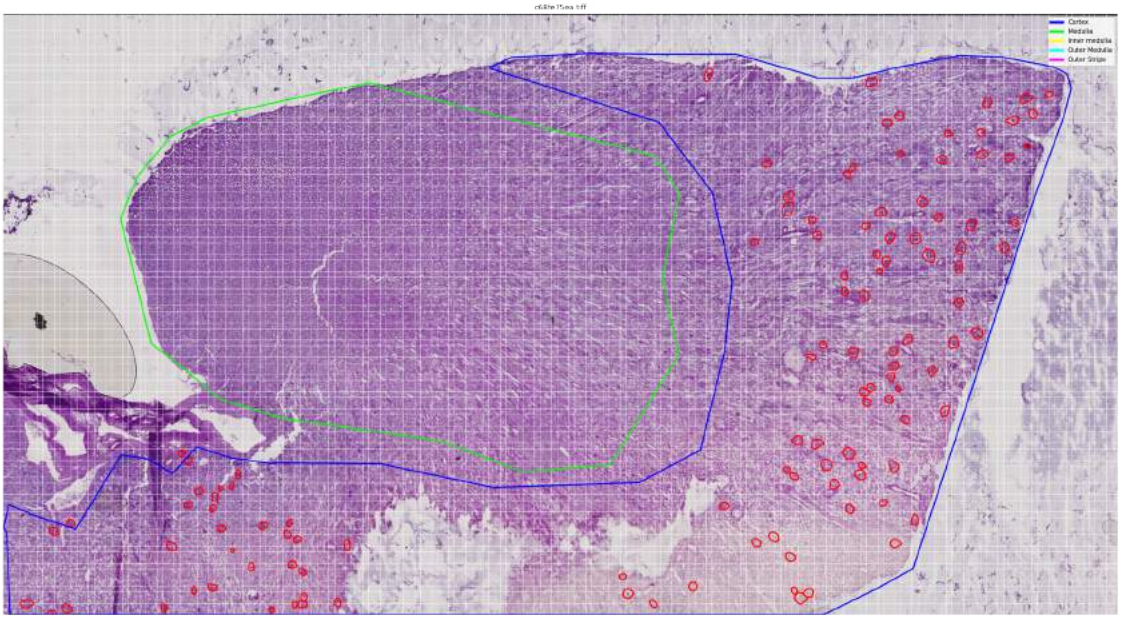


e79de561c.tiff

<Figure size 432x288 with 0 Axes>

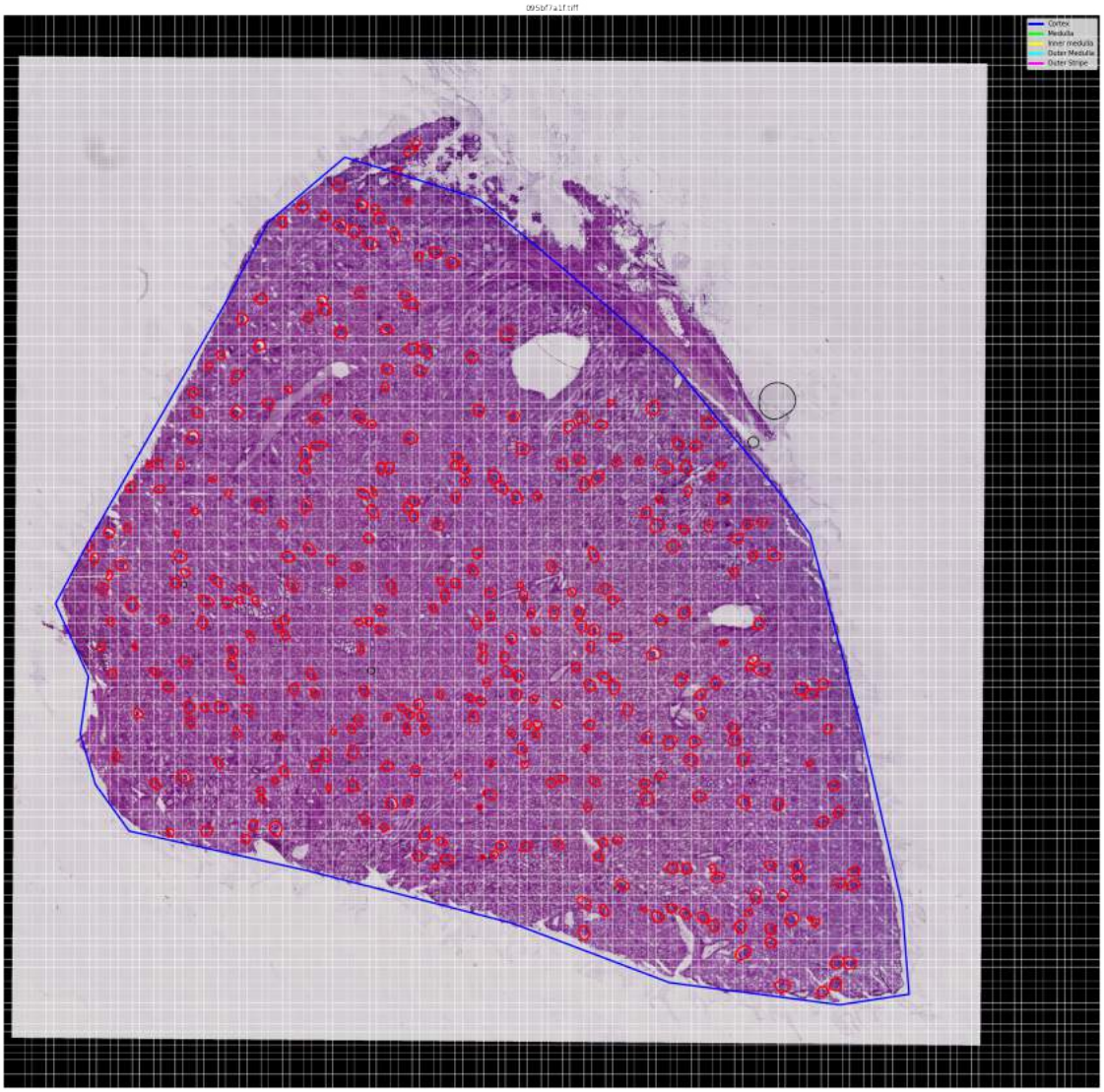
c68fe75ea.tiff

<Figure size 432x288 with 0 Axes>



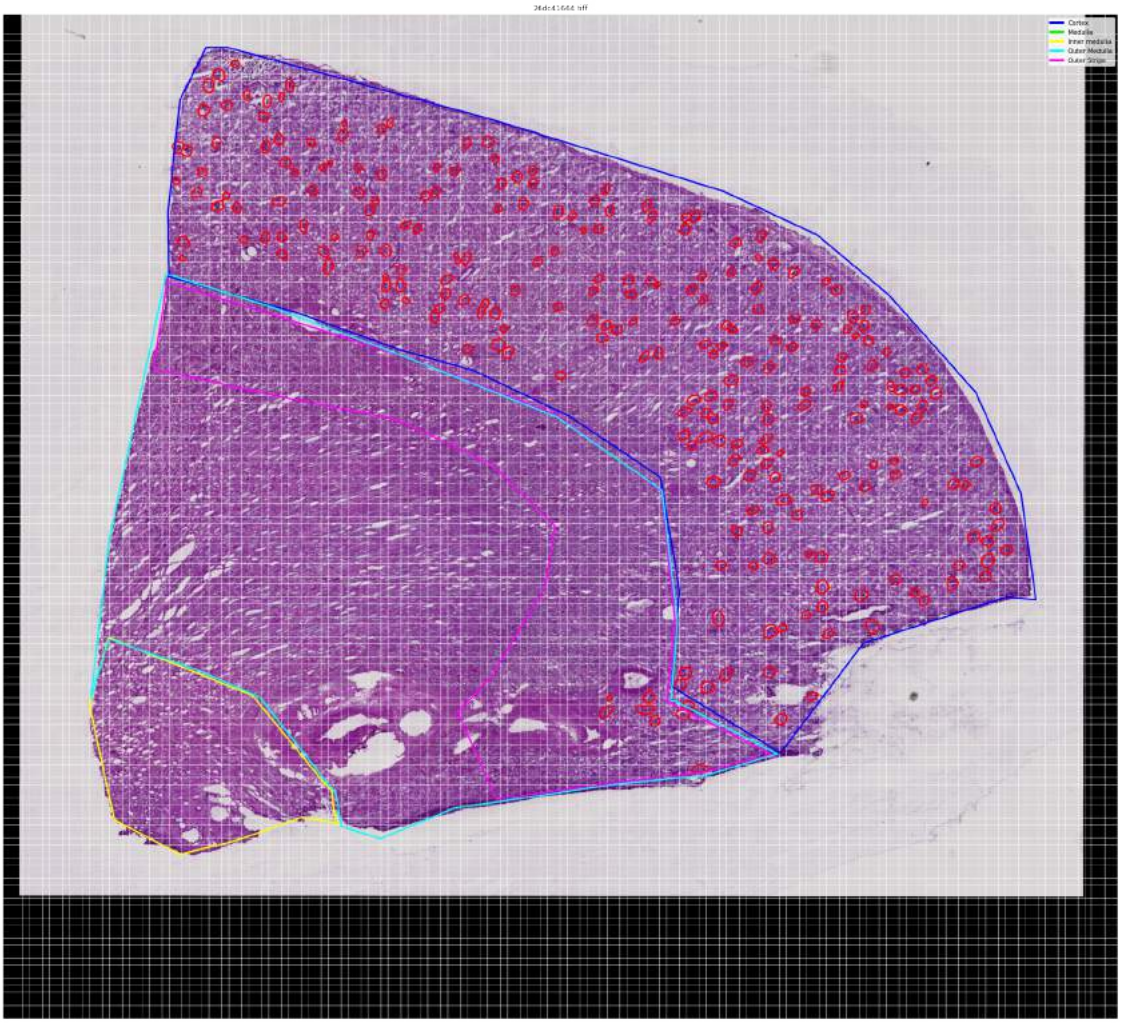
095bf7a1f.tiff

<Figure size 432x288 with 0 Axes>



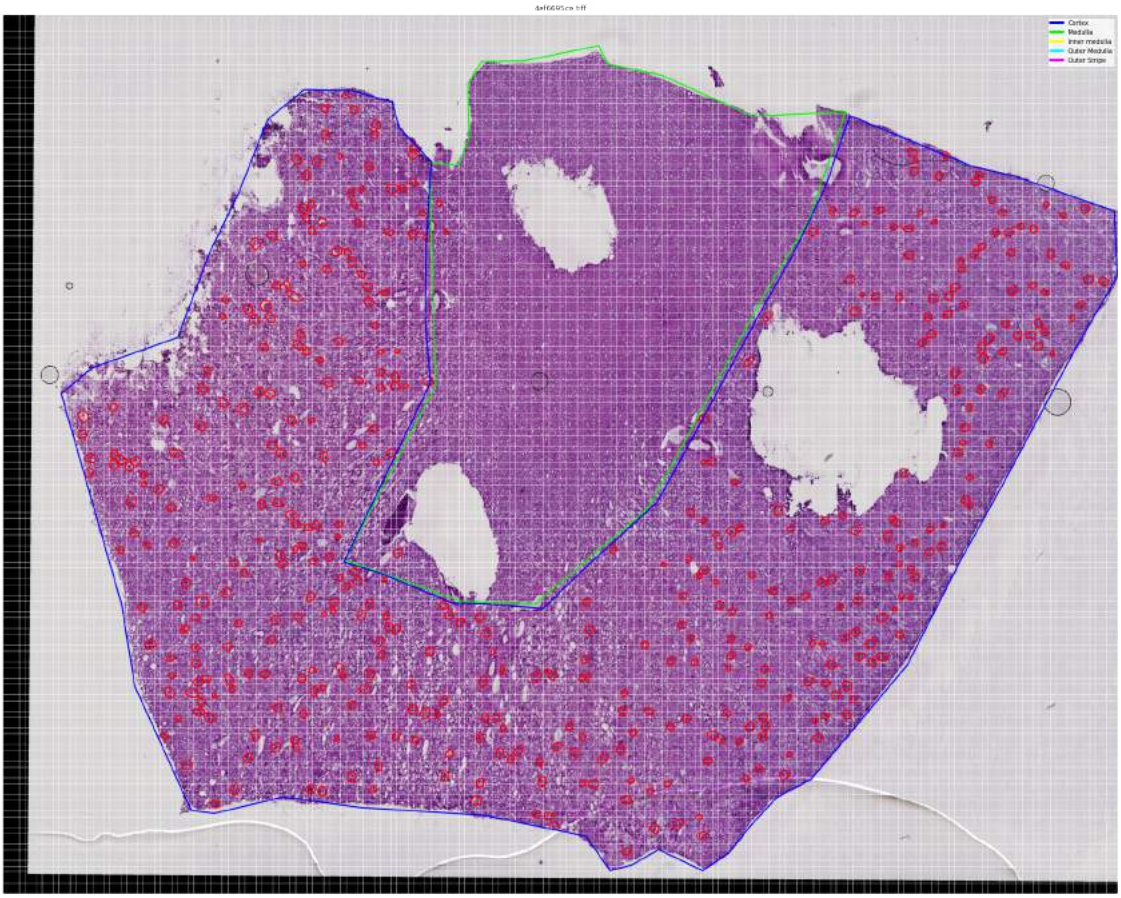
26dc41664.tiff

<Figure size 432x288 with 0 Axes>



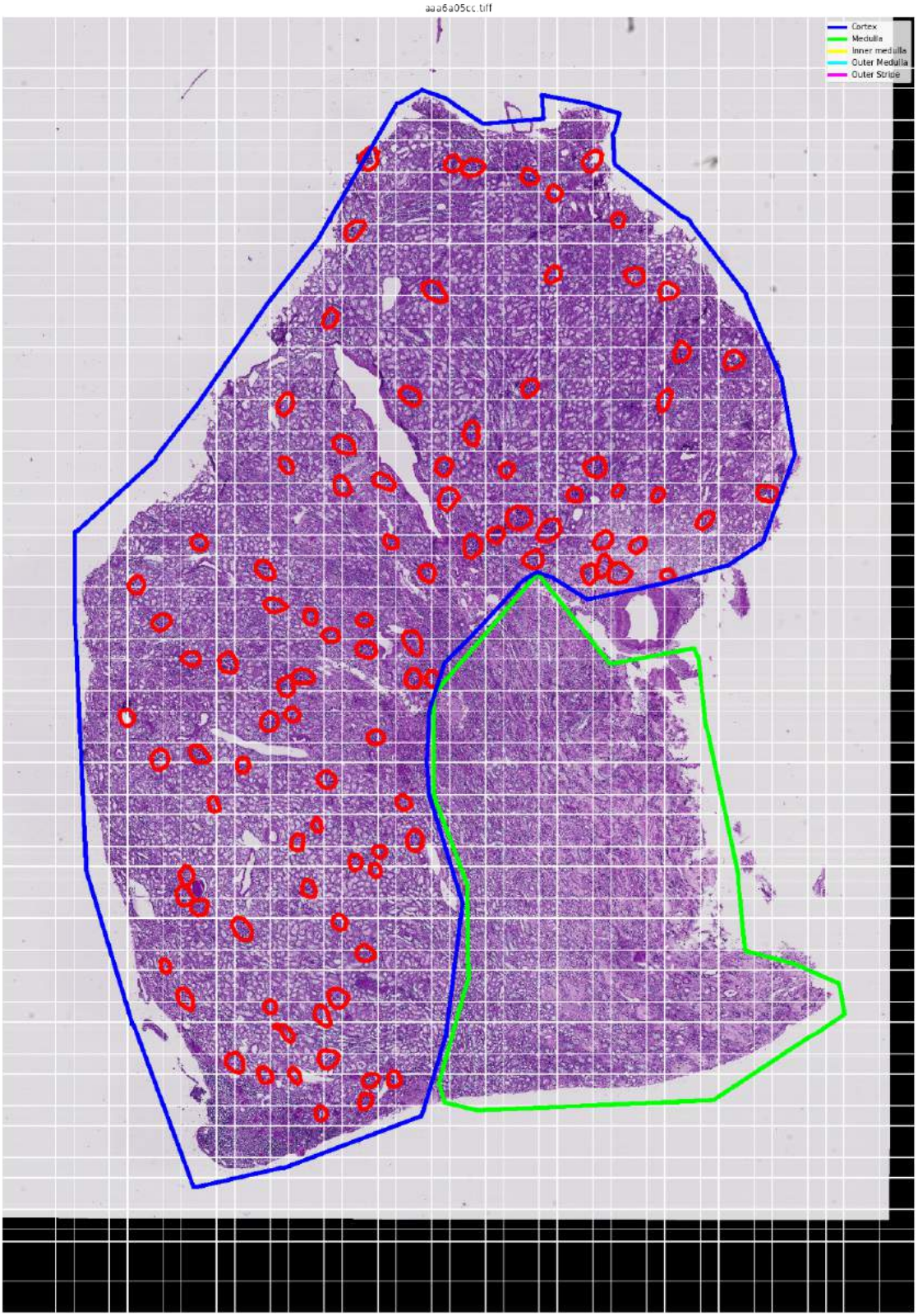
4ef6695ce.tiff

<Figure size 432x288 with 0 Axes>



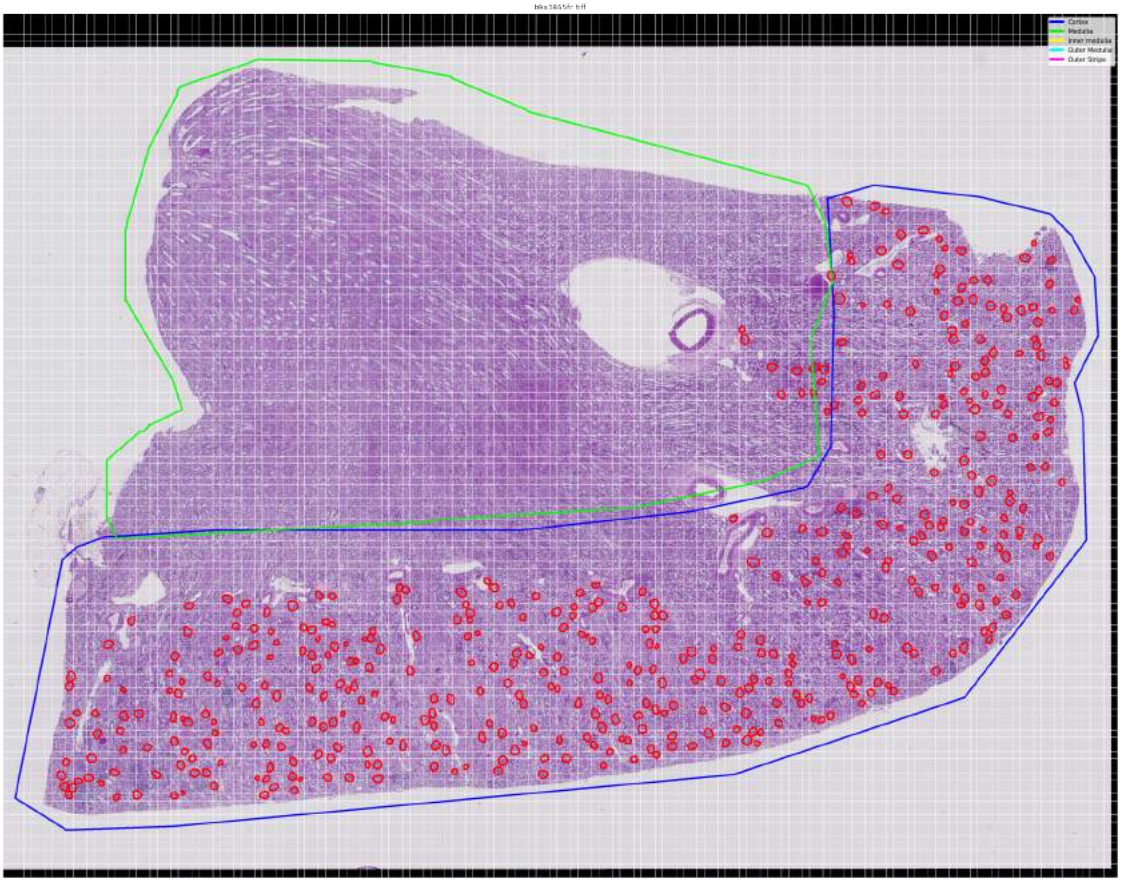
aaa6a05cc.tiff

<Figure size 432x288 with 0 Axes>


aaa6a05cc.tiff
Cortex
Medulla
Inner medulla
Outer Medulla
Outer Stripe

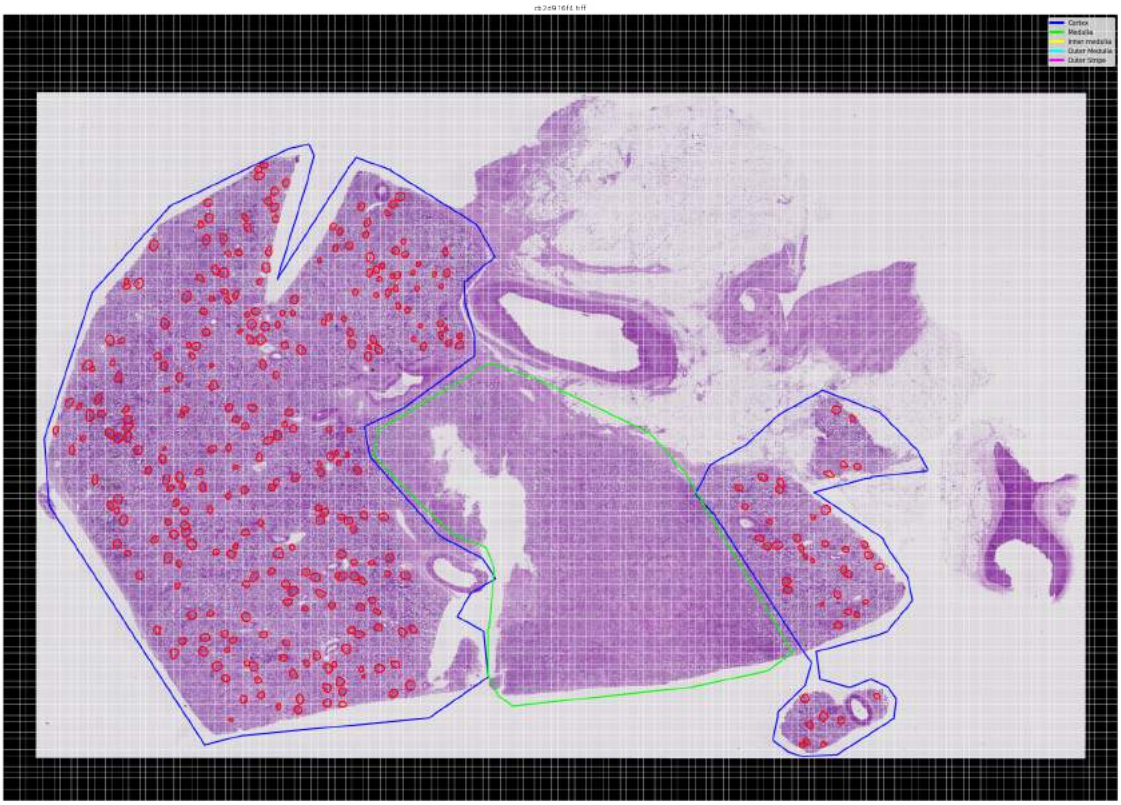
b9a3865fc.tiff

<Figure size 432x288 with 0 Axes>



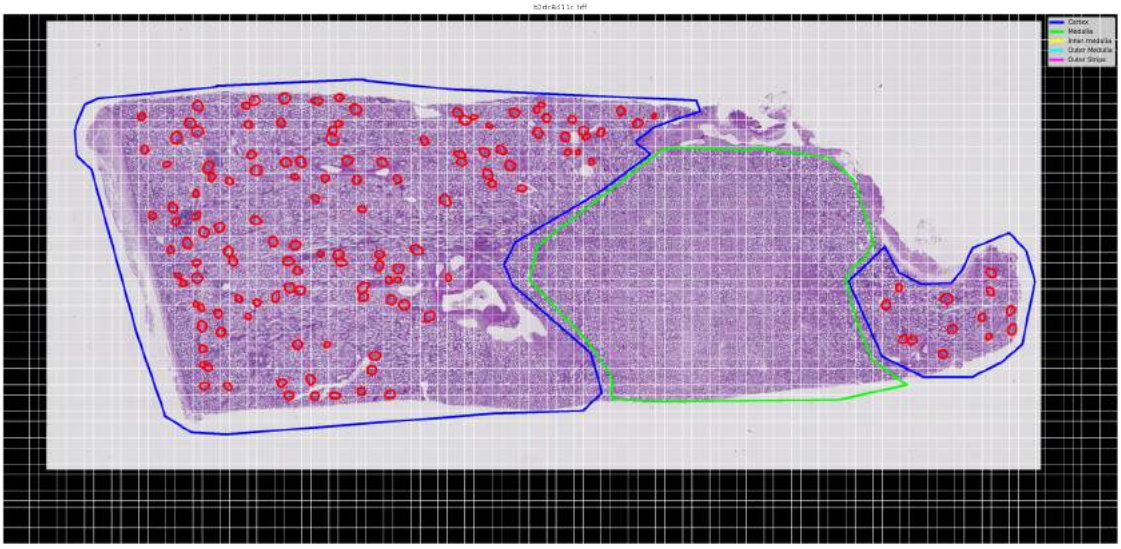
cb2d976f4.tiff

<Figure size 432x288 with 0 Axes>



b2dc8411c.tiff

<Figure size 432x288 with 0 Axes>



```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-15-b5441024a68b> in <module>()
----> 1 plot_masks(train_dataset_information, 'train', 1024, 256, True)

<ipython-input-14-c87f6c46683c> in plot_masks(dataset_information, folder,␣
 ↪frame_size, frame_overlap, plot_glom)
     12
     13         # open image file
---> 14         image = tiff.imread(os.path.join(folder_path,␣
 ↪image_metadata['image_file']))
     15         print(image_metadata['image_file'])
     16

/usr/local/lib/python3.7/dist-packages/tifffile/tifffile.py in imread(files,␣
 ↪aszarr, **kwargs)
    892                 if aszarr:
    893                     return tif.aszarr(**kwargs)
--> 894                 return tif.asarray(**kwargs)
    895
    896     with TiffSequence(files, **kwargs_seq) as imseq:

/usr/local/lib/python3.7/dist-packages/tifffile/tifffile.py in asarray(self,␣
 ↪key, series, level, squeeze, out, maxworkers)
   3372                 )
   3373         elif len(pages) == 1:
-> 3374             result = pages[0].asarray(out=out, maxworkers=maxworkers)
   3375         else:
   3376             result = stack_pages(pages, out=out, maxworkers=maxworkers)

/usr/local/lib/python3.7/dist-packages/tifffile/tifffile.py in asarray(self,␣
 ↪out, squeeze, lock, maxworkers)
   6815         else:
   6816             # decode individual strips or tiles
-> 6817             result = create_output(out, keyframe.shaped, keyframe._dtyp )
   6818             keyframe.decode  # init TiffPage.decode function
   6819

/usr/local/lib/python3.7/dist-packages/tifffile/tifffile.py in␣
 ↪create_output(out, shape, dtype, mode, suffix, fillvalue)
  16911             out[:] = fillvalue
  16912             return out
> 16913         return numpy.zeros(shape, dtype)
  16914     if isinstance(out, numpy.ndarray):
  16915         if product(shape) != product(out.shape):
```

```
KeyboardInterrupt:
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 2304x2160 with 0 Axes>
```

0.2 Test Data

Image Resolution

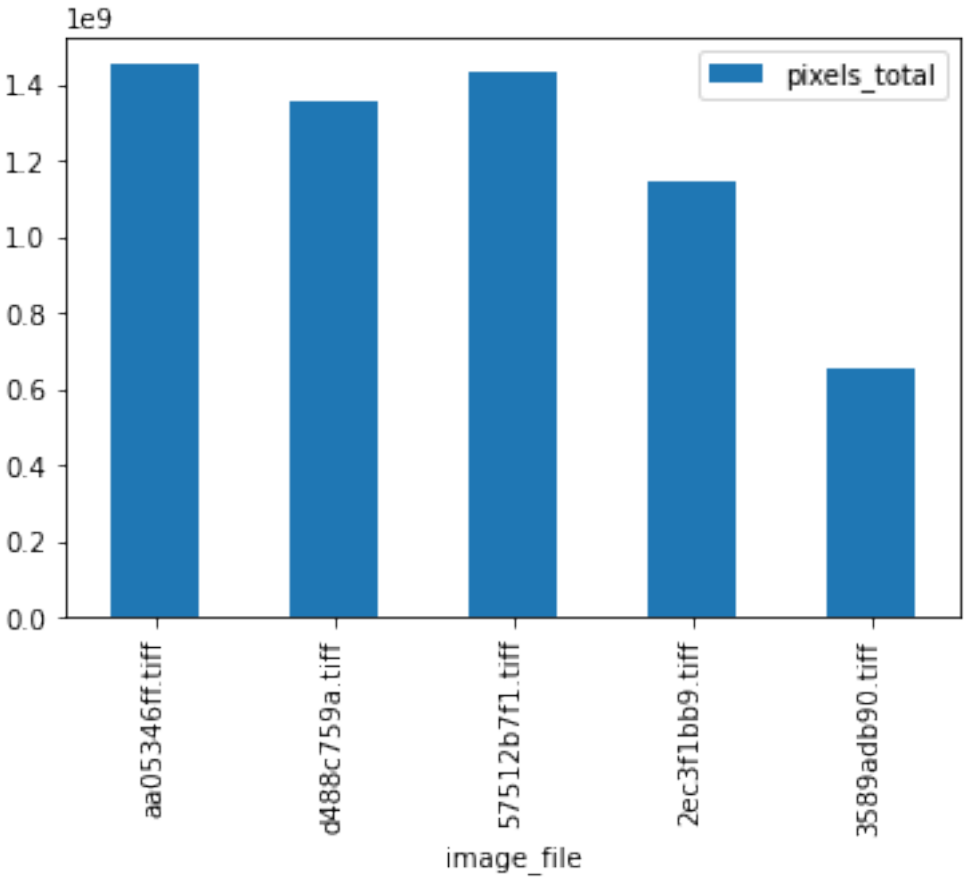
[ ]:
```
test_dataset_information.sort_values('pixels_total',␣
↪ascending=False)[['image_file', 'width_pixels','height_pixels']]
```

[ ]:

	image_file	width_pixels	height_pixels
0	aa05346ff.tiff	47340	30720
2	57512b7f1.tiff	43160	33240
1	d488c759a.tiff	29020	46660
3	2ec3f1bb9.tiff	47723	23990
4	3589adb90.tiff	22165	29433

[ ]:
```
test_dataset_information.plot.bar(x='image_file', y='pixels_total')
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7efde9e65290>
```



Anatomical Structure

[ ]:
```
test_anatomical_seg_files =␣
↪test_dataset_information['anatomical_structures_segmention_file'].to_list()
test_anatomical = list_structures(test_anatomical_seg_files, 'test')
```

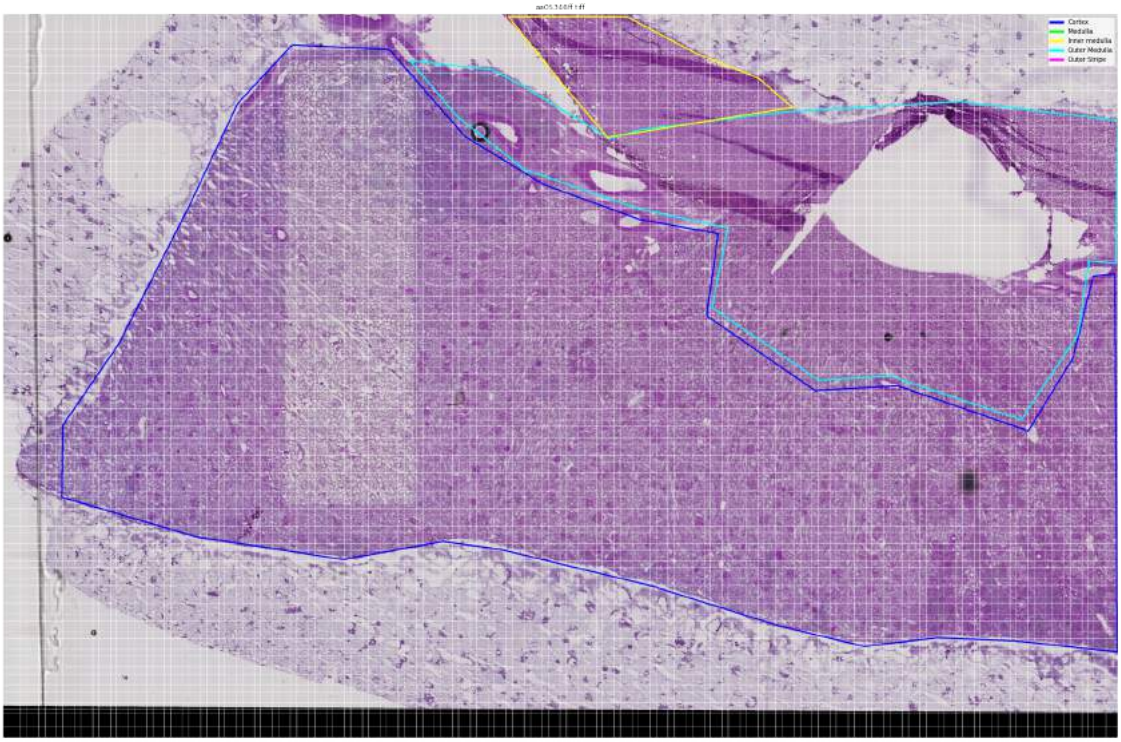
[ ]:
```
pd.DataFrame(test_anatomical.
↪explode('anatomical_structure')['anatomical_structure'].unique(),␣
↪columns=['unique_structures'])
```

[ ]:
```
  unique_structures
0            Cortex
1     Outer Medulla
2     Inner medulla
3           Medulla
```

Mask Visualization

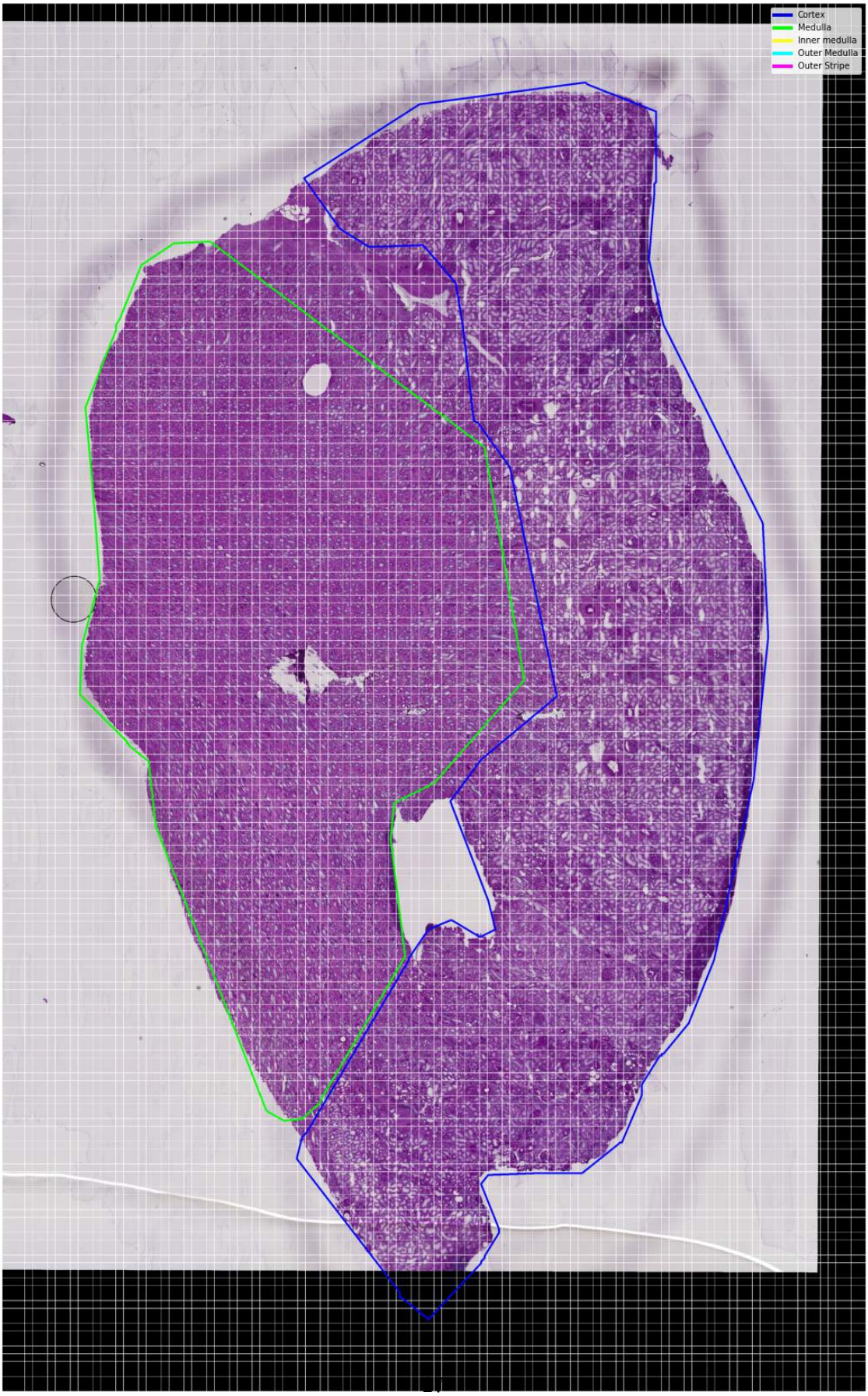
[ ]:
```
plot_masks(test_dataset_information, 'test', 1024, 256, False)
```

aa05346ff.tiff

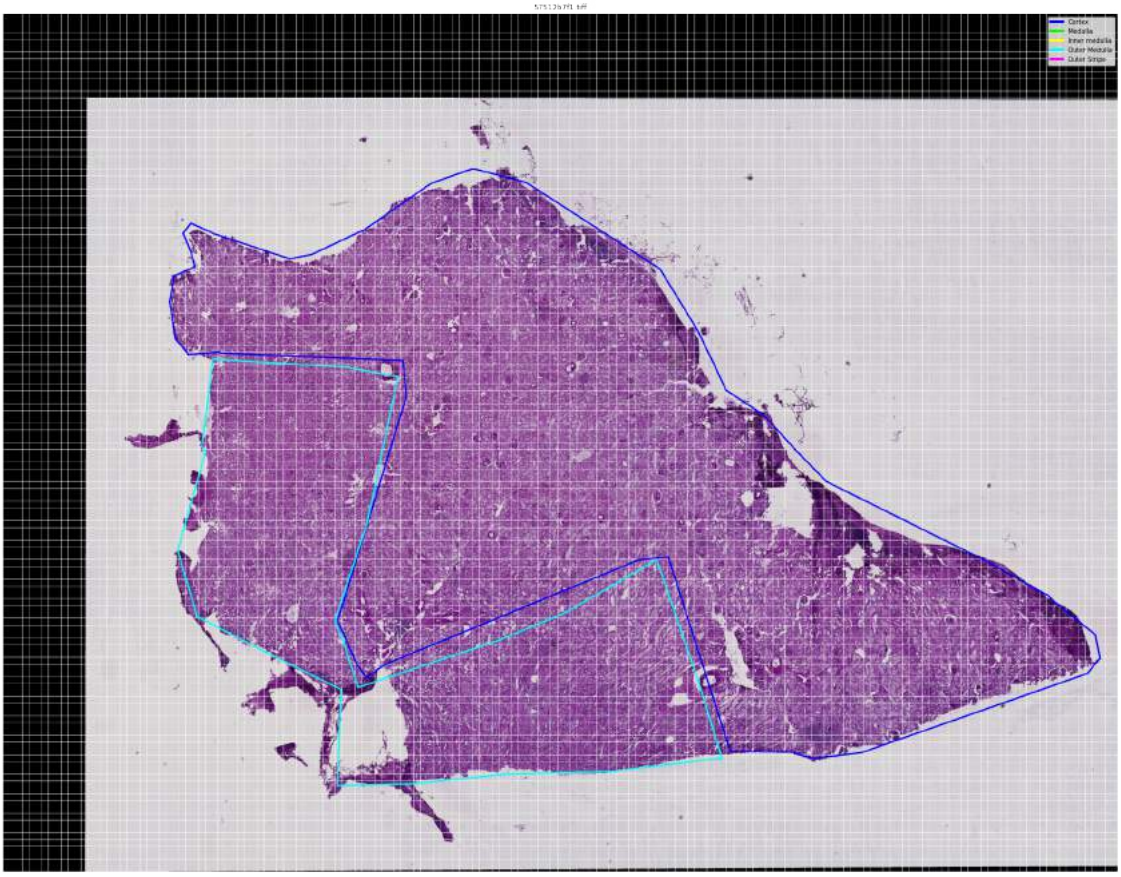


d488c759a.tiff

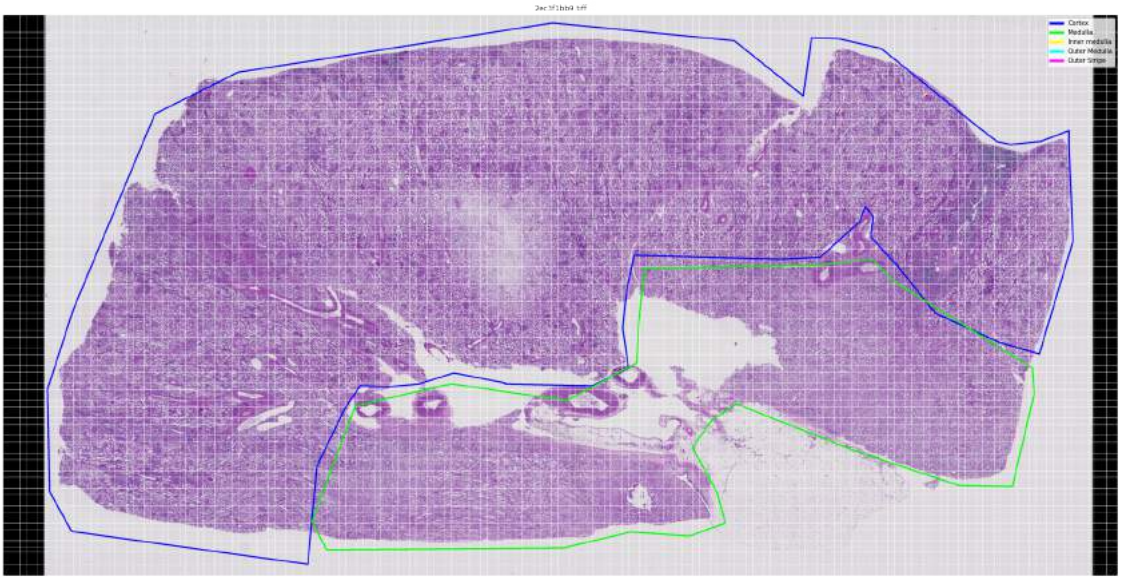
d488c759a.tiff



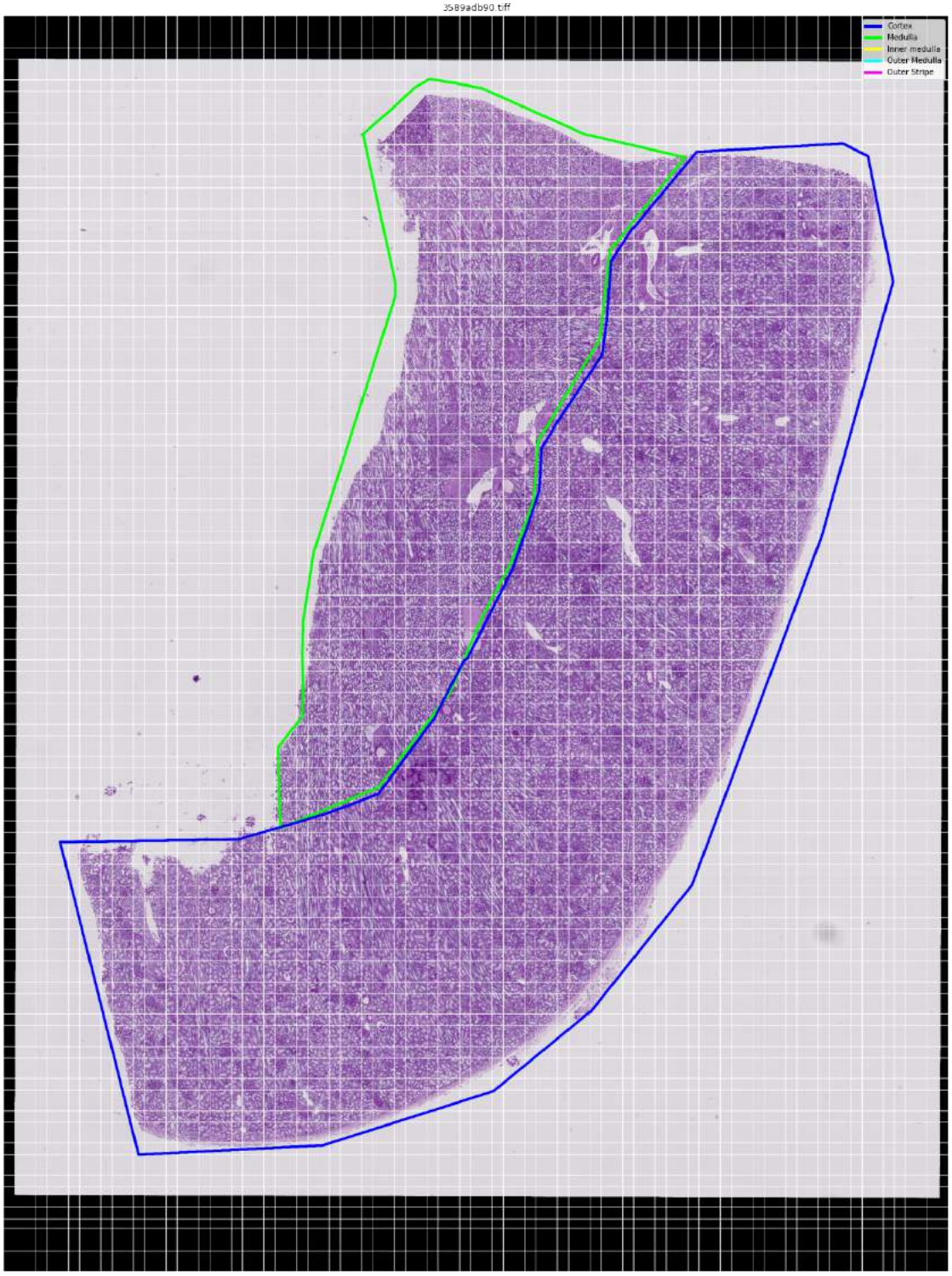
57512b7f1.tiff



2ec3f1bb9.tiff



3589adb90.tiff


3589adb90.tiff
Cortex
Medulla
Inner medulla
Outer Medulla
Outer Stripe