Black Hole Simulation Analysis

April 15, 2023

A long standing goal in astrophysics is to directly observe the immediate environment of a black hole with angular resolution comparable to the event horizon. Such observations could lead to images of strong gravity effects that are expected near a black hole, and to the direct detection of dynamics near the black hole as matter orbits at near light speeds. This capability would open a new window on the study of general relativity in the strong field regime, accretion and outflow processes at the edge of a black hole, the existence of event horizons, and fundamental black hole physics.

The EHT is an international collaboration that has formed to continue the steady long-term progress on improving the capability of Very Long Baseline Interferometry (VLBI) at short wavelengths in pursuit of this goal. This technique of linking radio dishes across the globe to create an Earth-sized interferometer, has been used to measure the size of the emission regions of the two supermassive black holes with the largest apparent event horizons: SgrA* at the center of the Milky Way and M87 in the center of the Virgo A galaxy. In both cases, the sizes match that of the predicted silhouette caused by the extreme lensing of light by the black hole. Addition of key millimeter and submillimeter wavelength facilities at high altitude sites has now opened the possibility of imaging such features and sensing the dynamic evolution of black hole accretion. The EHT project includes theoretical and simulation studies that are framing questions rooted at the black hole boundary that may soon be answered through observations.

By linking together existing telescopes using novel systems, the EHT leverages considerable global investment to create a fundamentally new instrument with angular resolving power that is the highest possible from the surface of the Earth. Over the coming years, the international EHT team will mount observing campaigns of increasing resolving power and sensitivity, aiming to bring black holes into focus.

[95]:
```
!pip install opencv-python
```

```
Requirement already satisfied: opencv-python in
c:\users\krishna\anaconda3\lib\site-packages (4.7.0.72)
Requirement already satisfied: numpy>=1.17.0 in
c:\users\krishna\anaconda3\lib\site-packages (from opencv-python) (1.23.5)
```

A long standing goal in astrophysics is to directly observe the immediate environment of a black hole with angular resolution comparable to the event horizon. Such observations could lead to images of strong gravity effects that are expected near a black hole, and to the direct detection of dynamics near the black hole as matter orbits at near light speeds. This capability would open a new window on the study of general relativity in the strong field regime, accretion and outflow processes at the edge of a black hole, the existence of event horizons, and fundamental black hole physics.

The EHT is an international collaboration that has formed to continue the steady long-term progress on improving the capability of Very Long Baseline Interferometry (VLBI) at short wavelengths in pursuit of this goal. This technique of linking radio dishes across the globe to create an Earth-sized interferometer, has been used to measure the size of the emission regions of the two supermassive black holes with the largest apparent event horizons: SgrA* at the center of the Milky Way and M87 in the center of the Virgo A galaxy. In both cases, the sizes match that of the predicted silhouette caused by the extreme lensing of light by the black hole. Addition of key millimeter and submillimeter wavelength facilities at high altitude sites has now opened the possibility of imaging such features and sensing the dynamic evolution of black hole accretion. The EHT project includes theoretical and simulation studies that are framing questions rooted at the black hole boundary that may soon be answered through observations.

By linking together existing telescopes using novel systems, the EHT leverages considerable global investment to create a fundamentally new instrument with angular resolving power that is the highest possible from the surface of the Earth. Over the coming years, the international EHT team will mount observing campaigns of increasing resolving power and sensitivity, aiming to bring black holes into focus.

[96]:
```
#GENERAL

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
import random
import time
```

[97]:
```
#Path Process

import os
import os.path
from pathlib import Path
import glob
from scipy.io import loadmat
import csv
```

[98]:
```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in
c:\users\krishna\anaconda3\lib\site-packages (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in
c:\users\krishna\anaconda3\lib\site-packages (from scikit-learn) (1.1.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\krishna\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: numpy>=1.17.3 in
c:\users\krishna\anaconda3\lib\site-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in
c:\users\krishna\anaconda3\lib\site-packages (from scikit-learn) (1.10.0)
```

```
Note: you may need to restart the kernel to use updated packages.
```

[99]:
```
pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\krishna\anaconda3\lib\site-
packages (6.4.4)
Requirement already satisfied: nbformat>=4.4 in
c:\users\krishna\anaconda3\lib\site-packages (from nbconvert) (5.7.0)
Requirement already satisfied: traitlets>=5.0 in
c:\users\krishna\anaconda3\lib\site-packages (from nbconvert) (5.7.1)
Requirement already satisfied: jupyterlab-pygments in
c:\users\krishna\anaconda3\lib\site-packages (from nbconvert) (0.1.2)
Requirement already satisfied: mistune<2,>=0.8.1 in
c:\users\krishna\anaconda3\lib\site-packages (from nbconvert) (0.8.4)
Requirement already satisfied: pygments>=2.4.1 in
c:\users\krishna\anaconda3\lib\site-packages (from nbconvert) (2.11.2)
Requirement already satisfied: pandocfilters>=1.4.1 in
c:\users\krishna\anaconda3\lib\site-packages (from nbconvert) (1.5.0)
Requirement already satisfied: testpath in c:\users\krishna\anaconda3\lib\site-
packages (from nbconvert) (0.6.0)
Requirement already satisfied: jupyter-core in
c:\users\krishna\anaconda3\lib\site-packages (from nbconvert) (5.3.0)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in
c:\users\krishna\anaconda3\lib\site-packages (from nbconvert) (0.5.13)
Requirement already satisfied: jinja2>=2.4 in
c:\users\krishna\anaconda3\lib\site-packages (from nbconvert) (2.11.3)
Requirement already satisfied: defusedxml in
c:\users\krishna\anaconda3\lib\site-packages (from nbconvert) (0.7.1)
Requirement already satisfied: beautifulsoup4 in
c:\users\krishna\anaconda3\lib\site-packages (from nbconvert) (4.11.1)
Requirement already satisfied: entrypoints>=0.2.2 in
c:\users\krishna\anaconda3\lib\site-packages (from nbconvert) (0.4)
Requirement already satisfied: bleach in c:\users\krishna\anaconda3\lib\site-
packages (from nbconvert) (4.1.0)
Requirement already satisfied: MarkupSafe>=0.23 in
c:\users\krishna\anaconda3\lib\site-packages (from jinja2>=2.4->nbconvert)
(1.1.1)
Requirement already satisfied: jupyter-client>=6.1.5 in
c:\users\krishna\anaconda3\lib\site-packages (from
nbclient<0.6.0,>=0.5.0->nbconvert) (7.4.9)
Requirement already satisfied: nest-asyncio in
c:\users\krishna\anaconda3\lib\site-packages (from
nbclient<0.6.0,>=0.5.0->nbconvert) (1.5.6)
Requirement already satisfied: fastjsonschema in
c:\users\krishna\anaconda3\lib\site-packages (from nbformat>=4.4->nbconvert)
(2.16.2)
Requirement already satisfied: jsonschema>=2.6 in
c:\users\krishna\anaconda3\lib\site-packages (from nbformat>=4.4->nbconvert)
```

```
(4.17.3)
Requirement already satisfied: soupsieve>1.2 in
c:\users\krishna\anaconda3\lib\site-packages (from beautifulsoup4->nbconvert)
(2.3.2.post1)
Requirement already satisfied: webencodings in
c:\users\krishna\anaconda3\lib\site-packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: packaging in c:\users\krishna\anaconda3\lib\site-
packages (from bleach->nbconvert) (23.0)
Requirement already satisfied: six>=1.9.0 in
c:\users\krishna\anaconda3\lib\site-packages (from bleach->nbconvert) (1.16.0)
Requirement already satisfied: platformdirs>=2.5 in
c:\users\krishna\anaconda3\lib\site-packages (from jupyter-core->nbconvert)
(2.5.2)
Requirement already satisfied: pywin32>=300 in
c:\users\krishna\anaconda3\lib\site-packages (from jupyter-core->nbconvert)
(305.1)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in
c:\users\krishna\anaconda3\lib\site-packages (from
jsonschema>=2.6->nbformat>=4.4->nbconvert) (0.18.0)
Requirement already satisfied: attrs>=17.4.0 in
c:\users\krishna\anaconda3\lib\site-packages (from
jsonschema>=2.6->nbformat>=4.4->nbconvert) (22.1.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\krishna\anaconda3\lib\site-packages (from jupyter-
client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (2.8.2)
Requirement already satisfied: pyzmq>=23.0 in
c:\users\krishna\anaconda3\lib\site-packages (from jupyter-
client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (23.2.0)
Requirement already satisfied: tornado>=6.2 in
c:\users\krishna\anaconda3\lib\site-packages (from jupyter-
client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (6.2)
Note: you may need to restart the kernel to use updated packages.
```

[100]:
```
pip install pandoc
```

```
Requirement already satisfied: pandoc in c:\users\krishna\anaconda3\lib\site-
packages (2.3)
Requirement already satisfied: ply in c:\users\krishna\anaconda3\lib\site-
packages (from pandoc) (3.11)
Requirement already satisfied: plumbum in c:\users\krishna\anaconda3\lib\site-
packages (from pandoc) (1.8.1)
Requirement already satisfied: pywin32 in c:\users\krishna\anaconda3\lib\site-
packages (from plumbum->pandoc) (305.1)
Note: you may need to restart the kernel to use updated packages.
```

[107]:
```
#Image Process

from PIL import Image
```

```
import cv2
from scipy.ndimage import gaussian_filter
from keras.preprocessing import image
from skimage.feature import hessian_matrix, hessian_matrix_eigvals
from scipy.ndimage.filters import convolve
from skimage import data, io, filters
import skimage
from skimage.morphology import convex_hull_image, erosion
from IPython import display
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
import matplotlib.patches as patches
from tensorflow.keras.preprocesing.image import ImageDataGenerator
from keras.applications.vgg16 import preprocess_input, decode_pridctions
from keras.preprocessing import image
```

[108]:
```
#SCALER & TRANSFORMATION
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split
from keras import regularizers
from sklearn.preprocessing import LabelEncoder
```

[109]:
```
#ACCURACY CONTROL
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report, roc_auc_score, roc_curve
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import mean_squared_error, r2_score
```

[110]:
```
#OPTIMIZER
from keras.optimizers import RMSprop,Adam,Optimizer,Optimizer, SGD
#MODEL LAYERS
from tensorflow.keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D,␣
 ↪BatchNormalization,MaxPooling2D,BatchNormalization,\
                        Permute, TimeDistributed, Bidirectional,GRU, SimpleRNN,\
LSTM, GlobalAveragePooling2D, SeparableConv2D, ZeroPadding2D, Convolution2D,␣
 ↪ZeroPadding2D,Reshape, Conv2DTranspose,\
LeakyReLU, GaussianNoise, GlobalMaxPooling2D, ReLU, Input, Concatenate
from keras import models
from keras import layers
import tensorflow as tf
from keras.applications import VGG16,VGG19,inception_v3
from keras import backend as K
from keras.utils import plot_model
from keras.datasets import mnist
import keras
```

```
from keras.models import Model
```

[111]:
```
#IGNORING WARNINGS
from warnings import filterwarnings
filterwarnings("ignore",category=DeprecationWarning)
filterwarnings("ignore", category=FutureWarning)
filterwarnings("ignore", category=UserWarning)
```

[112]:
```
Dir_Path = Path("C:/Users/krishna/Downloads/Black_Hole_Real_Simulations")
```

[113]:
```
MP4_Path = list(Dir_Path.glob(r"*.mp4"))
MP4_Path.append(Dir_Path.glob(r"*.mov"))
```

[114]:
```
print("LEN VIDEO LIST: ",len(MP4_Path))
```

```
LEN VIDEO LIST:  9
```

[118]:
```
Video_Series = pd.Series(MP4_Path, name="VIDEO").astype(str)
```

[119]:
```
print(Video_Series.head(-1))
```

```
0    C:\Users\krishna\Downloads\Black_Hole_Real_Sim…
1    C:\Users\krishna\Downloads\Black_Hole_Real_Sim…
2    C:\Users\krishna\Downloads\Black_Hole_Real_Sim…
3    C:\Users\krishna\Downloads\Black_Hole_Real_Sim…
4    C:\Users\krishna\Downloads\Black_Hole_Real_Sim…
5    C:\Users\krishna\Downloads\Black_Hole_Real_Sim…
6    C:\Users\krishna\Downloads\Black_Hole_Real_Sim…
7    C:\Users\krishna\Downloads\Black_Hole_Real_Sim…
Name: VIDEO, dtype: object
```

[120]:
```
print(Video_Series[0])
print("---"*10)
print(Video_Series[2])
print("---"*10)
print(Video_Series[7])
print("---"*10)
```

```
C:\Users\krishna\Downloads\Black_Hole_Real_Simulations\1 mm Wavelength.mp4
------------------------------
C:\Users\krishna\Downloads\Black_Hole_Real_Simulations\1 mm Wavelength_III.mp4
------------------------------
C:\Users\krishna\Downloads\Black_Hole_Real_Simulations\Varying Wavelength.mp4
------------------------------
```

[121]:
```
# VIDEO TRANSFORMATION

#we will use just Accretion Disk Video, but I want to show hoW to export all␣
 ↪frames
```

[122]:
```
#Total
Black_Hole_Image = []

for file_name in Video_Series:
    Video_File = file_name

    Video_Caption = cv2.VideoCapture(Video_File)

    while Video_Caption.isOpened():
        _,frame = Video_Caption.read()

        if _ != True:
            break

        if Video_Caption.isOpened():

            Transformed_IMG = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            Resized_IMG = cv2.resize(Transformed_IMG, (180,180),␣
 ↪interpolation=cv2.INTER_CUBIC)
            Black_Hole_Image.append(Resized_IMG)
```

[123]:
```
print("LEN LIST: ",len(Black_Hole_Image))
print("LEN ARRAY: ",np.shape(np.array(Black_Hole_Image)))
```

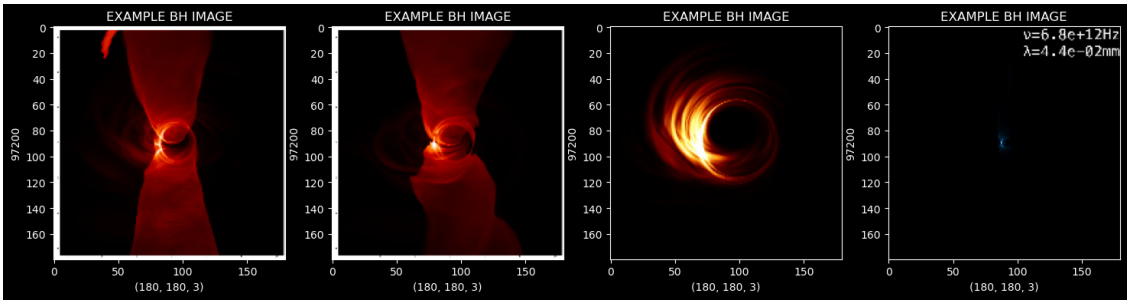
```
LEN LIST:  7235
LEN ARRAY:  (7235, 180, 180, 3)
```

[124]:
```
plt.style.use("dark_background")
```

[125]:
```
figure,axis = plt.subplots(1,4,figsize=(18,18))

axis[0].set_xlabel(Black_Hole_Image[0].shape)
axis[0].set_ylabel(Black_Hole_Image[0].size)
axis[0].set_title("EXAMPLE BH IMAGE")
axis[0].imshow(Black_Hole_Image[0])

axis[1].set_xlabel(Black_Hole_Image[567].shape)
axis[1].set_ylabel(Black_Hole_Image[567].size)
axis[1].set_title("EXAMPLE BH IMAGE")
axis[1].imshow(Black_Hole_Image[567])

axis[2].set_xlabel(Black_Hole_Image[3400].shape)
axis[2].set_ylabel(Black_Hole_Image[3400].size)
axis[2].set_title("EXAMPLE BH IMAGE")
axis[2].imshow(Black_Hole_Image[3400])

axis[3].set_xlabel(Black_Hole_Image[7000].shape)
axis[3].set_ylabel(Black_Hole_Image[7000].size)
```

```
axis[3].set_title("EXAMPLE BH IMAGE")
axis[3].imshow(Black_Hole_Image[7000])
```

[125]: <matplotlib.image.AxesImage at 0x21f1217a3d0>



#SPECIFIC

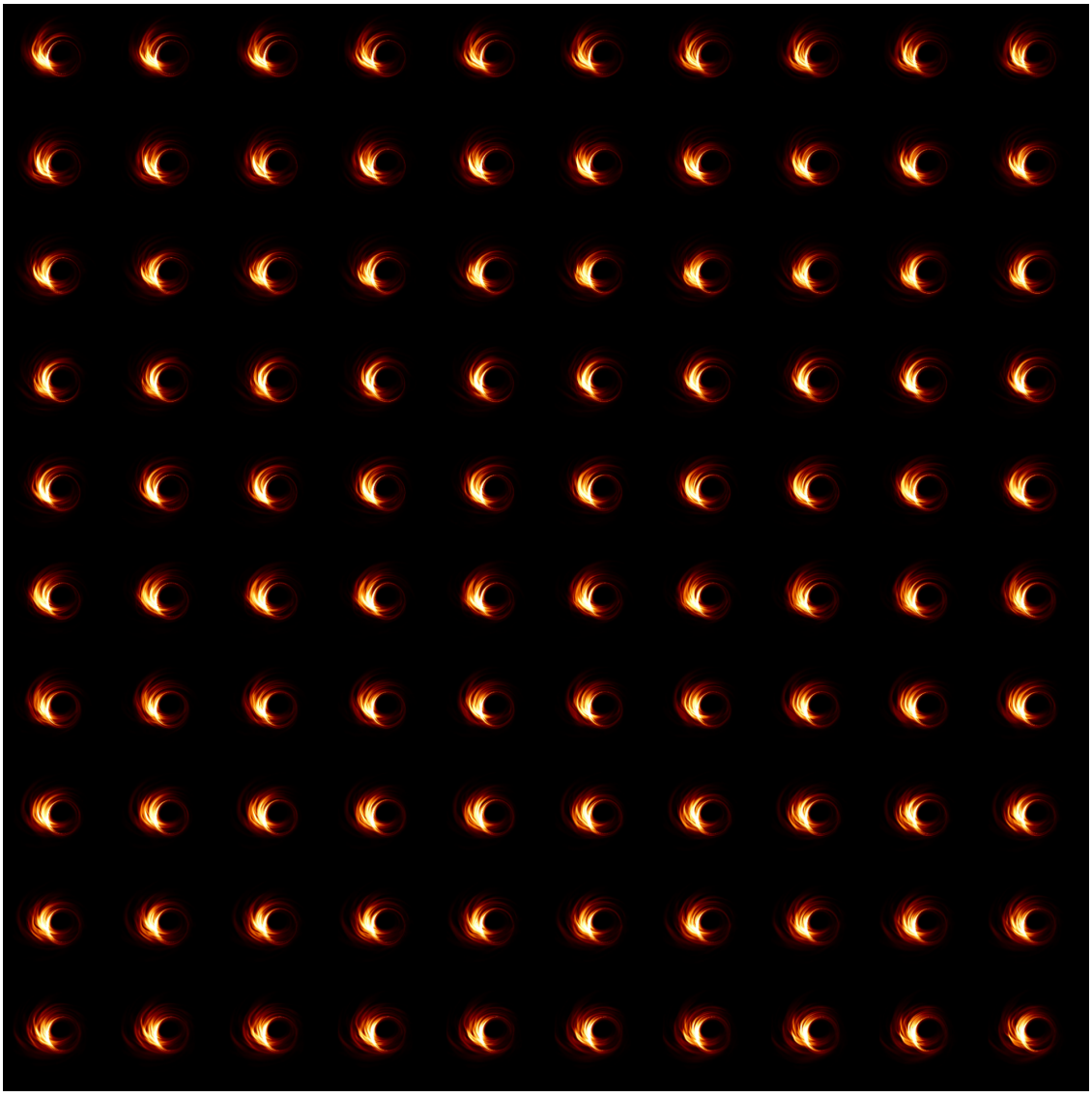
[127]:
```
Accretion_Disk_Path = "C:/Users/krishna/Downloads/Black_Hole_Real_Simulations/
 ↪Accretion Disk.mp4"

Acc_List = []
Acc_Video = cv2.VideoCapture(Accretion_Disk_Path)

while Acc_Video.isOpened():

    _,frame = Acc_Video.read()

    if _ != True:
        break

    if Acc_Video.isOpened():
        Transformed_IMG = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        Resized_IMG = cv2.resize(Transformed_IMG,(180, 180), interpolation=cv2.
 ↪INTER_CUBIC)
        Acc_List.append(Resized_IMG)
```

[128]:
```
print("LEN LIST:", len(Acc_List))
print("LEN ARRAY: ", np.shape(np.array(Acc_List)))
```

```
LEN LIST: 513
LEN ARRAY:  (513, 180, 180, 3)
```

[129]:
```
figure, axis = plt.subplots(10,10,figsize=(20,20))
for indexing, operations in enumerate(axis.flat):

    Picking_IMG = Acc_List[indexing*5]
```

```
    operations.imshow(Picking_IMG)
    operations.axis("off")

plt.tight_layout()
plt.show()
```



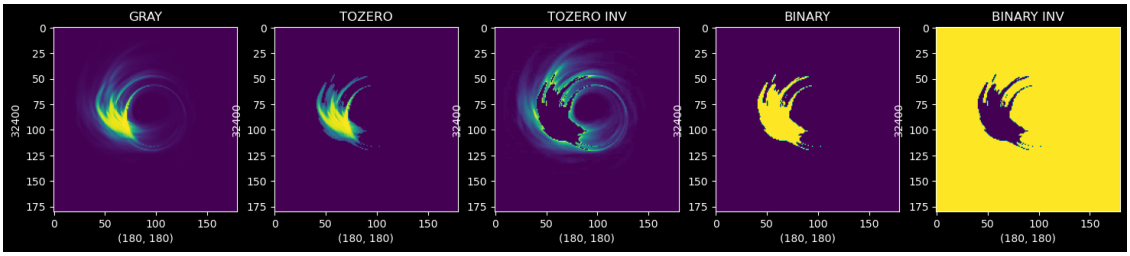
ANALYSIS

THRESHOLD TYPE

[130]:
```
figure,axis = plt.subplots(1,5,figsize=(18,18))
Picking_IMG = Acc_List[1]
```

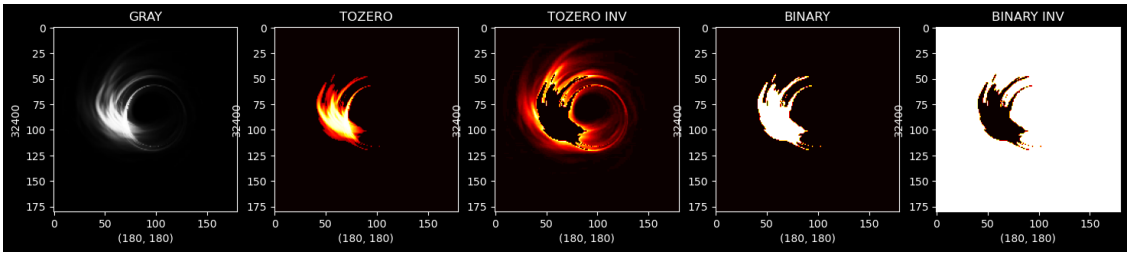
```
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,55,255,cv2.THRESH_TOZERO)
_,Threshold_IMG_TOZERO_INV = cv2.threshold(Gray_IMG,55,255,cv2.
 ↪THRESH_TOZERO_INV)
_,Threshold_IMG_BINARY = cv2.threshold(Gray_IMG,55,255,cv2.THRESH_BINARY)
_,Threshold_IMG_BINARY_INV = cv2.threshold(Gray_IMG,55,255,cv2.
 ↪THRESH_BINARY_INV)

axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG)

axis[1].set_xlabel(Threshold_IMG_TOZERO.shape)
axis[1].set_ylabel(Threshold_IMG_TOZERO.size)
axis[1].set_title("TOZERO")
axis[1].imshow(Threshold_IMG_TOZERO)

axis[2].set_xlabel(Threshold_IMG_TOZERO_INV.shape)
axis[2].set_ylabel(Threshold_IMG_TOZERO_INV.size)
axis[2].set_title("TOZERO INV")
axis[2].imshow(Threshold_IMG_TOZERO_INV)

axis[3].set_xlabel(Threshold_IMG_BINARY.shape)
axis[3].set_ylabel(Threshold_IMG_BINARY.size)
axis[3].set_title("BINARY")
axis[3].imshow(Threshold_IMG_BINARY)

axis[4].set_xlabel(Threshold_IMG_BINARY_INV.shape)
axis[4].set_ylabel(Threshold_IMG_BINARY_INV.size)
axis[4].set_title("BINARY INV")
axis[4].imshow(Threshold_IMG_BINARY_INV)
```

[130]: <matplotlib.image.AxesImage at 0x21f1cdd9460>



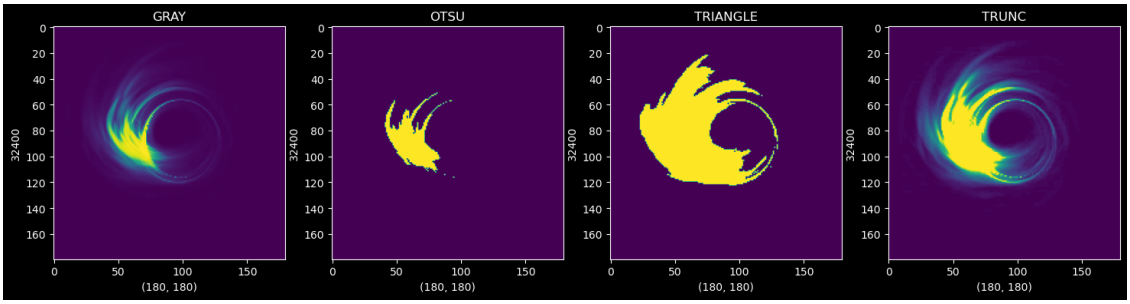
[131]:
```
figure,axis = plt.subplots(1,5,figsize=(18,18))
Picking_IMG = Acc_List[1]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,55,255,cv2.THRESH_TOZERO)
_,Threshold_IMG_TOZERO_INV = cv2.threshold(Gray_IMG,55,255,cv2.
 ↪THRESH_TOZERO_INV)
_,Threshold_IMG_BINARY = cv2.threshold(Gray_IMG,55,255,cv2.THRESH_BINARY)
_,Threshold_IMG_BINARY_INV = cv2.threshold(Gray_IMG,55,255,cv2.
 ↪THRESH_BINARY_INV)

axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Threshold_IMG_TOZERO.shape)
axis[1].set_ylabel(Threshold_IMG_TOZERO.size)
axis[1].set_title("TOZERO")
axis[1].imshow(Threshold_IMG_TOZERO,cmap="hot")

axis[2].set_xlabel(Threshold_IMG_TOZERO_INV.shape)
axis[2].set_ylabel(Threshold_IMG_TOZERO_INV.size)
axis[2].set_title("TOZERO INV")
axis[2].imshow(Threshold_IMG_TOZERO_INV,cmap="hot")

axis[3].set_xlabel(Threshold_IMG_BINARY.shape)
axis[3].set_ylabel(Threshold_IMG_BINARY.size)
axis[3].set_title("BINARY")
axis[3].imshow(Threshold_IMG_BINARY,cmap="hot")

axis[4].set_xlabel(Threshold_IMG_BINARY_INV.shape)
axis[4].set_ylabel(Threshold_IMG_BINARY_INV.size)
axis[4].set_title("BINARY INV")
axis[4].imshow(Threshold_IMG_BINARY_INV,cmap="hot")
```

[131]:
```
<matplotlib.image.AxesImage at 0x21f1d200970>
```



[132]:
```
figure,axis = plt.subplots(1,4,figsize=(18,18))
Picking_IMG = Acc_List[1]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_OTSU = cv2.threshold(Gray_IMG,55,255,cv2.THRESH_OTSU)
_,Threshold_IMG_TRIANGLE = cv2.threshold(Gray_IMG,55,255,cv2.THRESH_TRIANGLE)
_,Threshold_IMG_TRUNC = cv2.threshold(Gray_IMG,55,255,cv2.THRESH_TRUNC)

axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG)

axis[1].set_xlabel(Threshold_IMG_OTSU.shape)
axis[1].set_ylabel(Threshold_IMG_OTSU.size)
axis[1].set_title("OTSU")
axis[1].imshow(Threshold_IMG_OTSU)

axis[2].set_xlabel(Threshold_IMG_TRIANGLE.shape)
axis[2].set_ylabel(Threshold_IMG_TRIANGLE.size)
axis[2].set_title("TRIANGLE")
axis[2].imshow(Threshold_IMG_TRIANGLE)

axis[3].set_xlabel(Threshold_IMG_TRUNC.shape)
axis[3].set_ylabel(Threshold_IMG_TRUNC.size)
axis[3].set_title("TRUNC")
axis[3].imshow(Threshold_IMG_TRUNC)
```

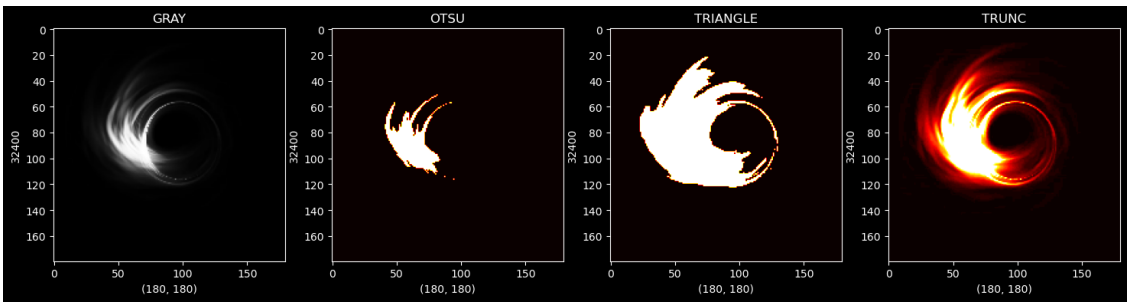
[132]:
```
<matplotlib.image.AxesImage at 0x21f1e5c6c70>
```



[133]:
```
figure,axis = plt.subplots(1,4,figsize=(18,18))
Picking_IMG = Acc_List[1]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_OTSU = cv2.threshold(Gray_IMG,55,255,cv2.THRESH_OTSU)
_,Threshold_IMG_TRIANGLE = cv2.threshold(Gray_IMG,55,255,cv2.THRESH_TRIANGLE)
_,Threshold_IMG_TRUNC = cv2.threshold(Gray_IMG,55,255,cv2.THRESH_TRUNC)
```

```
axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Threshold_IMG_OTSU.shape)
axis[1].set_ylabel(Threshold_IMG_OTSU.size)
axis[1].set_title("OTSU")
axis[1].imshow(Threshold_IMG_OTSU,cmap="hot")

axis[2].set_xlabel(Threshold_IMG_TRIANGLE.shape)
axis[2].set_ylabel(Threshold_IMG_TRIANGLE.size)
axis[2].set_title("TRIANGLE")
axis[2].imshow(Threshold_IMG_TRIANGLE,cmap="hot")

axis[3].set_xlabel(Threshold_IMG_TRUNC.shape)
axis[3].set_ylabel(Threshold_IMG_TRUNC.size)
axis[3].set_title("TRUNC")
axis[3].imshow(Threshold_IMG_TRUNC,cmap="hot")
```

[133]: <matplotlib.image.AxesImage at 0x21f1e9a2310>

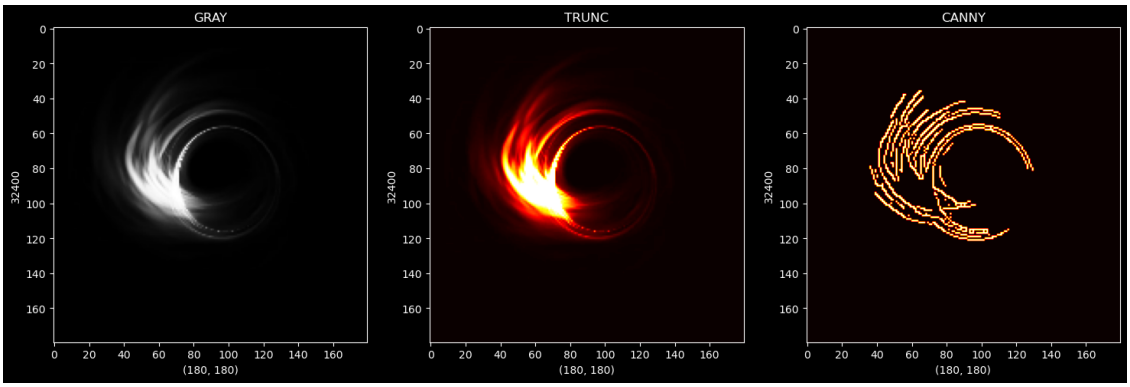


CANNY TYPE

[134]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))
Picking_IMG = Acc_List[1]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TRUNC = cv2.threshold(Gray_IMG,220,255,cv2.THRESH_TRUNC)
Canny_IMG = cv2.Canny(Threshold_IMG_TRUNC,55,100)

axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Threshold_IMG_TRUNC.shape)
```

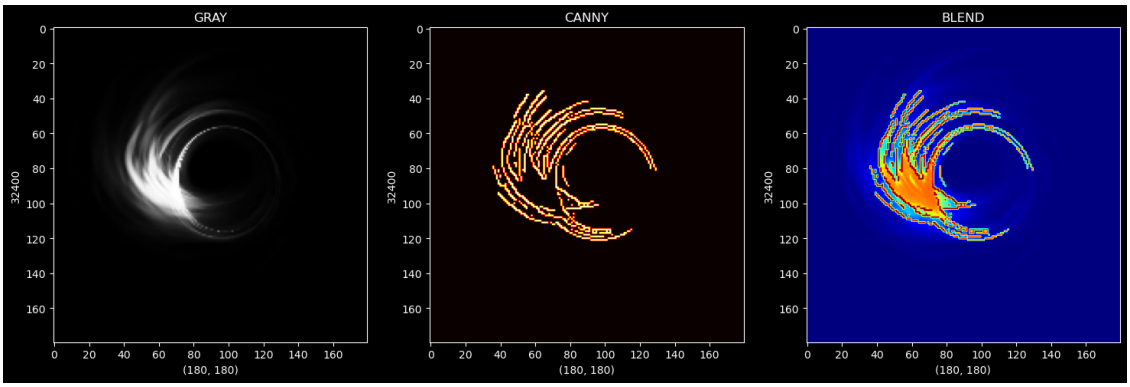
```
axis[1].set_ylabel(Threshold_IMG_TRUNC.size)
axis[1].set_title("TRUNC")
axis[1].imshow(Threshold_IMG_TRUNC,cmap="hot")

axis[2].set_xlabel(Canny_IMG.shape)
axis[2].set_ylabel(Canny_IMG.size)
axis[2].set_title("CANNY")
axis[2].imshow(Canny_IMG,cmap="hot")
```

[134]: <matplotlib.image.AxesImage at 0x21f1eb3ba60>



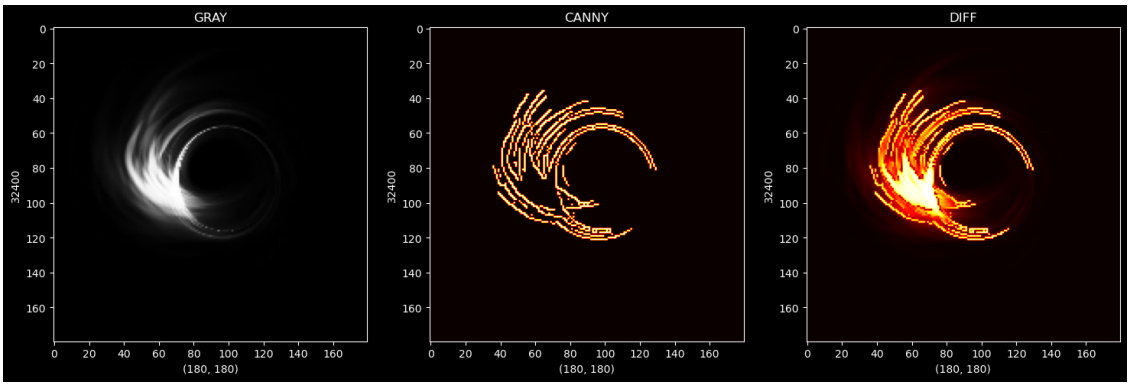
[135]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))
Picking_IMG = Acc_List[1]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TRUNC = cv2.threshold(Gray_IMG,220,255,cv2.THRESH_TRUNC)
Canny_IMG = cv2.Canny(Threshold_IMG_TRUNC,55,100)
Blend_IMG = cv2.addWeighted(Gray_IMG,0.8,Canny_IMG,0.8,0.1)

axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Canny_IMG.shape)
axis[1].set_ylabel(Canny_IMG.size)
axis[1].set_title("CANNY")
axis[1].imshow(Canny_IMG,cmap="hot")

axis[2].set_xlabel(Blend_IMG.shape)
axis[2].set_ylabel(Blend_IMG.size)
axis[2].set_title("BLEND")
axis[2].imshow(Blend_IMG,cmap="jet")
```

[135]: <matplotlib.image.AxesImage at 0x21f201d38e0>



[136]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))
Picking_IMG = Acc_List[1]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TRUNC = cv2.threshold(Gray_IMG,220,255,cv2.THRESH_TRUNC)
Canny_IMG = cv2.Canny(Threshold_IMG_TRUNC,55,100)
Diff_Image = cv2.absdiff(Gray_IMG,Canny_IMG,np.zeros((2,2)))

axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Canny_IMG.shape)
axis[1].set_ylabel(Canny_IMG.size)
axis[1].set_title("CANNY")
axis[1].imshow(Canny_IMG,cmap="hot")

axis[2].set_xlabel(Diff_Image.shape)
axis[2].set_ylabel(Diff_Image.size)
axis[2].set_title("DIFF")
axis[2].imshow(Diff_Image,cmap="hot")
```

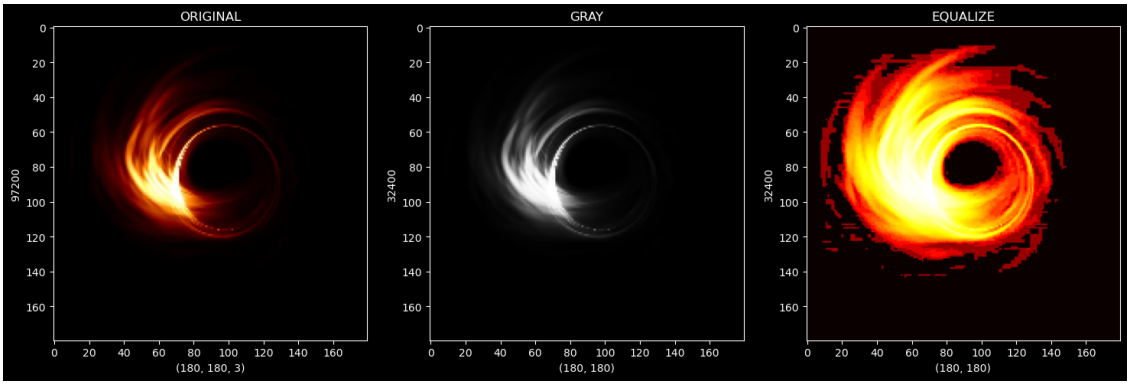
[136]: <matplotlib.image.AxesImage at 0x21f2163a940>



EQUALISE HISTOGRAM

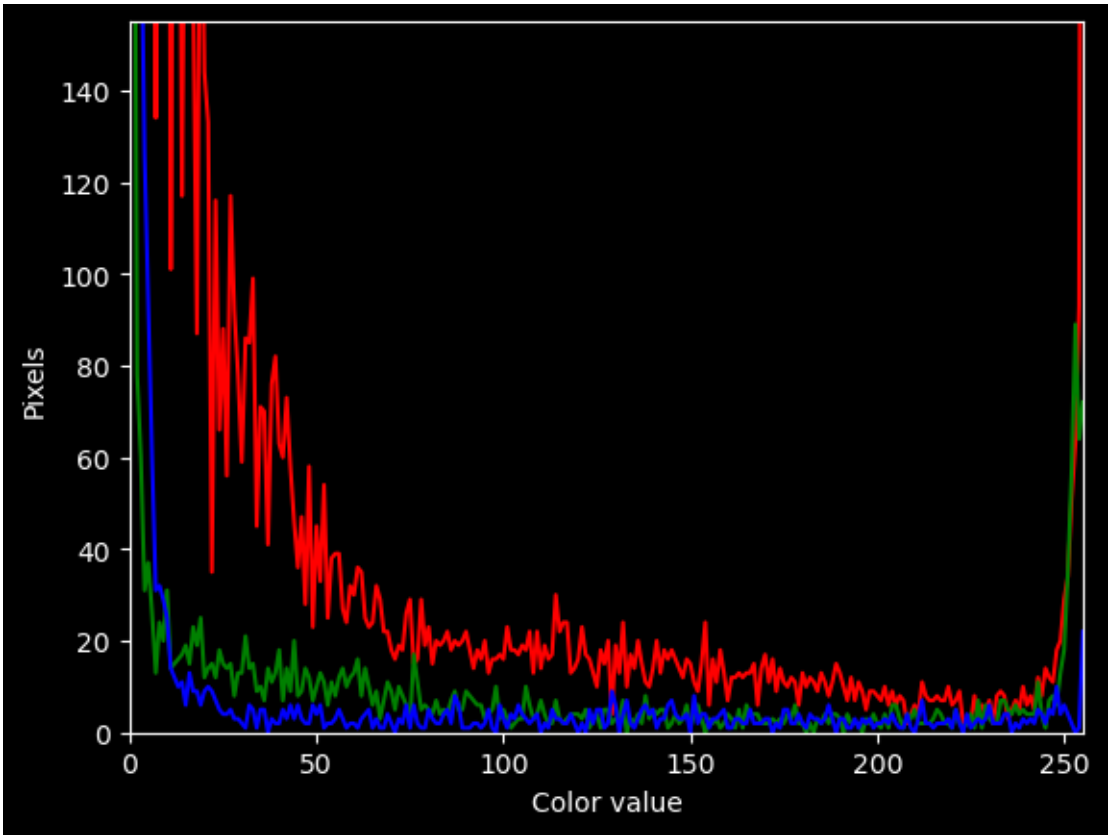
[137]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))
Picking_IMG = Acc_List[1]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
Equalize_IMG = cv2.equalizeHist(Gray_IMG)

axis[0].set_xlabel(Picking_IMG.shape)
axis[0].set_ylabel(Picking_IMG.size)
axis[0].set_title("ORIGINAL")
axis[0].imshow(Picking_IMG,cmap="gray")

axis[1].set_xlabel(Gray_IMG.shape)
axis[1].set_ylabel(Gray_IMG.size)
axis[1].set_title("GRAY")
axis[1].imshow(Gray_IMG,cmap="gray")

axis[2].set_xlabel(Equalize_IMG.shape)
axis[2].set_ylabel(Equalize_IMG.size)
axis[2].set_title("EQUALIZE")
axis[2].imshow(Equalize_IMG,cmap="hot")
```

[137]:
```
<matplotlib.image.AxesImage at 0x21f21aa0370>
```



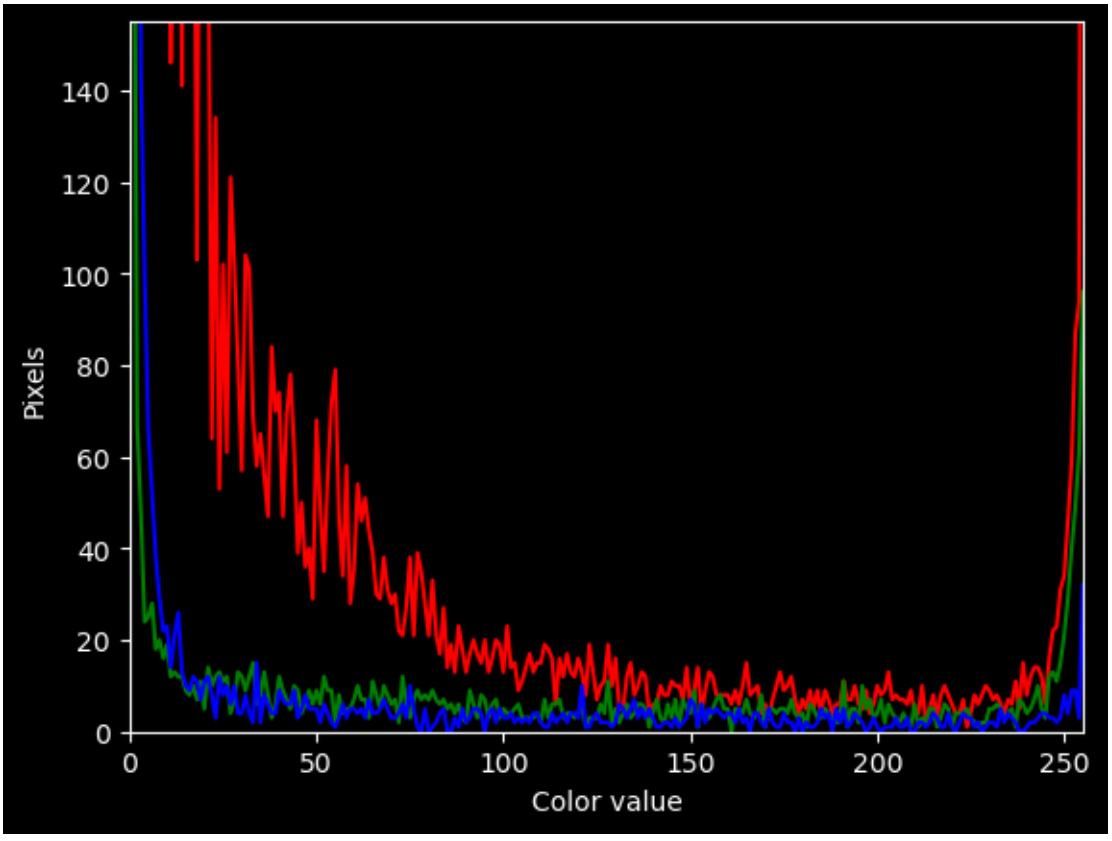
[138]:
```
Single_Image = Acc_List[1]

colors = ("red", "green", "blue")
channel_dim = (0, 1, 2)

plt.xlim([0, 255])
plt.ylim([0, 155])

for channel_id, c in zip(channel_dim, colors):
    histogram, bin_edges = np.histogram(
    Single_Image[:, :, channel_id], bins=256, range=(0, 256))
    plt.plot(bin_edges[0:-1], histogram, color=c)

plt.xlabel("Color value")
plt.ylabel("Pixels")
```

[138]:
```
Text(0, 0.5, 'Pixels')
```



[139]:
```
Single_Image = Acc_List[100]

colors = ("red", "green", "blue")
channel_dim = (0, 1, 2)

plt.xlim([0, 255])
plt.ylim([0, 155])

for channel_id, c in zip(channel_dim, colors):
    histogram, bin_edges = np.histogram(
    Single_Image[:, :, channel_id], bins=256, range=(0, 256))
    plt.plot(bin_edges[0:-1], histogram, color=c)

plt.xlabel("Color value")
plt.ylabel("Pixels")
```

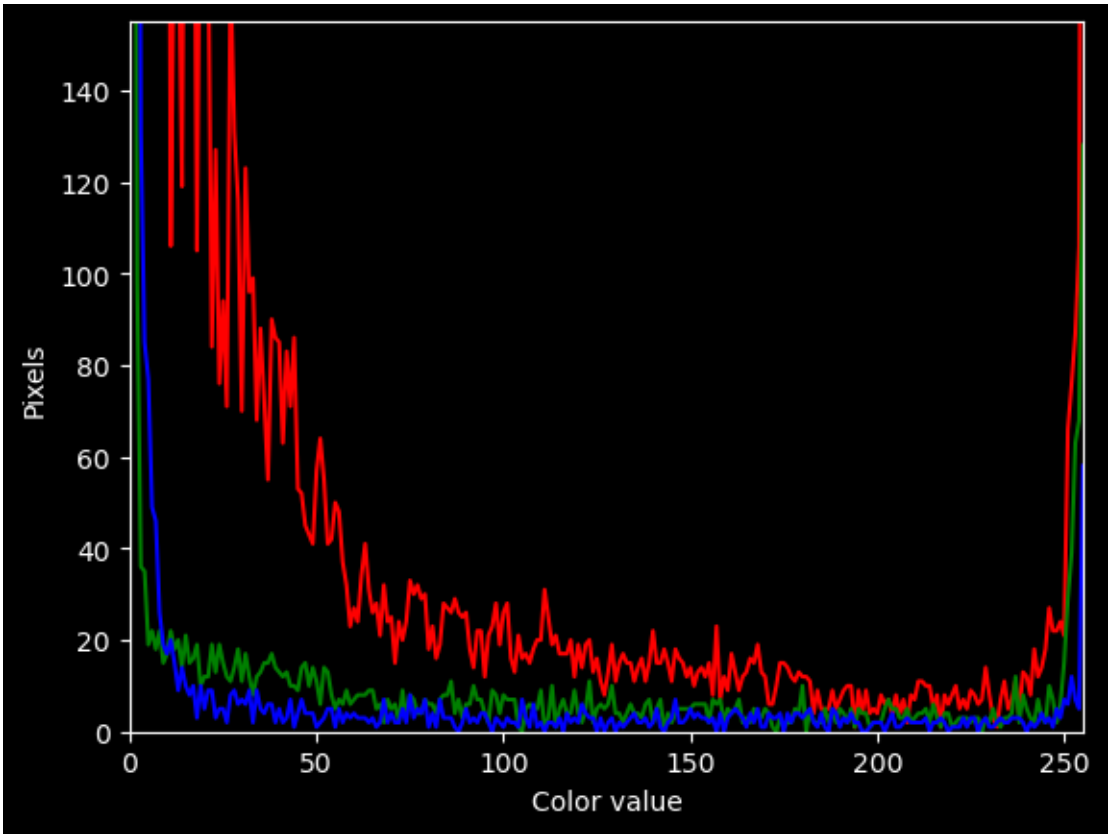
[139]:
```
Text(0, 0.5, 'Pixels')
```



[140]:
```
Single_Image = Acc_List[433]

colors = ("red", "green", "blue")
```

```
channel_dim = (0, 1, 2)

plt.xlim([0, 255])
plt.ylim([0, 155])

for channel_id, c in zip(channel_dim, colors):
    histogram, bin_edges = np.histogram(
    Single_Image[:, :, channel_id], bins=256, range=(0, 256))
    plt.plot(bin_edges[0:-1], histogram, color=c)

plt.xlabel("Color value")
plt.ylabel("Pixels")
```

[140]: Text(0, 0.5, 'Pixels')

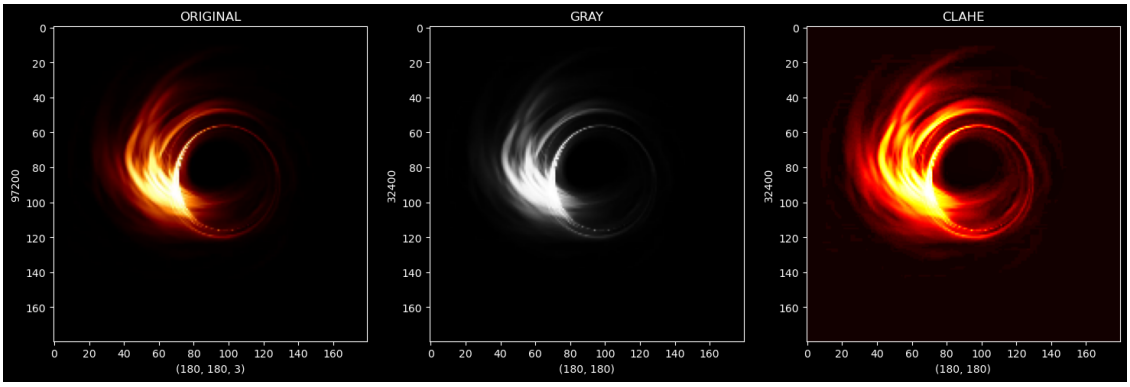


CLAHE TYPE

[141]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))
Picking_IMG = Acc_List[1]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
```

```
Clahe_Func = cv2.createCLAHE(clipLimit=5.0,tileGridSize=(8,8))
Apply_Clahe = Clahe_Func.apply(Gray_IMG)

axis[0].set_xlabel(Picking_IMG.shape)
axis[0].set_ylabel(Picking_IMG.size)
axis[0].set_title("ORIGINAL")
axis[0].imshow(Picking_IMG,cmap="gray")

axis[1].set_xlabel(Gray_IMG.shape)
axis[1].set_ylabel(Gray_IMG.size)
axis[1].set_title("GRAY")
axis[1].imshow(Gray_IMG,cmap="gray")

axis[2].set_xlabel(Apply_Clahe.shape)
axis[2].set_ylabel(Apply_Clahe.size)
axis[2].set_title("CLAHE")
axis[2].imshow(Apply_Clahe,cmap="hot")
```

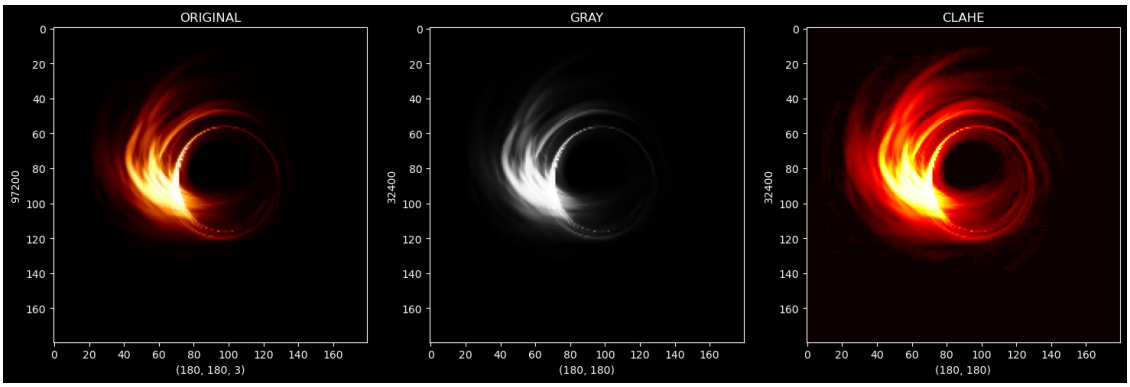
[141]: <matplotlib.image.AxesImage at 0x21f240155e0>



[142]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))
Picking_IMG = Acc_List[1]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)

Clahe_Func = cv2.createCLAHE(clipLimit=5.0,tileGridSize=(2,2))
Apply_Clahe = Clahe_Func.apply(Gray_IMG)

axis[0].set_xlabel(Picking_IMG.shape)
axis[0].set_ylabel(Picking_IMG.size)
axis[0].set_title("ORIGINAL")
axis[0].imshow(Picking_IMG,cmap="gray")

axis[1].set_xlabel(Gray_IMG.shape)
```

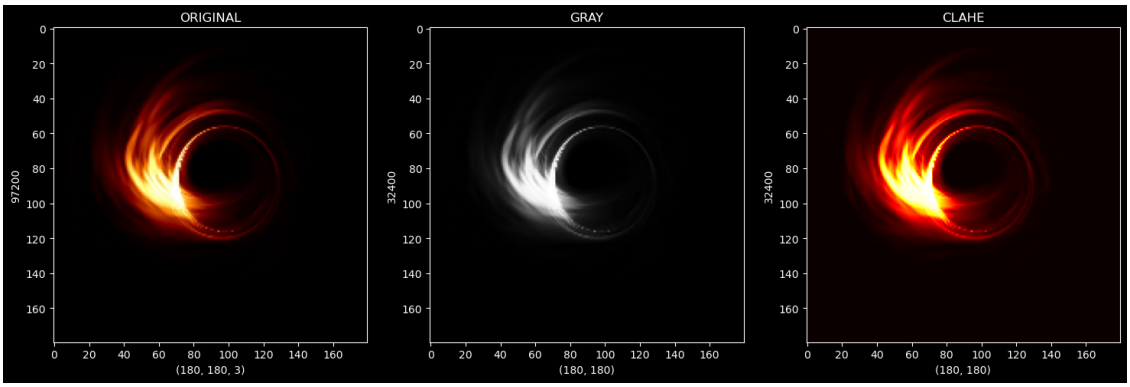
```
axis[1].set_ylabel(Gray_IMG.size)
axis[1].set_title("GRAY")
axis[1].imshow(Gray_IMG,cmap="gray")

axis[2].set_xlabel(Apply_Clahe.shape)
axis[2].set_ylabel(Apply_Clahe.size)
axis[2].set_title("CLAHE")
axis[2].imshow(Apply_Clahe,cmap="hot")
```

[142]: <matplotlib.image.AxesImage at 0x21f2447e460>



[143]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))
Picking_IMG = Acc_List[1]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)

Clahe_Func = cv2.createCLAHE(clipLimit=1.0,tileGridSize=(2,2))
Apply_Clahe = Clahe_Func.apply(Gray_IMG)

axis[0].set_xlabel(Picking_IMG.shape)
axis[0].set_ylabel(Picking_IMG.size)
axis[0].set_title("ORIGINAL")
axis[0].imshow(Picking_IMG,cmap="gray")

axis[1].set_xlabel(Gray_IMG.shape)
axis[1].set_ylabel(Gray_IMG.size)
axis[1].set_title("GRAY")
axis[1].imshow(Gray_IMG,cmap="gray")

axis[2].set_xlabel(Apply_Clahe.shape)
axis[2].set_ylabel(Apply_Clahe.size)
axis[2].set_title("CLAHE")
axis[2].imshow(Apply_Clahe,cmap="hot")
```

[143]: <matplotlib.image.AxesImage at 0x21f248f1190>

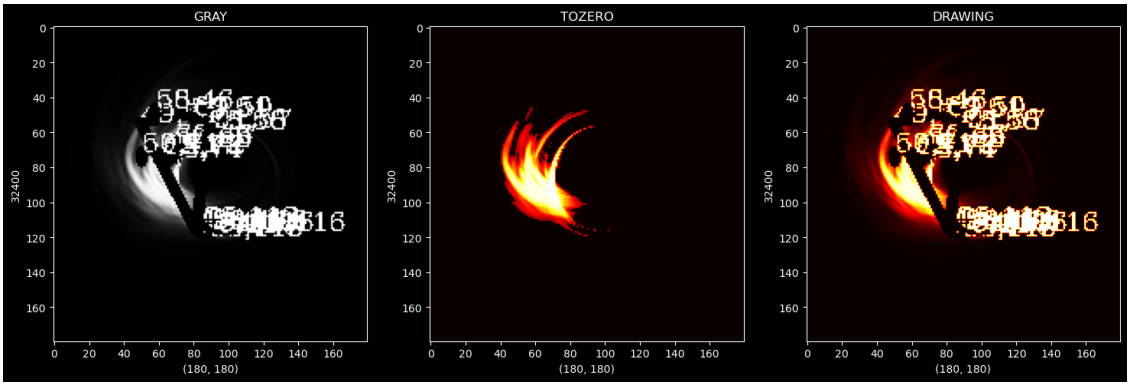


CONTURS AND POLYDP

[144]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))
Picking_IMG = Acc_List[1]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,55,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)

for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.09*cv2.arcLength(cnt,True),True)

    Drawing_Contour = cv2.drawContours(Gray_IMG,[approx],0,(0,255,0),5)

    n_count = approx.ravel()
    i = 0

    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)

            if (i == 0):
                cv2.putText(Gray_IMG,f"{x},{y}",(x,y),cv2.
 ↪FONT_HERSHEY_COMPLEX,0.5,(255,0,255))
            else:
                cv2.putText(Gray_IMG,"+",(x,y),cv2.FONT_HERSHEY_COMPLEX,0.
 ↪5,(0,0,255))
```

```
        i = i + 1


axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Threshold_IMG_TOZERO.shape)
axis[1].set_ylabel(Threshold_IMG_TOZERO.size)
axis[1].set_title("TOZERO")
axis[1].imshow(Threshold_IMG_TOZERO,cmap="hot")

axis[2].set_xlabel(Drawing_Contour.shape)
axis[2].set_ylabel(Drawing_Contour.size)
axis[2].set_title("DRAWING")
axis[2].imshow(Drawing_Contour,cmap="hot")
```

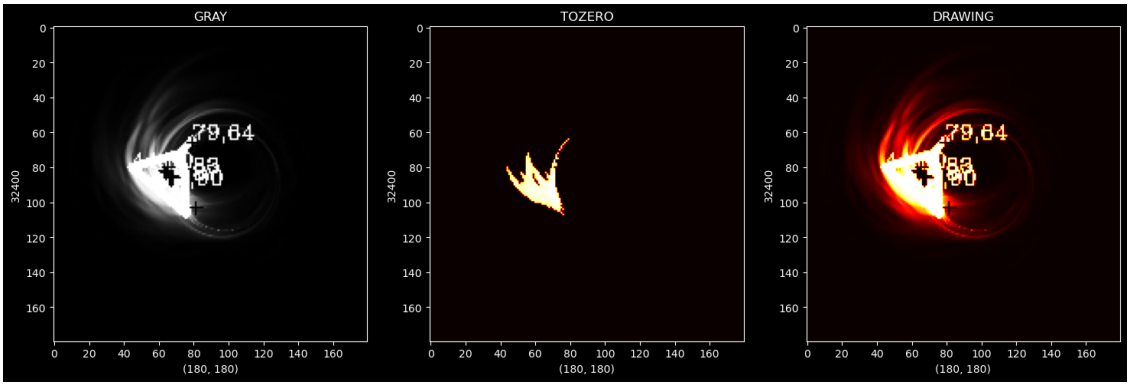
[144]: <matplotlib.image.AxesImage at 0x21f25d438b0>



[145]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))
Picking_IMG = Acc_List[1]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,220,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)

for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.09*cv2.arcLength(cnt,True),True)

    Drawing_Contour = cv2.drawContours(Gray_IMG,[approx],0,(255,255,0),3)
```

```
    n_count = approx.ravel()
    i = 0

    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)

            if (i == 0):
                cv2.putText(Gray_IMG,f"{x},{y}",(x,y),cv2.
↪FONT_HERSHEY_COMPLEX,0.4,(255,0,255))
            else:
                cv2.putText(Gray_IMG,"+",(x,y),cv2.FONT_HERSHEY_COMPLEX,0.
↪4,(0,0,255))

        i = i + 1


axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Threshold_IMG_TOZERO.shape)
axis[1].set_ylabel(Threshold_IMG_TOZERO.size)
axis[1].set_title("TOZERO")
axis[1].imshow(Threshold_IMG_TOZERO,cmap="hot")

axis[2].set_xlabel(Drawing_Contour.shape)
axis[2].set_ylabel(Drawing_Contour.size)
axis[2].set_title("DRAWING")
axis[2].imshow(Drawing_Contour,cmap="hot")
```

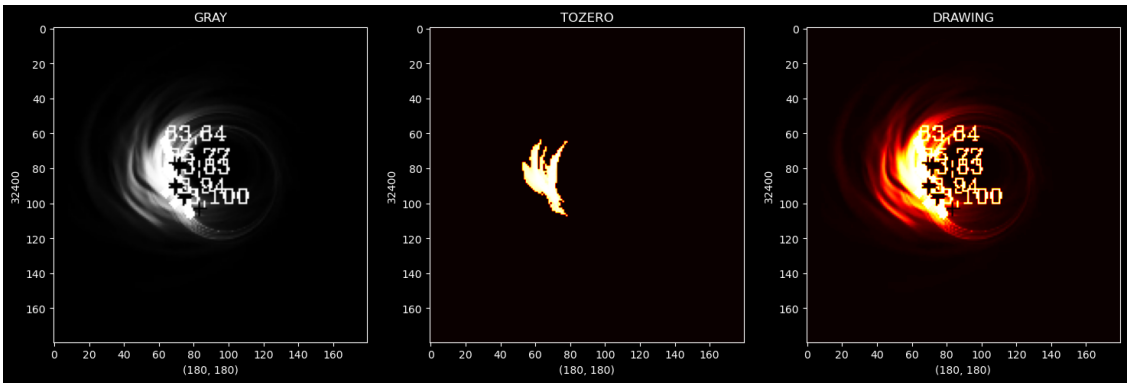
[145]: <matplotlib.image.AxesImage at 0x21f25ee60d0>



[146]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))
Picking_IMG = Acc_List[400]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,220,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)

for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.09*cv2.arcLength(cnt,True),True)

    Drawing_Contour = cv2.drawContours(Gray_IMG,[approx],0,(255,255,0),3)

    n_count = approx.ravel()
    i = 0

    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)

            if (i == 0):
                cv2.putText(Gray_IMG,f"{x},{y}",(x,y),cv2.
 ↪FONT_HERSHEY_COMPLEX,0.4,(255,0,255))
            else:
                cv2.putText(Gray_IMG,"+",(x,y),cv2.FONT_HERSHEY_COMPLEX,0.
 ↪4,(0,0,255))

        i = i + 1


axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Threshold_IMG_TOZERO.shape)
axis[1].set_ylabel(Threshold_IMG_TOZERO.size)
axis[1].set_title("TOZERO")
axis[1].imshow(Threshold_IMG_TOZERO,cmap="hot")

axis[2].set_xlabel(Drawing_Contour.shape)
```

```
axis[2].set_ylabel(Drawing_Contour.size)
axis[2].set_title("DRAWING")
axis[2].imshow(Drawing_Contour,cmap="hot")
```

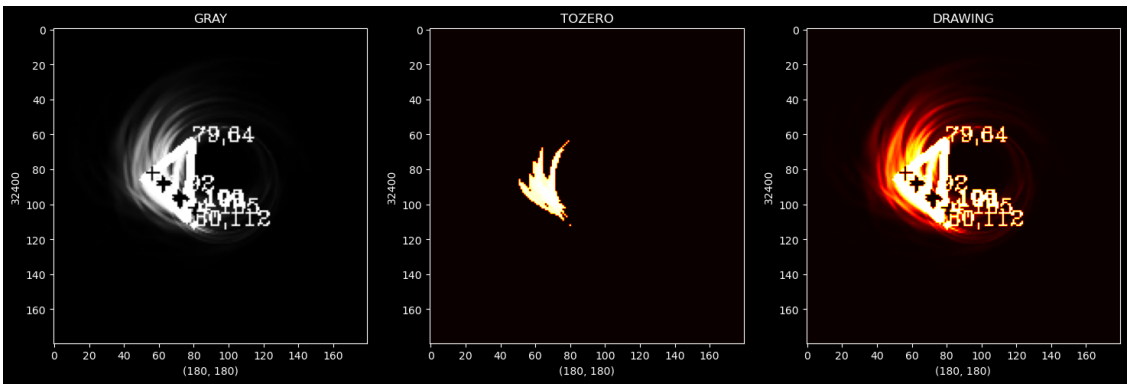
[146]: <matplotlib.image.AxesImage at 0x21f260737c0>



[147]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))
Picking_IMG = Acc_List[277]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,220,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)

for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.09*cv2.arcLength(cnt,True),True)

    Drawing_Contour = cv2.drawContours(Gray_IMG,[approx],0,(255,255,0),3)

    n_count = approx.ravel()
    i = 0

    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)

            if (i == 0):
                cv2.putText(Gray_IMG,f"{x},{y}",(x,y),cv2.
 ↪FONT_HERSHEY_COMPLEX,0.4,(255,0,255))
            else:
```

```
                cv2.putText(Gray_IMG,"+",(x,y),cv2.FONT_HERSHEY_COMPLEX,0.
↪4,(0,0,255))

        i = i + 1


axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Threshold_IMG_TOZERO.shape)
axis[1].set_ylabel(Threshold_IMG_TOZERO.size)
axis[1].set_title("TOZERO")
axis[1].imshow(Threshold_IMG_TOZERO,cmap="hot")

axis[2].set_xlabel(Drawing_Contour.shape)
axis[2].set_ylabel(Drawing_Contour.size)
axis[2].set_title("DRAWING")
axis[2].imshow(Drawing_Contour,cmap="hot")
```

[147]: <matplotlib.image.AxesImage at 0x21f2a6f7e80>



APPROX OVERVIEW AND PROCESS

[148]:
```
Picking_IMG = Acc_List[277]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,220,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
↪CHAIN_APPROX_SIMPLE)

for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.09*cv2.arcLength(cnt,True),True)
```

```
    Drawing_Contour = cv2.drawContours(Gray_IMG,[approx],0,(255,255,0),3)

    n_count = approx.ravel()
    i = 0

    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)
```

[150]:
```
print(n_count)
print("--"*10)
print(n_count.shape)
print("--"*10)
```

```
[56 92 57 91 58 92 57 93]
--------------------
(8,)
--------------------
```

[151]:
```
Reshaped_N = n_count.reshape(4,2)

print(Reshaped_N)
print("--"*10)
print(Reshaped_N.shape)
print("--"*10)
```

```
[[56 92]
 [57 91]
 [58 92]
 [57 93]]
--------------------
(4, 2)
--------------------
```

[152]:
```
print(string_coor)
```

```
56 92
```

[153]:
```
print("X: ",x)
print("Y: ",y)
```

```
X:  56
Y:  92
```

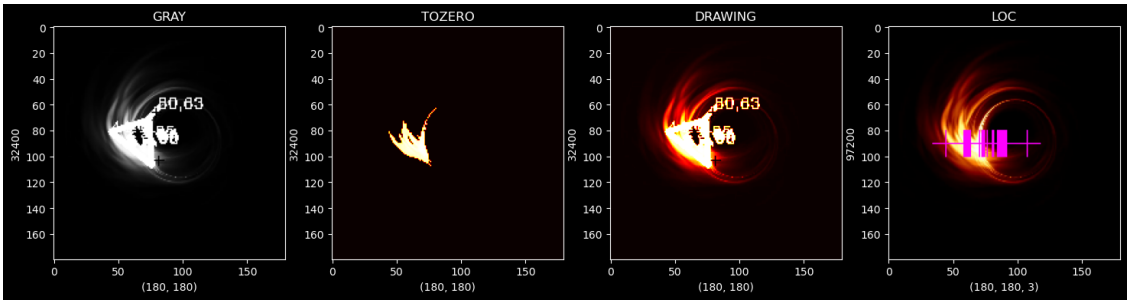
TRANSFORMATION OF COORDINATES / Measuring the Distance Between Singularity Point and Spiral Arms

[154]:
```
figure,axis = plt.subplots(1,4,figsize=(18,18))

Picking_IMG = Acc_List[2]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,220,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)


Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)

Total_Approx = []

for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.09*cv2.arcLength(cnt,True),True)

    Drawing_Contour = cv2.drawContours(Gray_IMG,[approx],0,(255,255,0),3)

    n_count = approx.ravel()
    i = 0

    for x_cor in n_count:
        x_cor = int(x_cor)
        Marker_IMG = cv2.drawMarker(Copy_Main_IMG,(x_cor,int(Copy_Main_IMG.
 ↪shape[1]/2)),(255,0,255),thickness=1)

    print(n_count)
    Total_Approx.append(n_count)

    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)

            if (i == 0):
                cv2.putText(Gray_IMG,f"{x},{y}",(x,y),cv2.
 ↪FONT_HERSHEY_COMPLEX,0.4,(255,0,255))
            else:
                cv2.putText(Gray_IMG,"+",(x,y),cv2.FONT_HERSHEY_COMPLEX,0.
 ↪4,(0,0,255))

        i = i + 1
```

```
axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Threshold_IMG_TOZERO.shape)
axis[1].set_ylabel(Threshold_IMG_TOZERO.size)
axis[1].set_title("TOZERO")
axis[1].imshow(Threshold_IMG_TOZERO,cmap="hot")

axis[2].set_xlabel(Drawing_Contour.shape)
axis[2].set_ylabel(Drawing_Contour.size)
axis[2].set_title("DRAWING")
axis[2].imshow(Drawing_Contour,cmap="hot")

axis[3].set_xlabel(Marker_IMG.shape)
axis[3].set_ylabel(Marker_IMG.size)
axis[3].set_title("LOC")
axis[3].imshow(Marker_IMG)
```

```
[ 44  81  76 107  73  72]
[61 90 62 89 63 90 62 91]
[60 89 61 88 62 89 61 90]
[59 87 60 86 61 88 60 89]
[58 85 59 84 60 86 59 87]
[80 63 74 70]
```

[154]: <matplotlib.image.AxesImage at 0x21f30d67430>



the distribution of the black hole can be observed depending on the distance between the lines

[155]:
```
print("TOTAL ARRAY SHAPE: ",np.shape(np.array(Total_Approx)))
```

```
TOTAL ARRAY SHAPE:  (6,)
```

[156]:
```
Array_Approx = np.array(Total_Approx)
```

[157]:
```
print(Total_Approx)
```

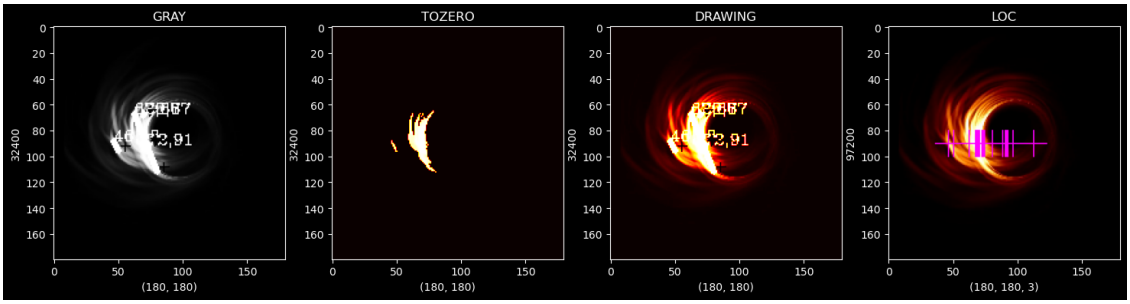
```
[array([ 44,  81,  76, 107,  73,  72], dtype=int32), array([61, 90, 62, 89, 63,
90, 62, 91], dtype=int32), array([60, 89, 61, 88, 62, 89, 61, 90], dtype=int32),
array([59, 87, 60, 86, 61, 88, 60, 89], dtype=int32), array([58, 85, 59, 84, 60,
86, 59, 87], dtype=int32), array([80, 63, 74, 70], dtype=int32)]
```

[158]:
```
figure,axis = plt.subplots(1,4,figsize=(18,18))

Picking_IMG = Acc_List[488]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,220,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)


Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)


for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.09*cv2.arcLength(cnt,True),True)

    Drawing_Contour = cv2.drawContours(Gray_IMG,[approx],0,(255,255,0),3)

    n_count = approx.ravel()
    i = 0

    for x_cor in n_count:
        x_cor = int(x_cor)
        Marker_IMG = cv2.drawMarker(Copy_Main_IMG,(x_cor,int(Copy_Main_IMG.
 ↪shape[1]/2)),(255,0,255),thickness=1)

    print(n_count)

    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)

            if (i == 0):
```

```
                cv2.putText(Gray_IMG,f"{x},{y}",(x,y),cv2.
↪FONT_HERSHEY_COMPLEX,0.4,(255,0,255))
            else:
                cv2.putText(Gray_IMG,"+",(x,y),cv2.FONT_HERSHEY_COMPLEX,0.
↪4,(0,0,255))

        i = i + 1



axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Threshold_IMG_TOZERO.shape)
axis[1].set_ylabel(Threshold_IMG_TOZERO.size)
axis[1].set_title("TOZERO")
axis[1].imshow(Threshold_IMG_TOZERO,cmap="hot")

axis[2].set_xlabel(Drawing_Contour.shape)
axis[2].set_ylabel(Drawing_Contour.size)
axis[2].set_title("DRAWING")
axis[2].imshow(Drawing_Contour,cmap="hot")

axis[3].set_xlabel(Marker_IMG.shape)
axis[3].set_ylabel(Marker_IMG.size)
axis[3].set_title("LOC")
axis[3].imshow(Marker_IMG)
```

```
[46 88 50 96]
[70 67 70 68 69 69 70 68]
[ 62  67  80 112]
[72 91 73 90 74 91 73 92]
```

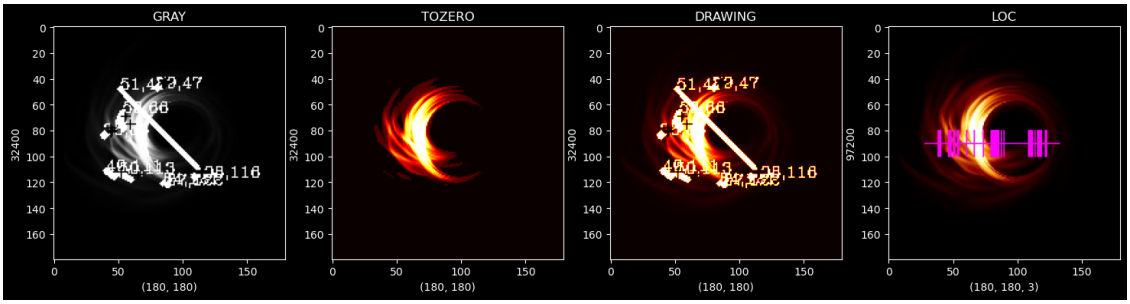
[158]: <matplotlib.image.AxesImage at 0x21f31174460>



[159]:
```
figure,axis = plt.subplots(1,4,figsize=(18,18))

Picking_IMG = Acc_List[510]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,20,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)


Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)


for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.09*cv2.arcLength(cnt,True),True)

    Drawing_Contour = cv2.drawContours(Gray_IMG,[approx],0,(255,255,0),3)

    n_count = approx.ravel()
    i = 0

    for x_cor in n_count:
        x_cor = int(x_cor)
        Marker_IMG = cv2.drawMarker(Copy_Main_IMG,(x_cor,int(Copy_Main_IMG.
 ↪shape[1]/2)),(255,0,255),thickness=1)

    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)

            if (i == 0):
                cv2.putText(Gray_IMG,f"{x},{y}",(x,y),cv2.
 ↪FONT_HERSHEY_COMPLEX,0.4,(255,0,255))
            else:
                cv2.putText(Gray_IMG,"+",(x,y),cv2.FONT_HERSHEY_COMPLEX,0.
 ↪4,(0,0,255))

        i = i + 1



axis[0].set_xlabel(Gray_IMG.shape)
```

```
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Threshold_IMG_TOZERO.shape)
axis[1].set_ylabel(Threshold_IMG_TOZERO.size)
axis[1].set_title("TOZERO")
axis[1].imshow(Threshold_IMG_TOZERO,cmap="hot")

axis[2].set_xlabel(Drawing_Contour.shape)
axis[2].set_ylabel(Drawing_Contour.size)
axis[2].set_title("DRAWING")
axis[2].imshow(Drawing_Contour,cmap="hot")

axis[3].set_xlabel(Marker_IMG.shape)
axis[3].set_ylabel(Marker_IMG.size)
axis[3].set_title("LOC")
axis[3].imshow(Marker_IMG)
```

[159]: <matplotlib.image.AxesImage at 0x21f334fb1f0>

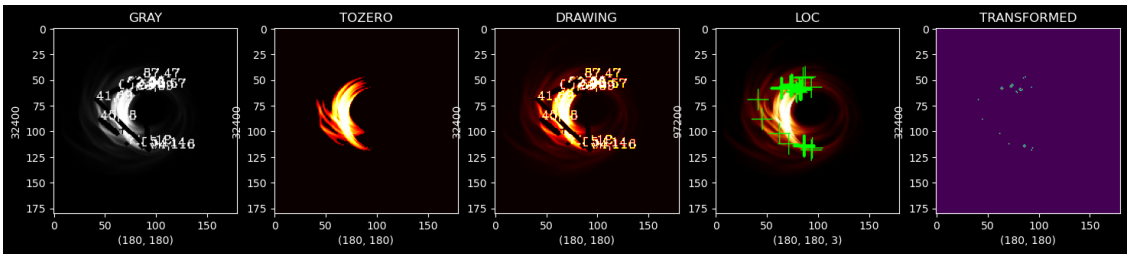


[160]:
```
figure,axis = plt.subplots(1,5,figsize=(18,18))

Picking_IMG = Acc_List[510]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,55,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)


Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)
```

```
for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.09*cv2.arcLength(cnt,True),True)

    Drawing_Contour = cv2.drawContours(Gray_IMG,[approx],0,(0,255,0),3)

    n_count = approx.ravel()
    i = 0


    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)

            Marker_IMG = cv2.
 ↪drawMarker(Copy_Main_IMG,(int(x),int(y)),(0,255,0),thickness=1)
            Trans_Empty_Zeros[int(y),int(x)] = 1

            if (i == 0):
                cv2.putText(Gray_IMG,f"{x},{y}",(x,y),cv2.
 ↪FONT_HERSHEY_COMPLEX,0.4,(255,0,255))
            else:
                cv2.putText(Gray_IMG,"+",(x,y),cv2.FONT_HERSHEY_COMPLEX,0.
 ↪4,(0,0,255))

        i = i + 1



axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Threshold_IMG_TOZERO.shape)
axis[1].set_ylabel(Threshold_IMG_TOZERO.size)
axis[1].set_title("TOZERO")
axis[1].imshow(Threshold_IMG_TOZERO,cmap="hot")

axis[2].set_xlabel(Drawing_Contour.shape)
axis[2].set_ylabel(Drawing_Contour.size)
axis[2].set_title("DRAWING")
axis[2].imshow(Drawing_Contour,cmap="hot")
```

```
axis[3].set_xlabel(Marker_IMG.shape)
axis[3].set_ylabel(Marker_IMG.size)
axis[3].set_title("LOC")
axis[3].imshow(Marker_IMG)

axis[4].set_xlabel(Trans_Empty_Zeros.shape)
axis[4].set_ylabel(Trans_Empty_Zeros.size)
axis[4].set_title("TRANSFORMED")
axis[4].imshow(Trans_Empty_Zeros)
```

[160]: <matplotlib.image.AxesImage at 0x21f3394bfa0>

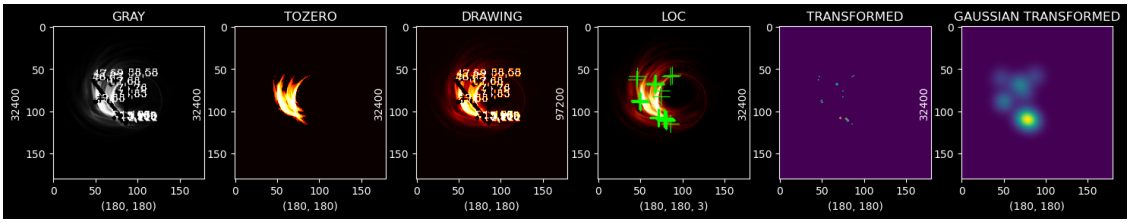


[161]:
```
figure,axis = plt.subplots(1,6,figsize=(18,18))

Picking_IMG = Acc_List[55]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,90,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)


Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)


for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.09*cv2.arcLength(cnt,True),True)

    Drawing_Contour = cv2.drawContours(Gray_IMG,[approx],0,(0,255,0),3)

    n_count = approx.ravel()
    i = 0


    for j in n_count:
```

```
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)

            Marker_IMG = cv2.
 ↪drawMarker(Copy_Main_IMG,(int(x),int(y)),(0,255,0),thickness=1)
            Trans_Empty_Zeros[int(y),int(x)] = 1

            if (i == 0):
                cv2.putText(Gray_IMG,f"{x},{y}",(x,y),cv2.
 ↪FONT_HERSHEY_COMPLEX,0.4,(255,0,255))
            else:
                cv2.putText(Gray_IMG,"+",(x,y),cv2.FONT_HERSHEY_COMPLEX,0.
 ↪4,(0,0,255))

        i = i + 1



Gaus_IMG = gaussian_filter(Trans_Empty_Zeros,sigma=8,truncate=6*6)

axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Threshold_IMG_TOZERO.shape)
axis[1].set_ylabel(Threshold_IMG_TOZERO.size)
axis[1].set_title("TOZERO")
axis[1].imshow(Threshold_IMG_TOZERO,cmap="hot")

axis[2].set_xlabel(Drawing_Contour.shape)
axis[2].set_ylabel(Drawing_Contour.size)
axis[2].set_title("DRAWING")
axis[2].imshow(Drawing_Contour,cmap="hot")

axis[3].set_xlabel(Marker_IMG.shape)
axis[3].set_ylabel(Marker_IMG.size)
axis[3].set_title("LOC")
axis[3].imshow(Marker_IMG)

axis[4].set_xlabel(Trans_Empty_Zeros.shape)
axis[4].set_ylabel(Trans_Empty_Zeros.size)
axis[4].set_title("TRANSFORMED")
axis[4].imshow(Trans_Empty_Zeros)
```

```
axis[5].set_xlabel(Gaus_IMG.shape)
axis[5].set_ylabel(Gaus_IMG.size)
axis[5].set_title("GAUSSIAN TRANSFORMED")
axis[5].imshow(Gaus_IMG)
```

[161]: <matplotlib.image.AxesImage at 0x21f4f888b20>

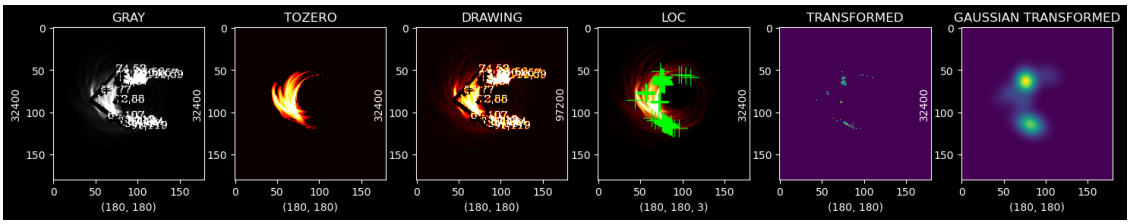


[162]:
```
figure,axis = plt.subplots(1,6,figsize=(18,18))

Picking_IMG = Acc_List[100]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,55,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)


Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)


for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.09*cv2.arcLength(cnt,True),True)

    Drawing_Contour = cv2.drawContours(Gray_IMG,[approx],0,(0,255,0),3)

    n_count = approx.ravel()
    i = 0


    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]
```

```
            string_coor = str(x) + " " + str(y)

            Marker_IMG = cv2.
 ↪drawMarker(Copy_Main_IMG,(int(x),int(y)),(0,255,0),thickness=1)
            Trans_Empty_Zeros[int(y),int(x)] = 1

            if (i == 0):
                cv2.putText(Gray_IMG,f"{x},{y}",(x,y),cv2.
 ↪FONT_HERSHEY_COMPLEX,0.4,(255,0,255))
            else:
                cv2.putText(Gray_IMG,"+",(x,y),cv2.FONT_HERSHEY_COMPLEX,0.
 ↪4,(0,0,255))

        i = i + 1



Gaus_IMG = gaussian_filter(Trans_Empty_Zeros,sigma=8,truncate=4*4)

axis[0].set_xlabel(Gray_IMG.shape)
axis[0].set_ylabel(Gray_IMG.size)
axis[0].set_title("GRAY")
axis[0].imshow(Gray_IMG,cmap="gray")

axis[1].set_xlabel(Threshold_IMG_TOZERO.shape)
axis[1].set_ylabel(Threshold_IMG_TOZERO.size)
axis[1].set_title("TOZERO")
axis[1].imshow(Threshold_IMG_TOZERO,cmap="hot")

axis[2].set_xlabel(Drawing_Contour.shape)
axis[2].set_ylabel(Drawing_Contour.size)
axis[2].set_title("DRAWING")
axis[2].imshow(Drawing_Contour,cmap="hot")

axis[3].set_xlabel(Marker_IMG.shape)
axis[3].set_ylabel(Marker_IMG.size)
axis[3].set_title("LOC")
axis[3].imshow(Marker_IMG)

axis[4].set_xlabel(Trans_Empty_Zeros.shape)
axis[4].set_ylabel(Trans_Empty_Zeros.size)
axis[4].set_title("TRANSFORMED")
axis[4].imshow(Trans_Empty_Zeros)


axis[5].set_xlabel(Gaus_IMG.shape)
axis[5].set_ylabel(Gaus_IMG.size)
```

```
axis[5].set_title("GAUSSIAN TRANSFORMED")
axis[5].imshow(Gaus_IMG)
```

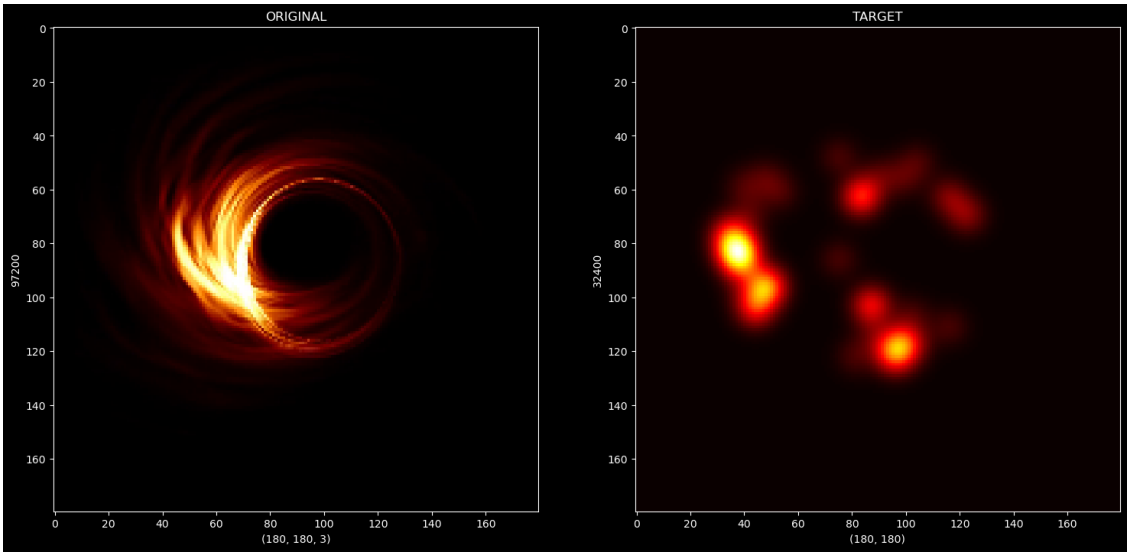
[162]: <matplotlib.image.AxesImage at 0x21f4fa0f8b0>



[163]:
```
figure,axis = plt.subplots(1,2,figsize=(18,18))

Picking_IMG = Acc_List[100]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,20,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)


Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)


for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.009*cv2.arcLength(cnt,True),True)
    n_count = approx.ravel()
    i = 0


    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)
            Trans_Empty_Zeros[int(y),int(x)] = 1

        i = i + 1
```

```
Gaus_IMG = gaussian_filter(Trans_Empty_Zeros,sigma=5,truncate=4*4)

axis[0].set_xlabel(Picking_IMG.shape)
axis[0].set_ylabel(Picking_IMG.size)
axis[0].set_title("ORIGINAL")
axis[0].imshow(Picking_IMG)

axis[1].set_xlabel(Gaus_IMG.shape)
axis[1].set_ylabel(Gaus_IMG.size)
axis[1].set_title("TARGET")
axis[1].imshow(Gaus_IMG,cmap="hot")
```

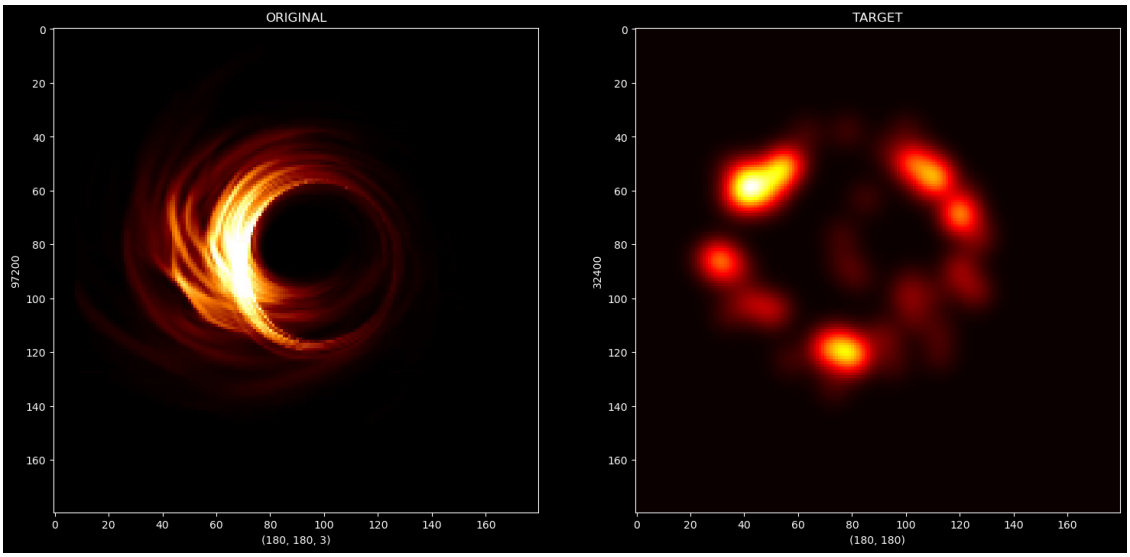
[163]: <matplotlib.image.AxesImage at 0x21f4fcfd640>



[164]:
```
figure,axis = plt.subplots(1,2,figsize=(18,18))

Picking_IMG = Acc_List[500]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,10,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)


Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)
```

```
for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.009*cv2.arcLength(cnt,True),True)
    n_count = approx.ravel()
    i = 0


    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)
            Trans_Empty_Zeros[int(y),int(x)] = 1

        i = i + 1



Gaus_IMG = gaussian_filter(Trans_Empty_Zeros,sigma=5,truncate=4*4)

axis[0].set_xlabel(Picking_IMG.shape)
axis[0].set_ylabel(Picking_IMG.size)
axis[0].set_title("ORIGINAL")
axis[0].imshow(Picking_IMG)

axis[1].set_xlabel(Gaus_IMG.shape)
axis[1].set_ylabel(Gaus_IMG.size)
axis[1].set_title("TARGET")
axis[1].imshow(Gaus_IMG,cmap="hot")
```

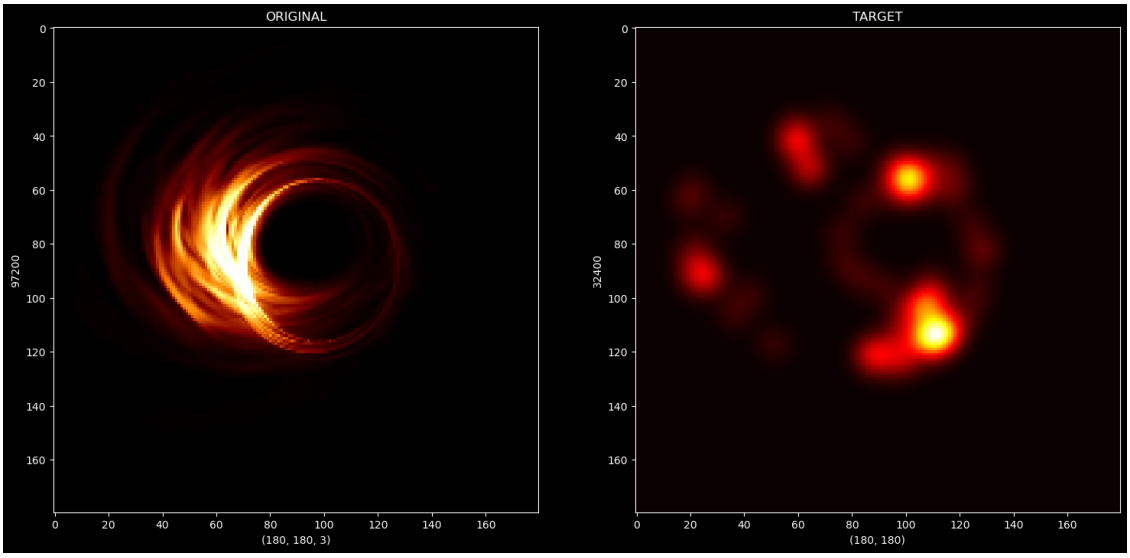
[164]: <matplotlib.image.AxesImage at 0x21f502300a0>



[165]:
```
figure,axis = plt.subplots(1,2,figsize=(18,18))

Picking_IMG = Acc_List[387]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,10,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)


Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)


for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.009*cv2.arcLength(cnt,True),True)
    n_count = approx.ravel()
    i = 0


    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)
            Trans_Empty_Zeros[int(y),int(x)] = 1
```

```
        i = i + 1



Gaus_IMG = gaussian_filter(Trans_Empty_Zeros,sigma=5,truncate=4*4)

axis[0].set_xlabel(Picking_IMG.shape)
axis[0].set_ylabel(Picking_IMG.size)
axis[0].set_title("ORIGINAL")
axis[0].imshow(Picking_IMG)

axis[1].set_xlabel(Gaus_IMG.shape)
axis[1].set_ylabel(Gaus_IMG.size)
axis[1].set_title("TARGET")
axis[1].imshow(Gaus_IMG,cmap="hot")
```

[165]: <matplotlib.image.AxesImage at 0x21f5033a4c0>

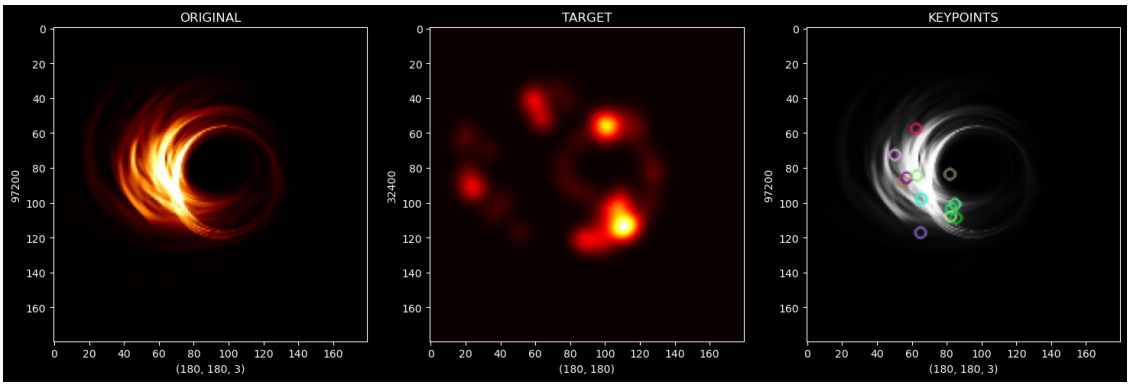


KEYPOINTS WITH COORDINATES

[166]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))

Picking_IMG = Acc_List[387]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,10,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)
```

```
Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)

for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.009*cv2.arcLength(cnt,True),True)
    n_count = approx.ravel()
    i = 0


    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)
            Trans_Empty_Zeros[int(y),int(x)] = 1

        i = i + 1



Gaus_IMG = gaussian_filter(Trans_Empty_Zeros,sigma=5,truncate=4*4)

Sift_Function = cv2.SIFT_create()
keypoints,desc = Sift_Function.detectAndCompute(Threshold_IMG_TOZERO,None)
Key_IMG = cv2.drawKeypoints(Gray_IMG,keypoints,Gray_IMG)

axis[0].set_xlabel(Picking_IMG.shape)
axis[0].set_ylabel(Picking_IMG.size)
axis[0].set_title("ORIGINAL")
axis[0].imshow(Picking_IMG)

axis[1].set_xlabel(Gaus_IMG.shape)
axis[1].set_ylabel(Gaus_IMG.size)
axis[1].set_title("TARGET")
axis[1].imshow(Gaus_IMG,cmap="hot")

axis[2].set_xlabel(Key_IMG.shape)
axis[2].set_ylabel(Key_IMG.size)
axis[2].set_title("KEYPOINTS")
axis[2].imshow(Key_IMG)
```

[166]: <matplotlib.image.AxesImage at 0x21f5047ad60>

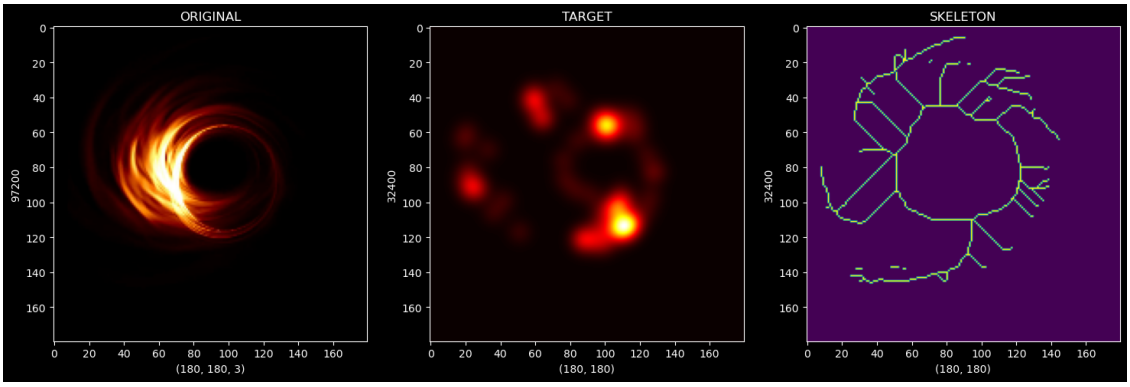


SKELETON WITH TARGET

[167]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))

Picking_IMG = Acc_List[387]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,10,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)


Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)

for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.009*cv2.arcLength(cnt,True),True)
    n_count = approx.ravel()
    i = 0


    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)
            Trans_Empty_Zeros[int(y),int(x)] = 1

        i = i + 1
```

```
Gaus_IMG = gaussian_filter(Trans_Empty_Zeros,sigma=5,truncate=4*4)
Array_Target = np.array(Gray_IMG > Gaus_IMG).astype(int)
Skeleton_IMG = skimage.morphology.skeletonize(Array_Target)

axis[0].set_xlabel(Picking_IMG.shape)
axis[0].set_ylabel(Picking_IMG.size)
axis[0].set_title("ORIGINAL")
axis[0].imshow(Picking_IMG)

axis[1].set_xlabel(Gaus_IMG.shape)
axis[1].set_ylabel(Gaus_IMG.size)
axis[1].set_title("TARGET")
axis[1].imshow(Gaus_IMG,cmap="hot")

axis[2].set_xlabel(Skeleton_IMG.shape)
axis[2].set_ylabel(Skeleton_IMG.size)
axis[2].set_title("SKELETON")
axis[2].imshow(Skeleton_IMG)
```

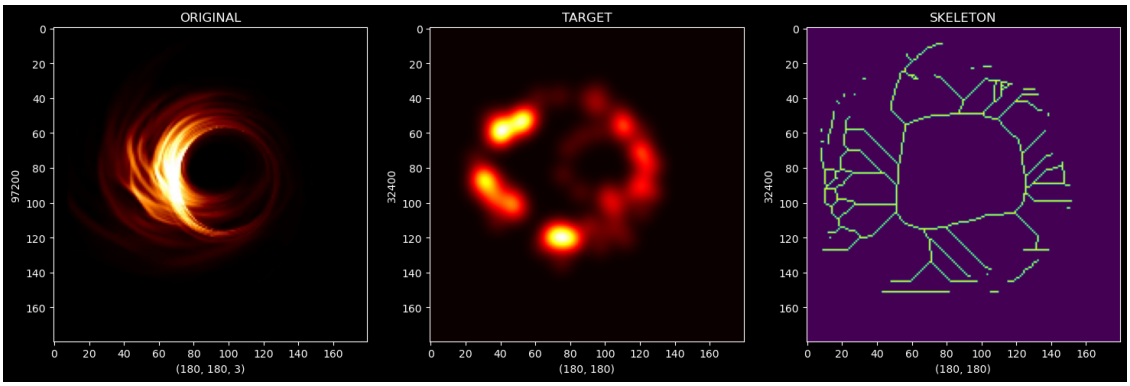
[167]: <matplotlib.image.AxesImage at 0x21f53e73880>



[168]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))

Picking_IMG = Acc_List[501]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,10,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)


Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)
```

```
for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.009*cv2.arcLength(cnt,True),True)
    n_count = approx.ravel()
    i = 0


    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)
            Trans_Empty_Zeros[int(y),int(x)] = 1

        i = i + 1



Gaus_IMG = gaussian_filter(Trans_Empty_Zeros,sigma=5,truncate=4*4)
Array_Target = np.array(Gray_IMG > Gaus_IMG).astype(int)
Skeleton_IMG = skimage.morphology.skeletonize(Array_Target)

axis[0].set_xlabel(Picking_IMG.shape)
axis[0].set_ylabel(Picking_IMG.size)
axis[0].set_title("ORIGINAL")
axis[0].imshow(Picking_IMG)

axis[1].set_xlabel(Gaus_IMG.shape)
axis[1].set_ylabel(Gaus_IMG.size)
axis[1].set_title("TARGET")
axis[1].imshow(Gaus_IMG,cmap="hot")

axis[2].set_xlabel(Skeleton_IMG.shape)
axis[2].set_ylabel(Skeleton_IMG.size)
axis[2].set_title("SKELETON")
axis[2].imshow(Skeleton_IMG)
```

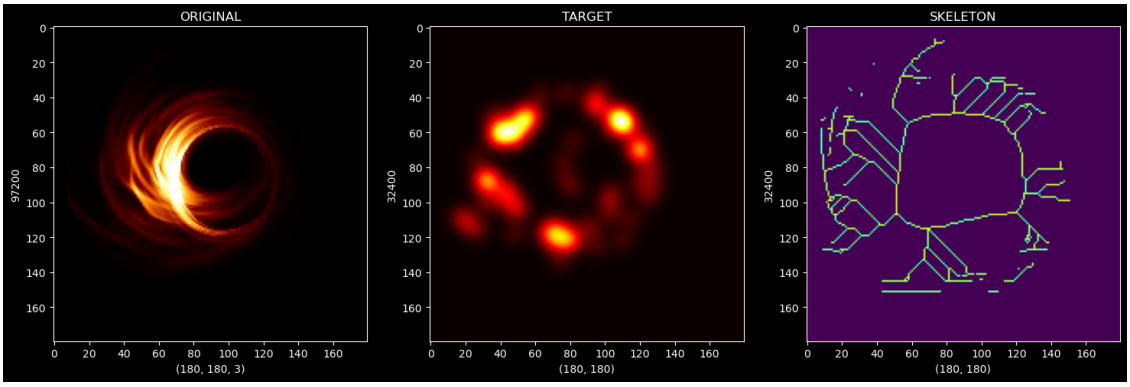
[168]: <matplotlib.image.AxesImage at 0x21f542db4f0>



[169]:
```
figure,axis = plt.subplots(1,3,figsize=(18,18))

Picking_IMG = Acc_List[499]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,10,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)


Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)

for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.009*cv2.arcLength(cnt,True),True)
    n_count = approx.ravel()
    i = 0


    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)
            Trans_Empty_Zeros[int(y),int(x)] = 1

        i = i + 1



Gaus_IMG = gaussian_filter(Trans_Empty_Zeros,sigma=5,truncate=4*4)
```

```
Array_Target = np.array(Gray_IMG > Gaus_IMG).astype(int)
Skeleton_IMG = skimage.morphology.skeletonize(Array_Target)

axis[0].set_xlabel(Picking_IMG.shape)
axis[0].set_ylabel(Picking_IMG.size)
axis[0].set_title("ORIGINAL")
axis[0].imshow(Picking_IMG)

axis[1].set_xlabel(Gaus_IMG.shape)
axis[1].set_ylabel(Gaus_IMG.size)
axis[1].set_title("TARGET")
axis[1].imshow(Gaus_IMG,cmap="hot")

axis[2].set_xlabel(Skeleton_IMG.shape)
axis[2].set_ylabel(Skeleton_IMG.size)
axis[2].set_title("SKELETON")
axis[2].imshow(Skeleton_IMG)
```

[169]: <matplotlib.image.AxesImage at 0x21f55828100>

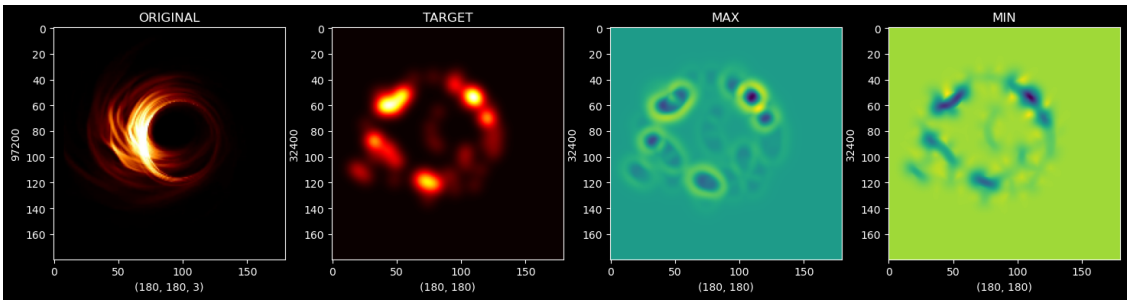


HESSIAN SPECTRUM WITH TARGET

[170]:
```
figure,axis = plt.subplots(1,4,figsize=(18,18))

Picking_IMG = Acc_List[499]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,10,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)


Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)
```

```
for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.009*cv2.arcLength(cnt,True),True)
    n_count = approx.ravel()
    i = 0


    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)
            Trans_Empty_Zeros[int(y),int(x)] = 1

        i = i + 1



Gaus_IMG = gaussian_filter(Trans_Empty_Zeros,sigma=5,truncate=4*4)

Hessian_IMG = hessian_matrix(Gaus_IMG,sigma=0.5,order="rc")
max_IMG,min_IMG = hessian_matrix_eigvals(Hessian_IMG)

axis[0].set_xlabel(Picking_IMG.shape)
axis[0].set_ylabel(Picking_IMG.size)
axis[0].set_title("ORIGINAL")
axis[0].imshow(Picking_IMG)

axis[1].set_xlabel(Gaus_IMG.shape)
axis[1].set_ylabel(Gaus_IMG.size)
axis[1].set_title("TARGET")
axis[1].imshow(Gaus_IMG,cmap="hot")

axis[2].set_xlabel(max_IMG.shape)
axis[2].set_ylabel(max_IMG.size)
axis[2].set_title("MAX")
axis[2].imshow(max_IMG)

axis[3].set_xlabel(min_IMG.shape)
axis[3].set_ylabel(min_IMG.size)
axis[3].set_title("MIN")
axis[3].imshow(min_IMG)
```

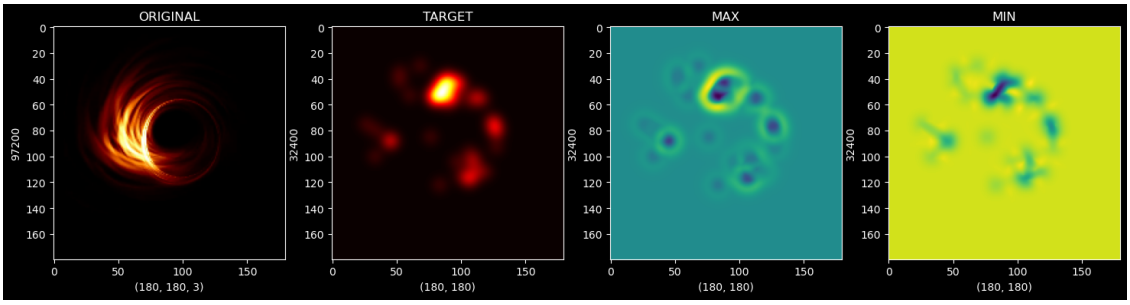
[170]: <matplotlib.image.AxesImage at 0x21f55cc5a00>



[171]:
```
figure,axis = plt.subplots(1,4,figsize=(18,18))

Picking_IMG = Acc_List[34]
Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
_,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,10,255,cv2.THRESH_TOZERO)
Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
 ↪CHAIN_APPROX_SIMPLE)


Copy_Main_IMG = Picking_IMG.copy()
Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
 ↪shape[1]),dtype=np.float32)

for cnt in Contours:

    approx = cv2.approxPolyDP(cnt,0.009*cv2.arcLength(cnt,True),True)
    n_count = approx.ravel()
    i = 0


    for j in n_count:
        if (i % 2 == 0):
            x = n_count[i]
            y = n_count[i + 1]

            string_coor = str(x) + " " + str(y)
            Trans_Empty_Zeros[int(y),int(x)] = 1

        i = i + 1



Gaus_IMG = gaussian_filter(Trans_Empty_Zeros,sigma=5,truncate=4*4)

Hessian_IMG = hessian_matrix(Gaus_IMG,sigma=0.5,order="rc")
```

```
max_IMG,min_IMG = hessian_matrix_eigvals(Hessian_IMG)

axis[0].set_xlabel(Picking_IMG.shape)
axis[0].set_ylabel(Picking_IMG.size)
axis[0].set_title("ORIGINAL")
axis[0].imshow(Picking_IMG)

axis[1].set_xlabel(Gaus_IMG.shape)
axis[1].set_ylabel(Gaus_IMG.size)
axis[1].set_title("TARGET")
axis[1].imshow(Gaus_IMG,cmap="hot")

axis[2].set_xlabel(max_IMG.shape)
axis[2].set_ylabel(max_IMG.size)
axis[2].set_title("MAX")
axis[2].imshow(max_IMG)

axis[3].set_xlabel(min_IMG.shape)
axis[3].set_ylabel(min_IMG.size)
axis[3].set_title("MIN")
axis[3].imshow(min_IMG)
```

[171]: <matplotlib.image.AxesImage at 0x21f560c7fa0>



DATA PROCESS

TRANSFORMATION

[172]:
```
Original_List = []
Target_List = []

for image_x in Acc_List:

    Picking_IMG = image_x
    Gray_IMG = cv2.cvtColor(Picking_IMG,cv2.COLOR_RGB2GRAY)
    _,Threshold_IMG_TOZERO = cv2.threshold(Gray_IMG,10,255,cv2.THRESH_TOZERO)
```

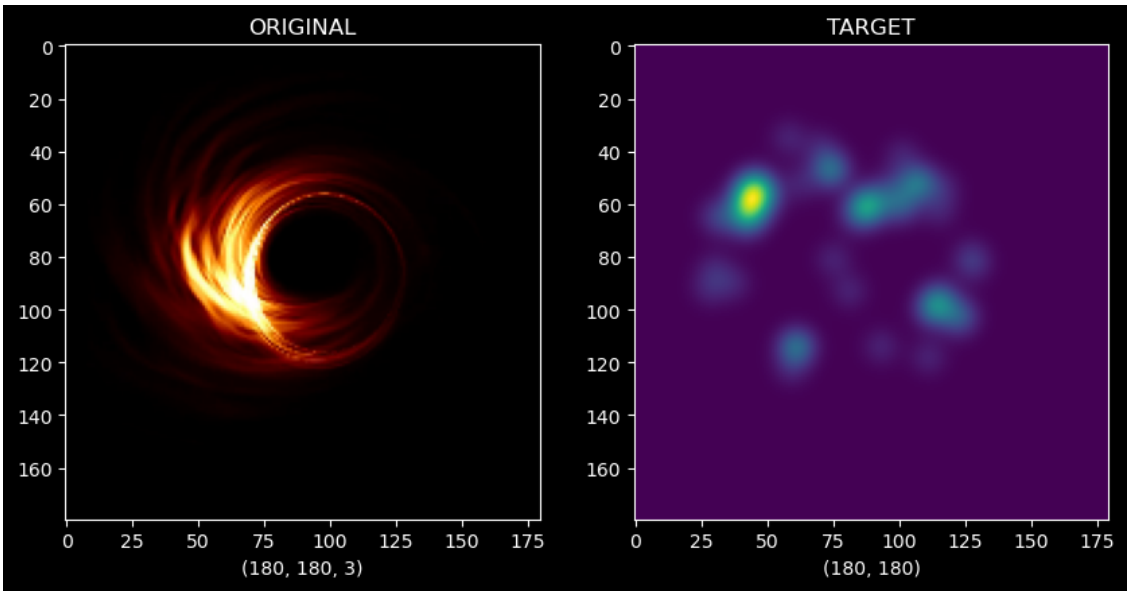
```
    Contours,_ = cv2.findContours(Threshold_IMG_TOZERO,cv2.RETR_TREE,cv2.
↪CHAIN_APPROX_SIMPLE)


    Copy_Main_IMG = Picking_IMG.copy()
    Trans_Empty_Zeros = np.zeros((Copy_Main_IMG.shape[0],Copy_Main_IMG.
↪shape[1]),dtype=np.float32)

    for cnt in Contours:

        approx = cv2.approxPolyDP(cnt,0.009*cv2.arcLength(cnt,True),True)
        n_count = approx.ravel()
        i = 0


        for j in n_count:
            if (i % 2 == 0):
                x = n_count[i]
                y = n_count[i + 1]

                string_coor = str(x) + " " + str(y)
                Trans_Empty_Zeros[int(y),int(x)] = 1

            i = i + 1



    Gaus_IMG = gaussian_filter(Trans_Empty_Zeros,sigma=5,truncate=4*4)

    Original_List.append(Picking_IMG)
    Target_List.append(Gaus_IMG)
```

[173]:
```
print("WHEN IT IS ARRAY ORIGINAL SHAPE: ",np.shape(np.array(Original_List)))
print("WHEN IT IS ARRAY TARGET SHAPE: ",np.shape(np.array(Target_List)))
```

```
WHEN IT IS ARRAY ORIGINAL SHAPE:  (513, 180, 180, 3)
WHEN IT IS ARRAY TARGET SHAPE:  (513, 180, 180)
```

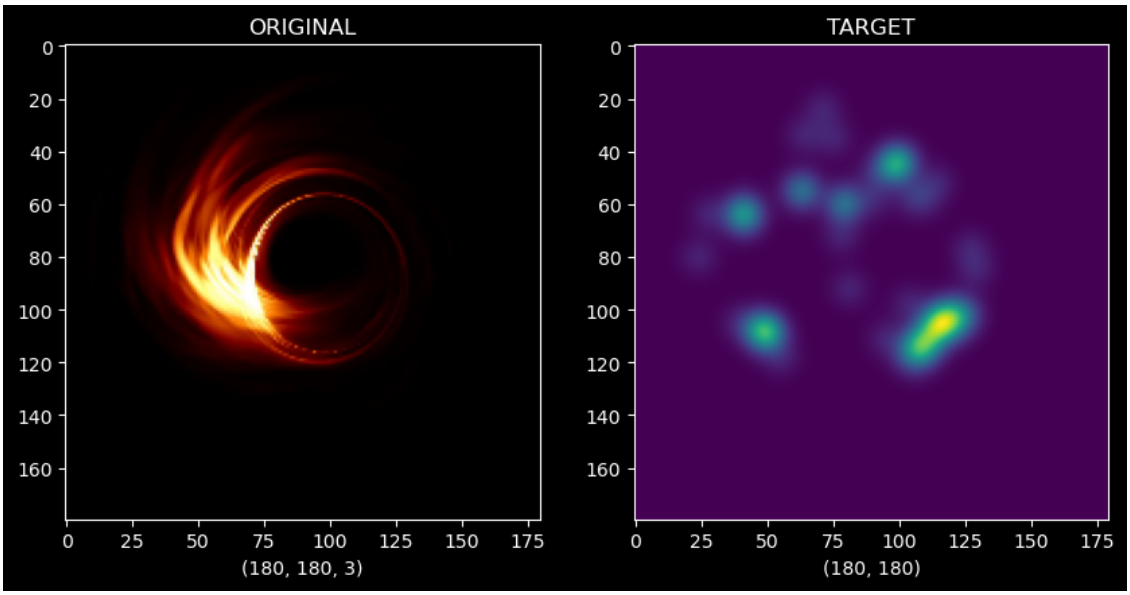
[174]:
```
figure,axis = plt.subplots(1,2,figsize=(10,10))

axis[0].imshow(Original_List[100])
axis[0].set_xlabel(Original_List[100].shape)
axis[0].set_title("ORIGINAL")
axis[1].imshow(Target_List[100])
axis[1].set_xlabel(Target_List[100].shape)
axis[1].set_title("TARGET")
```

[174]: Text(0.5, 1.0, 'TARGET')



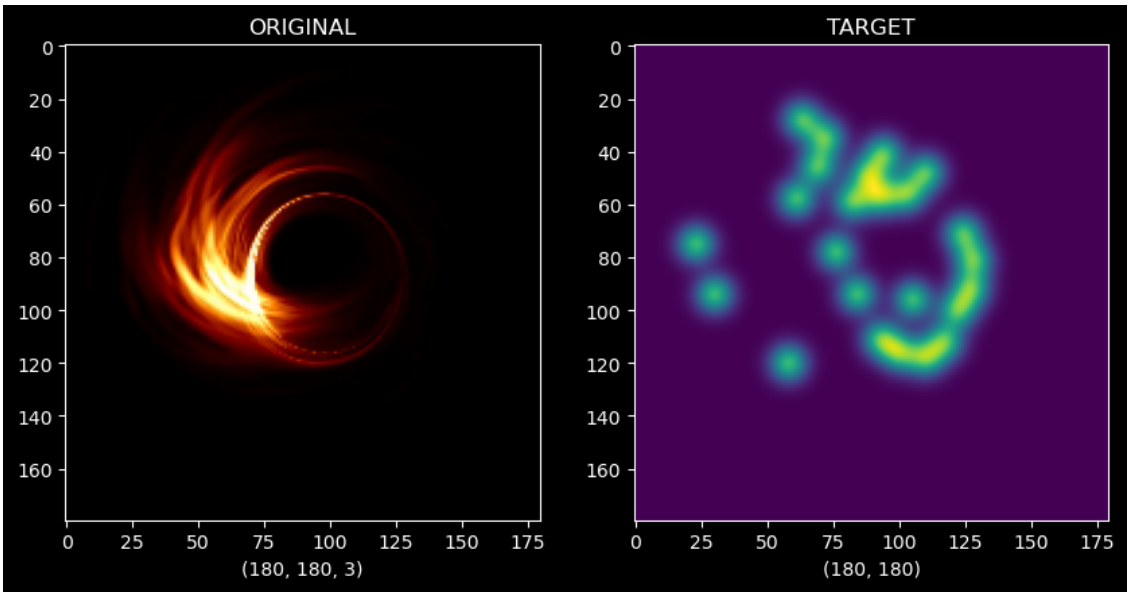
[175]:
```
figure,axis = plt.subplots(1,2,figsize=(10,10))

axis[0].imshow(Original_List[1])
axis[0].set_xlabel(Original_List[1].shape)
axis[0].set_title("ORIGINAL")
axis[1].imshow(Target_List[1])
axis[1].set_xlabel(Target_List[1].shape)
axis[1].set_title("TARGET")
```

[175]: Text(0.5, 1.0, 'TARGET')



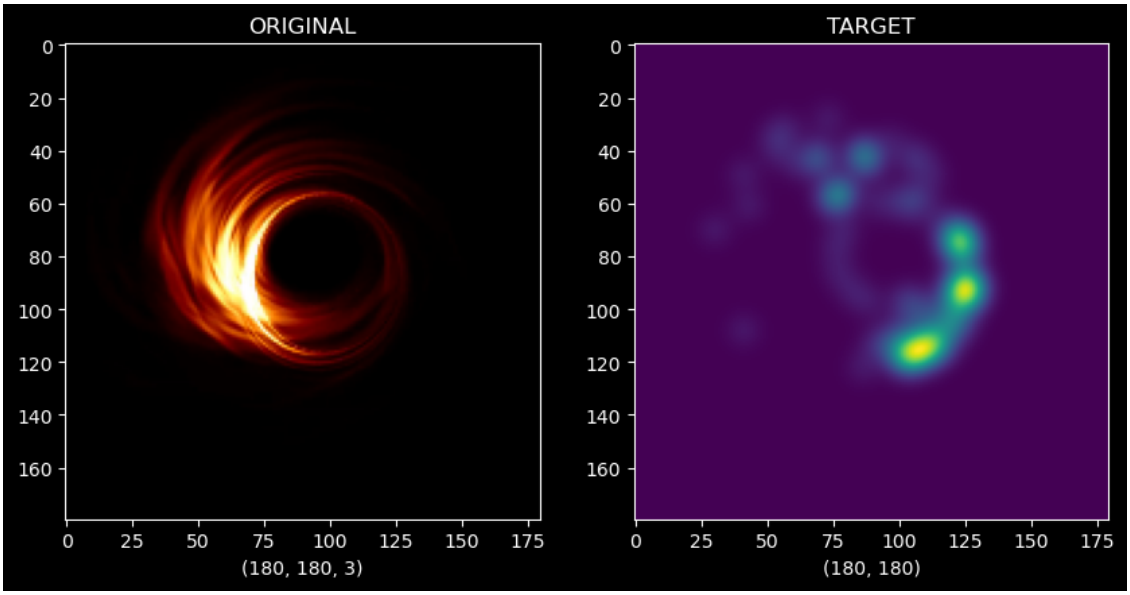
[176]:
```
figure,axis = plt.subplots(1,2,figsize=(10,10))

axis[0].imshow(Original_List[10])
axis[0].set_xlabel(Original_List[10].shape)
axis[0].set_title("ORIGINAL")
axis[1].imshow(Target_List[10])
axis[1].set_xlabel(Target_List[10].shape)
axis[1].set_title("TARGET")
```

[176]:
```
Text(0.5, 1.0, 'TARGET')
```

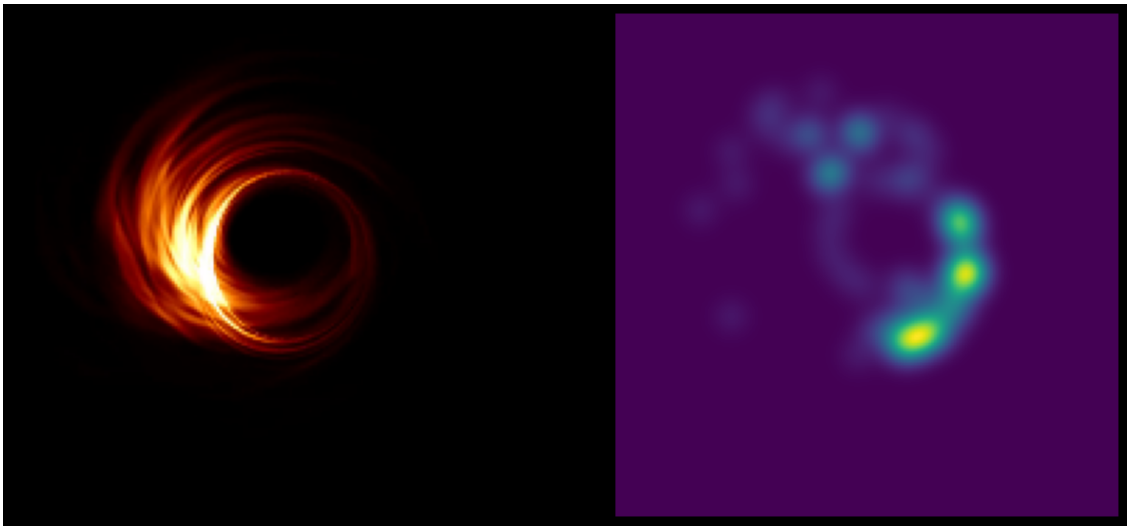


[177]:
```
figure,axis = plt.subplots(1,2,figsize=(10,10))

axis[0].imshow(Original_List[300])
axis[0].set_xlabel(Original_List[300].shape)
axis[0].set_title("ORIGINAL")
axis[1].imshow(Target_List[300])
axis[1].set_xlabel(Target_List[300].shape)
axis[1].set_title("TARGET")
```

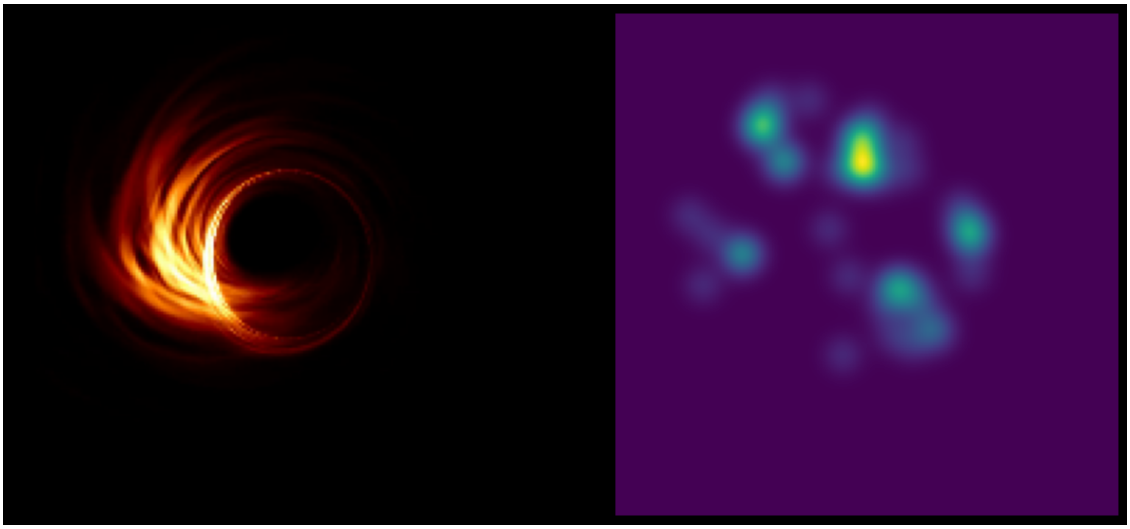
[177]:
```
Text(0.5, 1.0, 'TARGET')
```



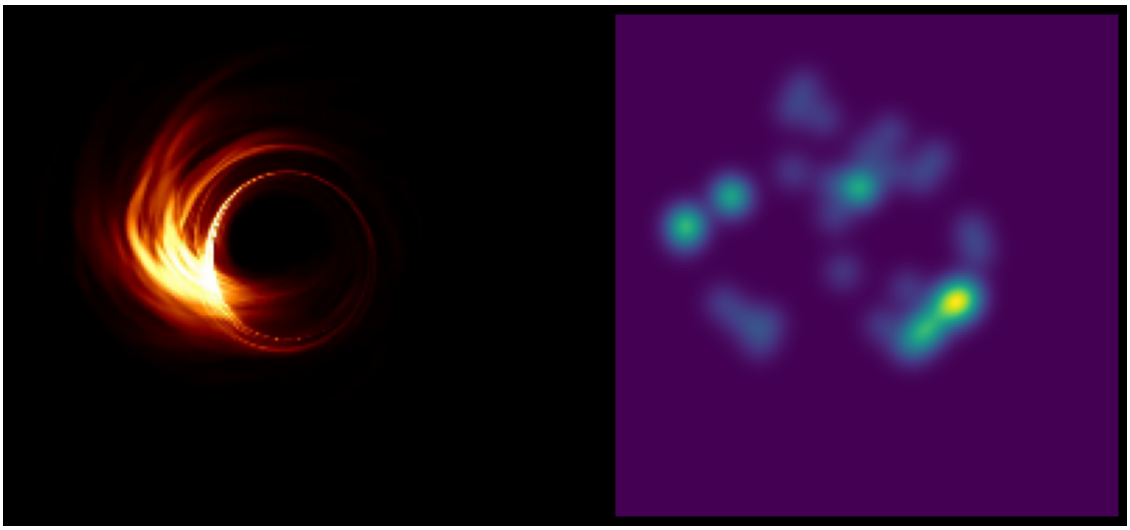
[178]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))

Count_IMG = 300

axis[0].imshow(Original_List[Count_IMG])
axis[0].axis("off")
axis[1].imshow(Target_List[Count_IMG])
axis[1].axis("off")

plt.savefig(f"IMG{Count_IMG}.png")
```



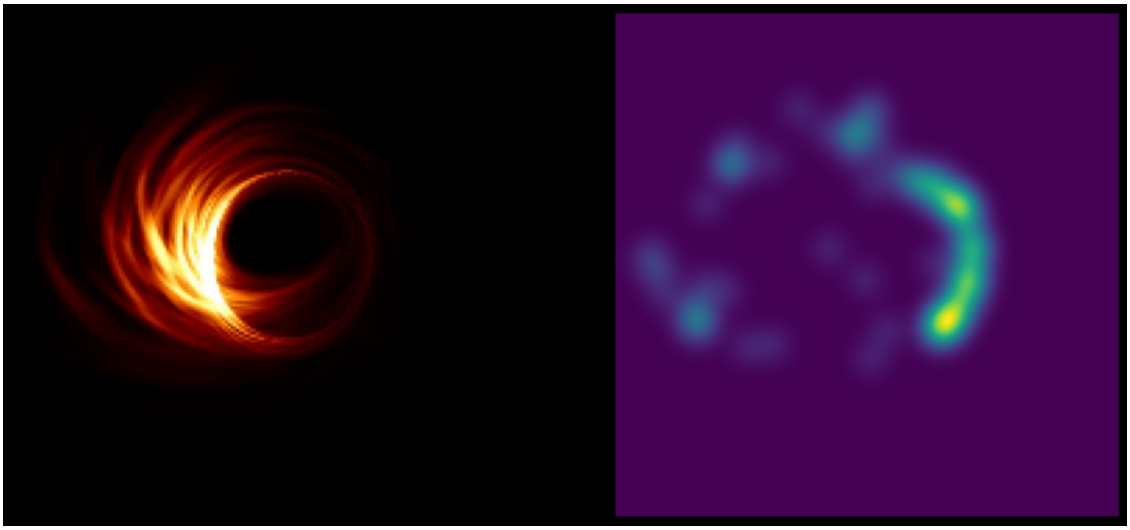
[179]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))

Count_IMG = 30

axis[0].imshow(Original_List[Count_IMG])
axis[0].axis("off")
axis[1].imshow(Target_List[Count_IMG])
axis[1].axis("off")

plt.savefig(f"IMG{Count_IMG}.png")
```



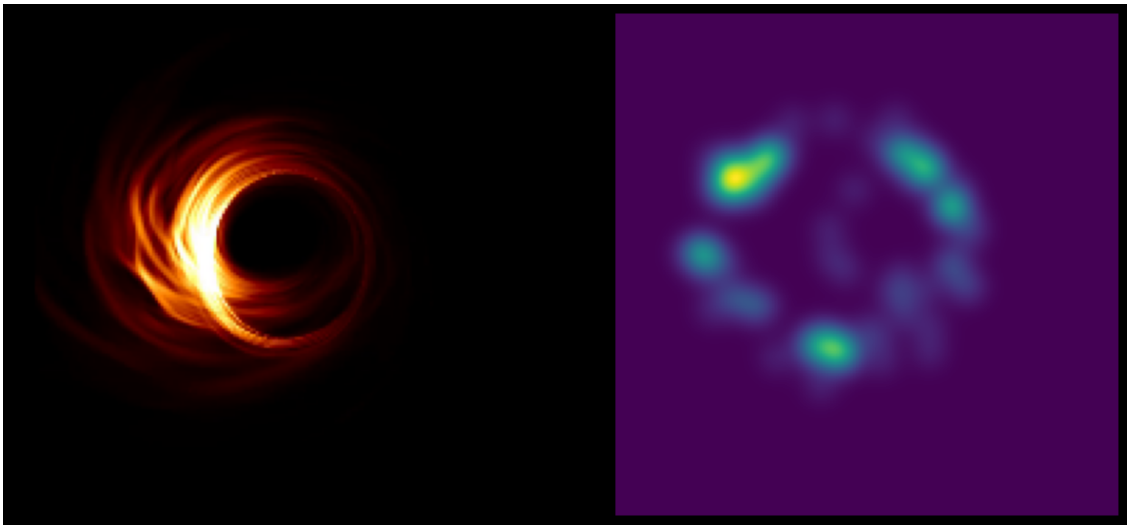
[180]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))

Count_IMG = 3

axis[0].imshow(Original_List[Count_IMG])
axis[0].axis("off")
axis[1].imshow(Target_List[Count_IMG])
axis[1].axis("off")

plt.savefig(f"IMG{Count_IMG}.png")
```



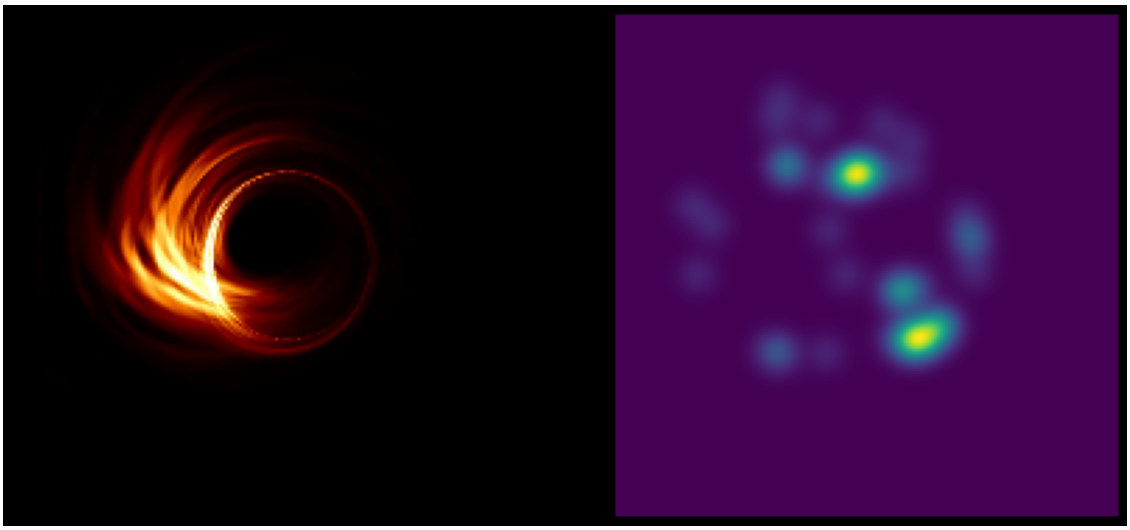
[181]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))

Count_IMG = 450

axis[0].imshow(Original_List[Count_IMG])
axis[0].axis("off")
axis[1].imshow(Target_List[Count_IMG])
axis[1].axis("off")

plt.savefig(f"IMG{Count_IMG}.png")
```



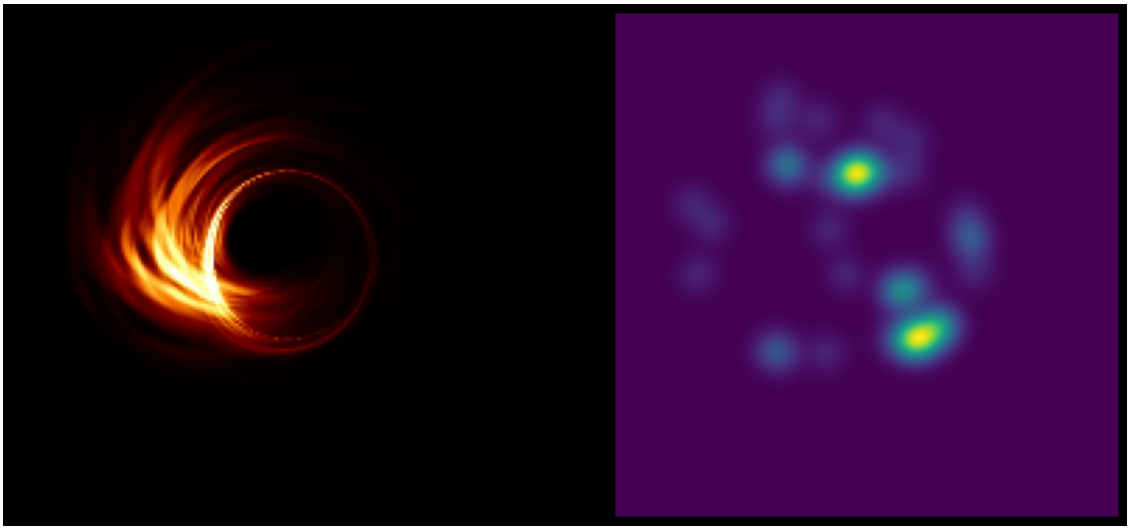
[182]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))

Count_IMG = 500

axis[0].imshow(Original_List[Count_IMG])
axis[0].axis("off")
axis[1].imshow(Target_List[Count_IMG])
axis[1].axis("off")

plt.savefig(f"IMG{Count_IMG}.png")
```



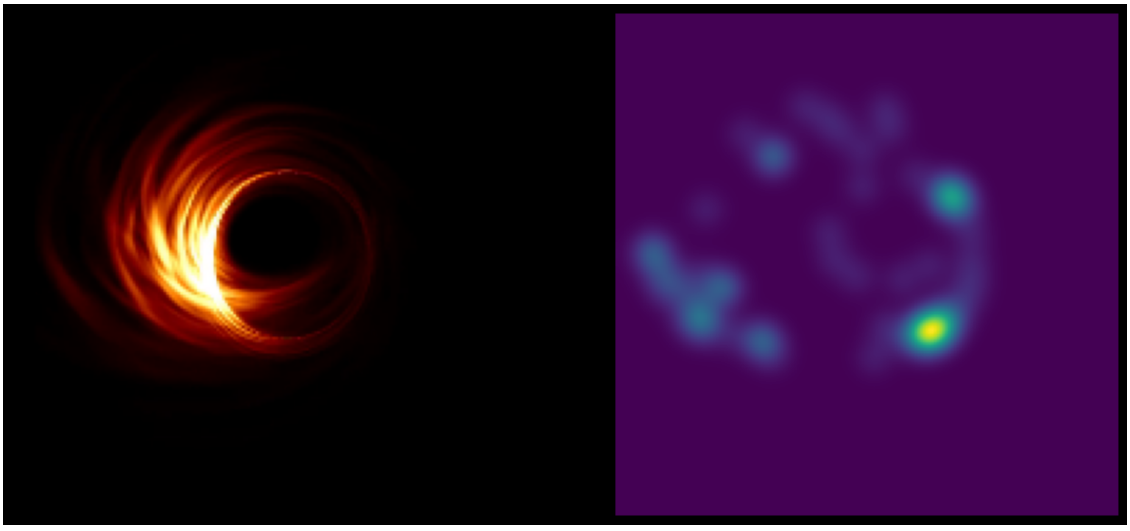
[183]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))

Count_IMG = 22

axis[0].imshow(Original_List[Count_IMG])
axis[0].axis("off")
axis[1].imshow(Target_List[Count_IMG])
axis[1].axis("off")

plt.savefig(f"IMG{Count_IMG}.png")
```



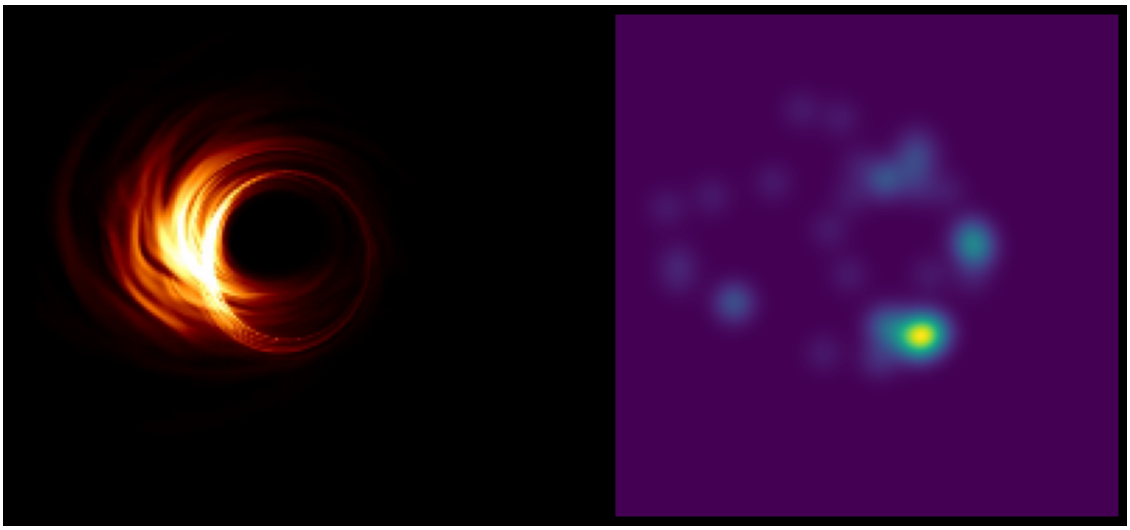
[184]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))

Count_IMG = 22

axis[0].imshow(Original_List[Count_IMG])
axis[0].axis("off")
axis[1].imshow(Target_List[Count_IMG])
axis[1].axis("off")

plt.savefig(f"IMG{Count_IMG}.png")
```



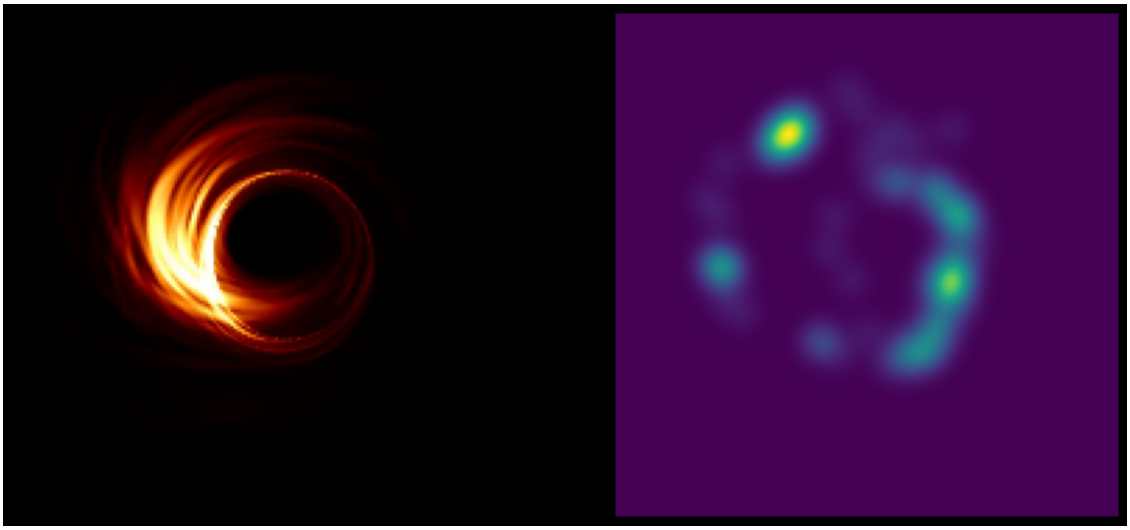
[185]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))

Count_IMG = 444

axis[0].imshow(Original_List[Count_IMG])
axis[0].axis("off")
axis[1].imshow(Target_List[Count_IMG])
axis[1].axis("off")

plt.savefig(f"IMG{Count_IMG}.png")
```



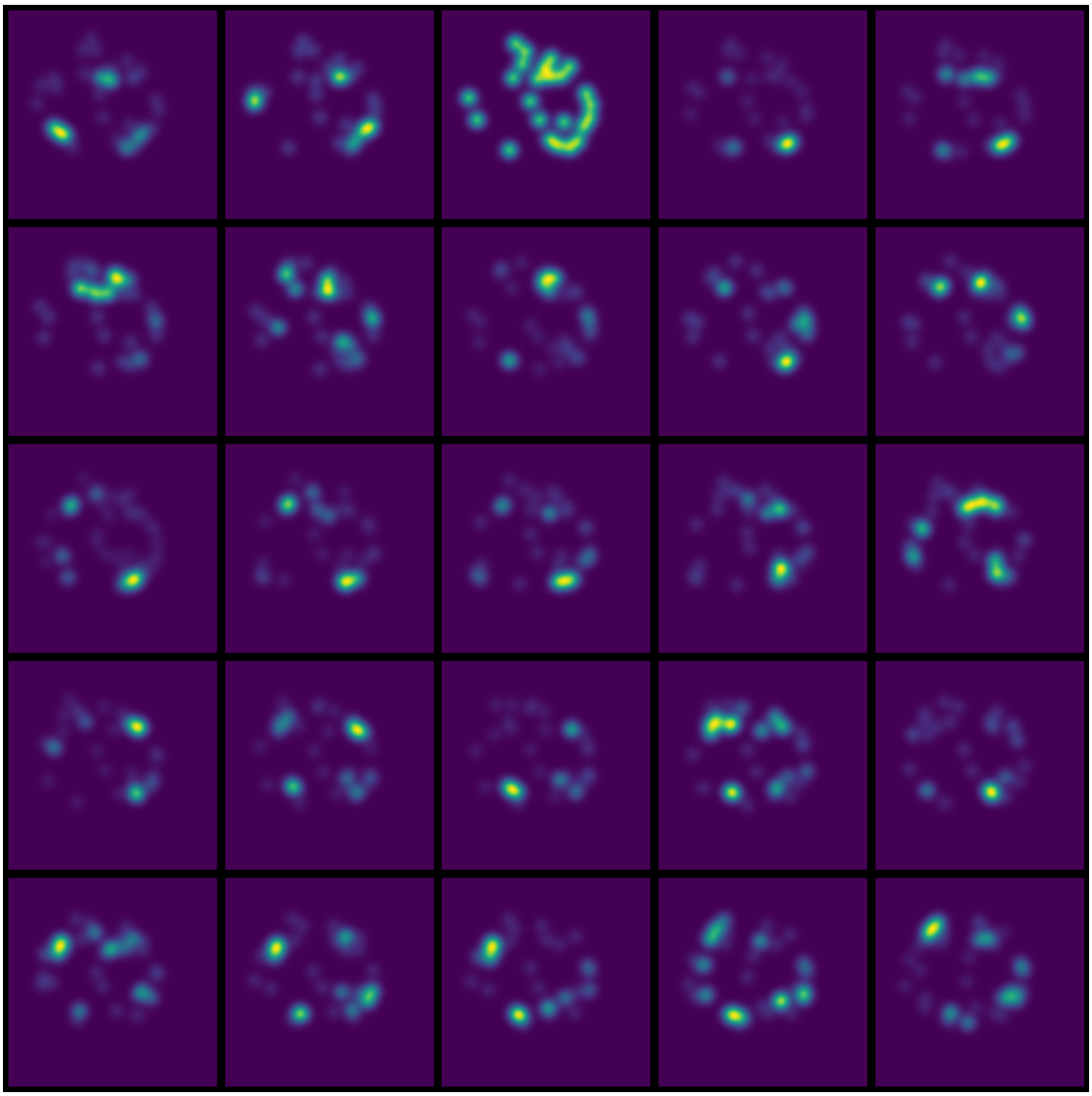
[186]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))

Count_IMG = 401

axis[0].imshow(Original_List[Count_IMG])
axis[0].axis("off")
axis[1].imshow(Target_List[Count_IMG])
axis[1].axis("off")

plt.savefig(f"IMG{Count_IMG}.png")
```



[187]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))

Count_IMG = 230

axis[0].imshow(Original_List[Count_IMG])
axis[0].axis("off")
axis[1].imshow(Target_List[Count_IMG])
axis[1].axis("off")

plt.savefig(f"IMG{Count_IMG}.png")
```



[188]:
```
figure,axis = plt.subplots(5,5,figsize=(20,20))

for indexing,operations in enumerate(axis.flat):

    Picking_IMG = Target_List[indexing*5]

    operations.imshow(Picking_IMG)
    operations.axis("off")
    plt.savefig(f"IMG_TOTAL{indexing}.png")

plt.tight_layout()
plt.show()
```



TO ARRAY

[189]:
```
Train_Set = np.array(Original_List,dtype="float32")
Transformation_Set = np.array(Target_List,dtype="float32")

Train_Set = Train_Set / 255.
Transformation_Set = Transformation_Set / 255.
```

[190]:
```
print(Train_Set.shape)
print(Transformation_Set.shape)
```

```
(513, 180, 180, 3)
(513, 180, 180)
```

MODEL PROCESS

[191]:
```
Checkpoint_Model = tf.keras.callbacks.ModelCheckpoint(monitor="val_accuracy",
                                                      save_best_only=True,
                                                      save_weights_only=True,
                                                      filepath="./modelcheck")
Reduce_Model = tf.keras.callbacks.ReduceLROnPlateau(monitor="val_accuracy",
                                                    factor=0.02,
                                                     min_delta=0.0001,
                                                    patience=5)

compile_loss = "binary_crossentropy"
compile_optimizer = Adam(lr=0.0000001)
output_class = 1
```

[192]:
```
Encoder_G = Sequential()
Encoder_G.add(Conv2D(32,(7,7),kernel_initializer = 'he_normal',use_bias=True))
Encoder_G.add(BatchNormalization())
Encoder_G.add(ReLU())
#
Encoder_G.add(Conv2D(64,(7,7),kernel_initializer = 'he_normal',use_bias=True))
Encoder_G.add(BatchNormalization())
Encoder_G.add(ReLU())
#
Encoder_G.add(Conv2D(128,(2,2),kernel_initializer = 'he_normal',use_bias=True))
Encoder_G.add(BatchNormalization())
Encoder_G.add(ReLU())


Decoder_G = Sequential()
Decoder_G.add(Conv2DTranspose(64,(7,7)))
Decoder_G.add(ReLU())
#
Decoder_G.add(Conv2DTranspose(32,(7,7)))
Decoder_G.add(ReLU())
#
Decoder_G.add(Conv2DTranspose(output_class,(2,2)))
```

[193]:
```
Auto_Encoder = Sequential([Encoder_G,Decoder_G])
Auto_Encoder.
 ↪compile(loss=compile_loss,optimizer=compile_optimizer,metrics=["mse"])
```

[194]:
```
Model_AutoEncoder = Auto_Encoder.
 ↪fit(Train_Set,Transformation_Set,epochs=25,callbacks=[Checkpoint_Model,Reduce_Model])
```

```
Epoch 1/25
17/17 [==============================] - ETA: 0s - loss: 0.2353 - mse: 0.1020
WARNING:tensorflow:Can save best model only with val_accuracy available,
```

```
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 223s 13s/step - loss: 0.2353 - mse:
0.1020 - lr: 1.0000e-07
Epoch 2/25
17/17 [==============================] - ETA: 0s - loss: 0.2223 - mse: 0.1019
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 236s 14s/step - loss: 0.2223 - mse:
0.1019 - lr: 1.0000e-07
Epoch 3/25
17/17 [==============================] - ETA: 0s - loss: 0.2118 - mse: 0.1021
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 225s 13s/step - loss: 0.2118 - mse:
0.1021 - lr: 1.0000e-07
Epoch 4/25
17/17 [==============================] - ETA: 0s - loss: 0.2025 - mse: 0.1025
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 254s 15s/step - loss: 0.2025 - mse:
0.1025 - lr: 1.0000e-07
Epoch 5/25
17/17 [==============================] - ETA: 0s - loss: 0.1938 - mse: 0.1030
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 254s 15s/step - loss: 0.1938 - mse:
0.1030 - lr: 1.0000e-07
Epoch 6/25
17/17 [==============================] - ETA: 0s - loss: 0.1869 - mse: 0.1036
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 257s 15s/step - loss: 0.1869 - mse:
0.1036 - lr: 1.0000e-07
Epoch 7/25
17/17 [==============================] - ETA: 0s - loss: 0.1808 - mse: 0.1042
WARNING:tensorflow:Can save best model only with val_accuracy available,
```

```
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 233s 14s/step - loss: 0.1808 - mse:
0.1042 - lr: 1.0000e-07
Epoch 8/25
17/17 [==============================] - ETA: 0s - loss: 0.1757 - mse: 0.1048
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 250s 15s/step - loss: 0.1757 - mse:
0.1048 - lr: 1.0000e-07
Epoch 9/25
17/17 [==============================] - ETA: 0s - loss: 0.1704 - mse: 0.1056
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 267s 16s/step - loss: 0.1704 - mse:
0.1056 - lr: 1.0000e-07
Epoch 10/25
17/17 [==============================] - ETA: 0s - loss: 0.1653 - mse: 0.1064
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 264s 16s/step - loss: 0.1653 - mse:
0.1064 - lr: 1.0000e-07
Epoch 11/25
17/17 [==============================] - ETA: 0s - loss: 0.1602 - mse: 0.1073
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 244s 14s/step - loss: 0.1602 - mse:
0.1073 - lr: 1.0000e-07
Epoch 12/25
17/17 [==============================] - ETA: 0s - loss: 0.1554 - mse: 0.1083
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 254s 15s/step - loss: 0.1554 - mse:
0.1083 - lr: 1.0000e-07
Epoch 13/25
17/17 [==============================] - ETA: 0s - loss: 0.1509 - mse: 0.1092
WARNING:tensorflow:Can save best model only with val_accuracy available,
```

```
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 252s 15s/step - loss: 0.1509 - mse:
0.1092 - lr: 1.0000e-07
Epoch 14/25
17/17 [==============================] - ETA: 0s - loss: 0.1463 - mse: 0.1103
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 246s 14s/step - loss: 0.1463 - mse:
0.1103 - lr: 1.0000e-07
Epoch 15/25
17/17 [==============================] - ETA: 0s - loss: 0.1414 - mse: 0.1116
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 257s 15s/step - loss: 0.1414 - mse:
0.1116 - lr: 1.0000e-07
Epoch 16/25
17/17 [==============================] - ETA: 0s - loss: 0.1368 - mse: 0.1129
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 248s 14s/step - loss: 0.1368 - mse:
0.1129 - lr: 1.0000e-07
Epoch 17/25
17/17 [==============================] - ETA: 0s - loss: 0.1322 - mse: 0.1143
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 257s 15s/step - loss: 0.1322 - mse:
0.1143 - lr: 1.0000e-07
Epoch 18/25
17/17 [==============================] - ETA: 0s - loss: 0.1275 - mse: 0.1159
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 252s 15s/step - loss: 0.1275 - mse:
0.1159 - lr: 1.0000e-07
Epoch 19/25
17/17 [==============================] - ETA: 0s - loss: 0.1226 - mse: 0.1177
WARNING:tensorflow:Can save best model only with val_accuracy available,
```

```
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 278s 17s/step - loss: 0.1226 - mse:
0.1177 - lr: 1.0000e-07
Epoch 20/25
17/17 [==============================] - ETA: 0s - loss: 0.1187 - mse: 0.1192
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 252s 15s/step - loss: 0.1187 - mse:
0.1192 - lr: 1.0000e-07
Epoch 21/25
17/17 [==============================] - ETA: 0s - loss: 0.1149 - mse: 0.1207
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 246s 14s/step - loss: 0.1149 - mse:
0.1207 - lr: 1.0000e-07
Epoch 22/25
17/17 [==============================] - ETA: 0s - loss: 0.1111 - mse: 0.1224
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 278s 16s/step - loss: 0.1111 - mse:
0.1224 - lr: 1.0000e-07
Epoch 23/25
17/17 [==============================] - ETA: 0s - loss: 0.1072 - mse: 0.1241
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 288s 17s/step - loss: 0.1072 - mse:
0.1241 - lr: 1.0000e-07
Epoch 24/25
17/17 [==============================] - ETA: 0s - loss: 0.1038 - mse: 0.1258
WARNING:tensorflow:Can save best model only with val_accuracy available,
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 269s 16s/step - loss: 0.1038 - mse:
0.1258 - lr: 1.0000e-07
Epoch 25/25
17/17 [==============================] - ETA: 0s - loss: 0.1009 - mse: 0.1273
WARNING:tensorflow:Can save best model only with val_accuracy available,
```

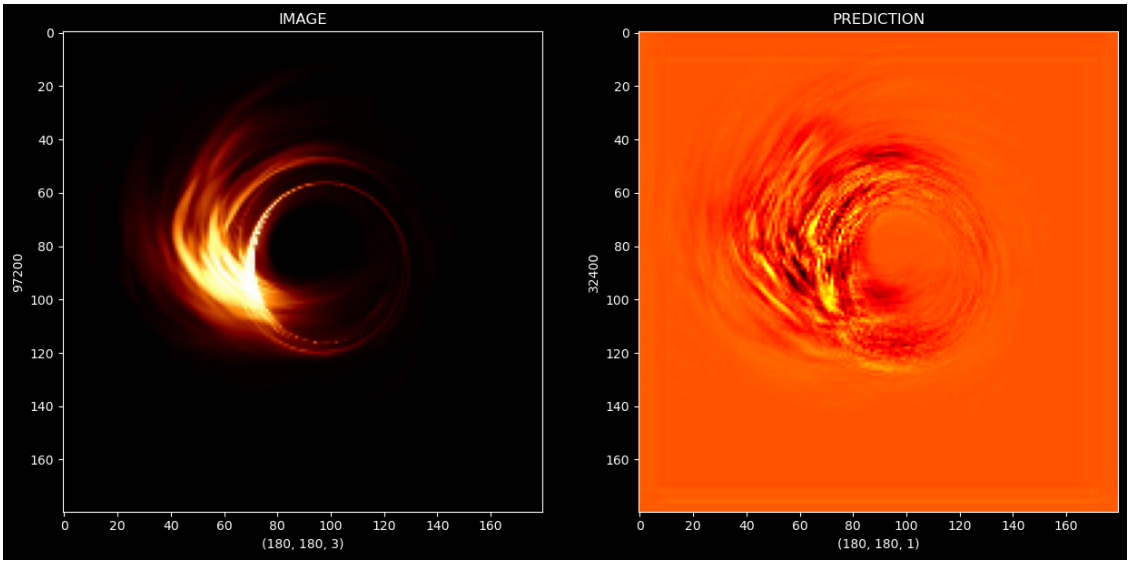
```
skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric
`val_accuracy` which is not available. Available metrics are: loss,mse,lr
17/17 [==============================] - 242s 14s/step - loss: 0.1009 - mse:
0.1273 - lr: 1.0000e-07
```

[195]:
```
Prediction_IMG = Auto_Encoder.predict(Train_Set)
```

```
17/17 [==============================] - 77s 4s/step
```

[196]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))
prediction_img_number = 1

Original_Img = Train_Set[prediction_img_number]
Predict_Target = Prediction_IMG[prediction_img_number]

axis[0].imshow(Original_Img)
axis[0].set_xlabel(Original_Img.shape)
axis[0].set_ylabel(Original_Img.size)
axis[0].set_title("IMAGE")
axis[1].imshow(Predict_Target,cmap="hot")
axis[1].set_xlabel(Predict_Target.shape)
axis[1].set_ylabel(Predict_Target.size)
axis[1].set_title("PREDICTION")
```

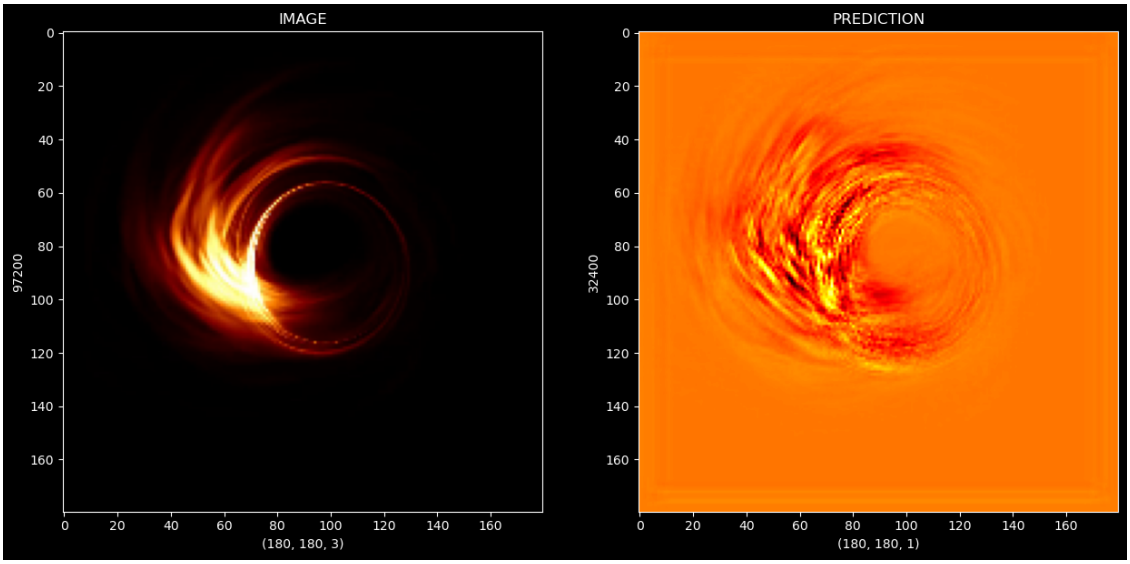
[196]:
```
Text(0.5, 1.0, 'PREDICTION')
```



[197]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))
prediction_img_number = 5
```

```
Original_Img = Train_Set[prediction_img_number]
Predict_Target = Prediction_IMG[prediction_img_number]

axis[0].imshow(Original_Img)
axis[0].set_xlabel(Original_Img.shape)
axis[0].set_ylabel(Original_Img.size)
axis[0].set_title("IMAGE")
axis[1].imshow(Predict_Target,cmap="hot")
axis[1].set_xlabel(Predict_Target.shape)
axis[1].set_ylabel(Predict_Target.size)
axis[1].set_title("PREDICTION")
```

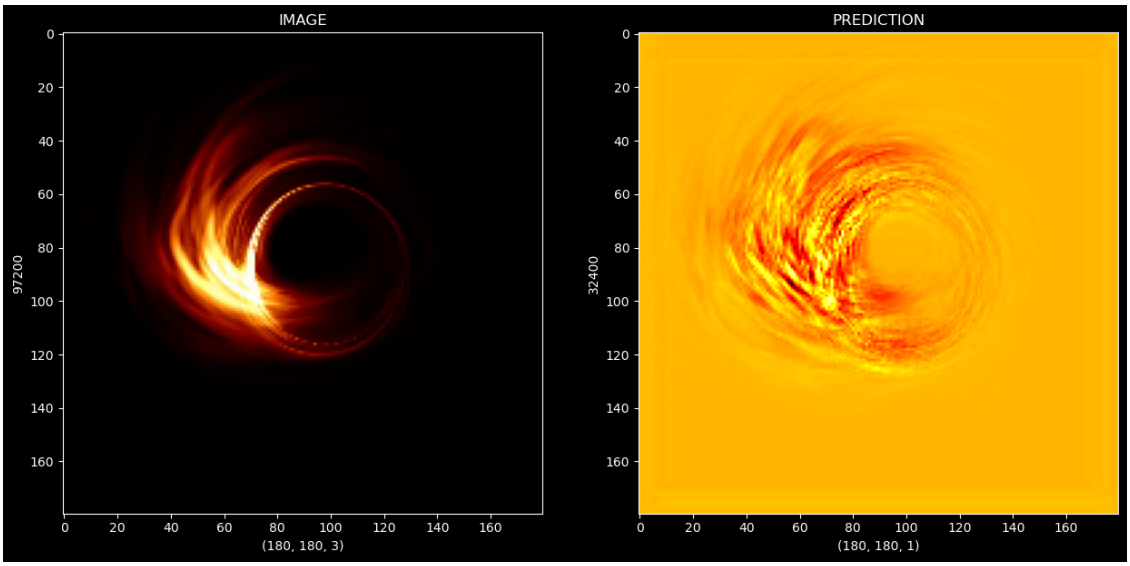
[197]:
```
Text(0.5, 1.0, 'PREDICTION')
```



[198]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))
prediction_img_number = 8

Original_Img = Train_Set[prediction_img_number]
Predict_Target = Prediction_IMG[prediction_img_number]

axis[0].imshow(Original_Img)
axis[0].set_xlabel(Original_Img.shape)
axis[0].set_ylabel(Original_Img.size)
axis[0].set_title("IMAGE")
axis[1].imshow(Predict_Target,cmap="hot")
axis[1].set_xlabel(Predict_Target.shape)
axis[1].set_ylabel(Predict_Target.size)
```

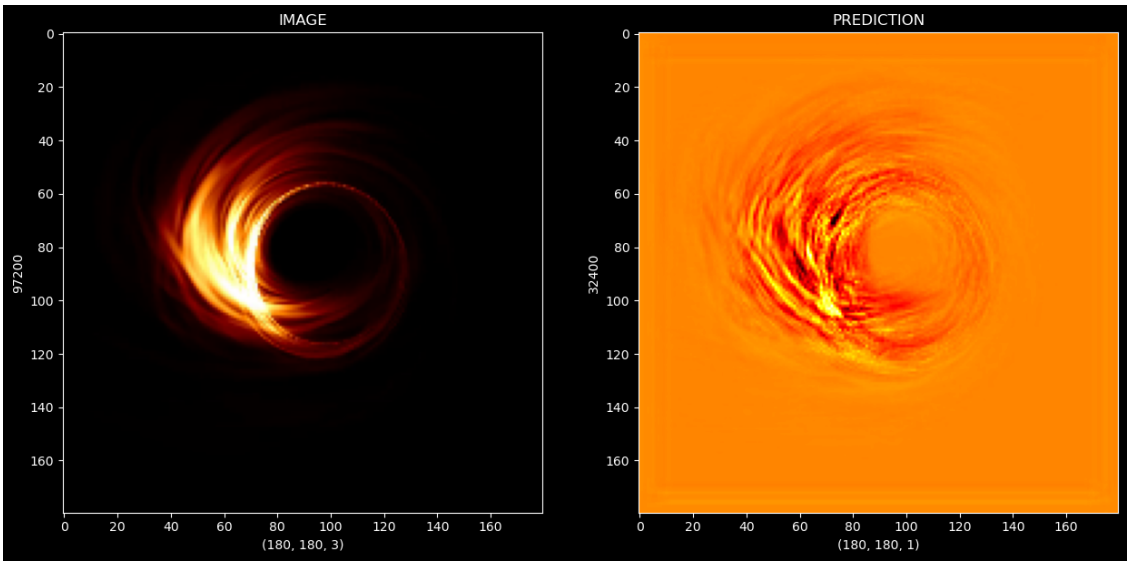
```
axis[1].set_title("PREDICTION")
```

[198]: Text(0.5, 1.0, 'PREDICTION')



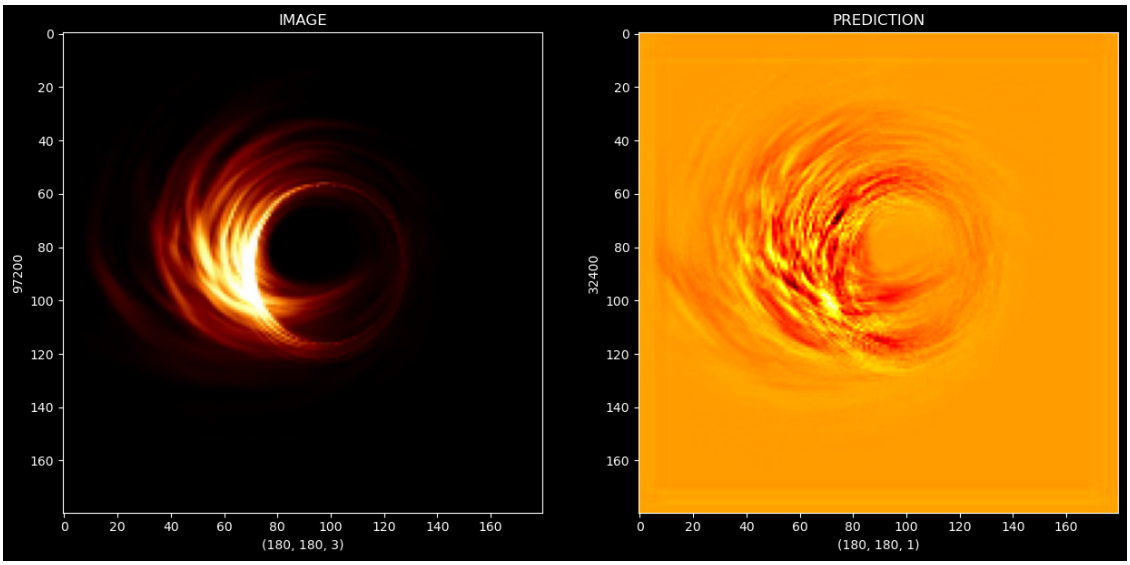
[199]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))
prediction_img_number = 250

Original_Img = Train_Set[prediction_img_number]
Predict_Target = Prediction_IMG[prediction_img_number]

axis[0].imshow(Original_Img)
axis[0].set_xlabel(Original_Img.shape)
axis[0].set_ylabel(Original_Img.size)
axis[0].set_title("IMAGE")
axis[1].imshow(Predict_Target,cmap="hot")
axis[1].set_xlabel(Predict_Target.shape)
axis[1].set_ylabel(Predict_Target.size)
axis[1].set_title("PREDICTION")
```

[199]: Text(0.5, 1.0, 'PREDICTION')



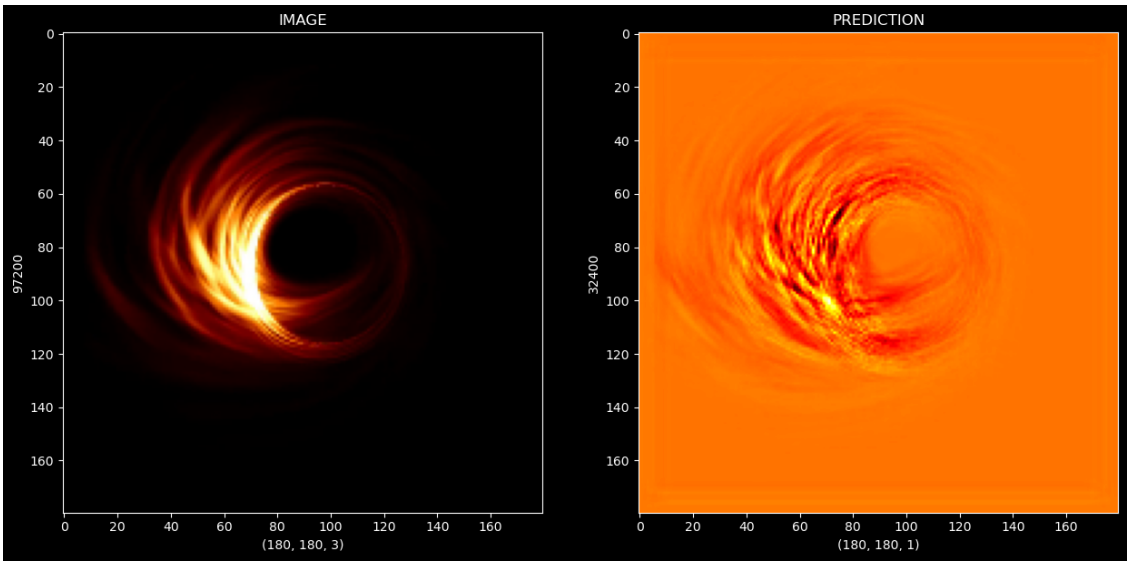
[200]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))
prediction_img_number = 444

Original_Img = Train_Set[prediction_img_number]
Predict_Target = Prediction_IMG[prediction_img_number]

axis[0].imshow(Original_Img)
axis[0].set_xlabel(Original_Img.shape)
axis[0].set_ylabel(Original_Img.size)
axis[0].set_title("IMAGE")
axis[1].imshow(Predict_Target,cmap="hot")
axis[1].set_xlabel(Predict_Target.shape)
axis[1].set_ylabel(Predict_Target.size)
axis[1].set_title("PREDICTION")
```

[200]:
```
Text(0.5, 1.0, 'PREDICTION')
```



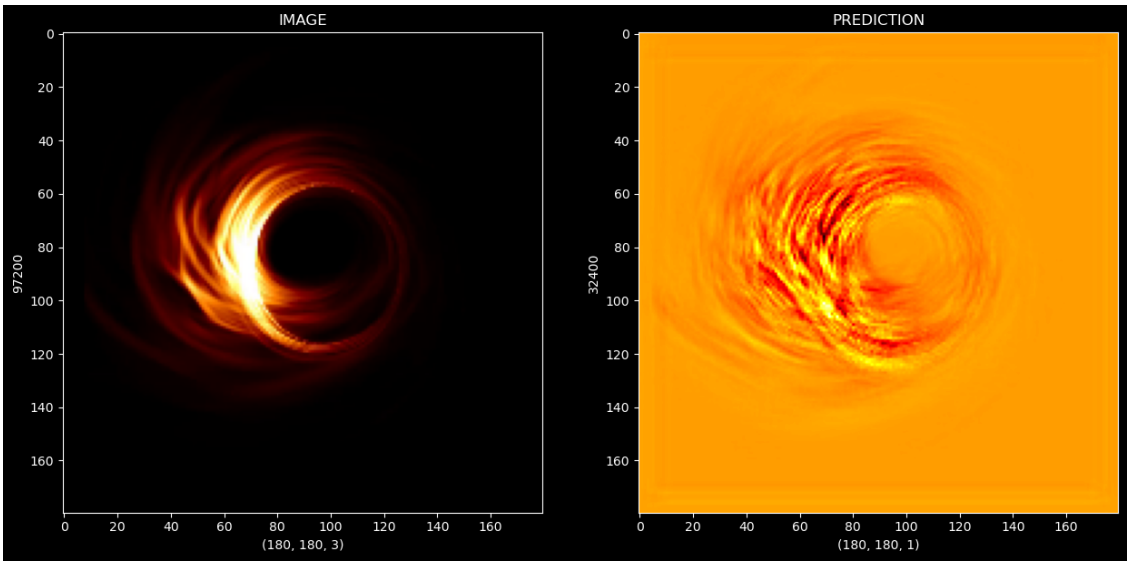
[201]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))
prediction_img_number = 450

Original_Img = Train_Set[prediction_img_number]
Predict_Target = Prediction_IMG[prediction_img_number]

axis[0].imshow(Original_Img)
axis[0].set_xlabel(Original_Img.shape)
axis[0].set_ylabel(Original_Img.size)
axis[0].set_title("IMAGE")
axis[1].imshow(Predict_Target,cmap="hot")
axis[1].set_xlabel(Predict_Target.shape)
axis[1].set_ylabel(Predict_Target.size)
axis[1].set_title("PREDICTION")
```

[201]:
```
Text(0.5, 1.0, 'PREDICTION')
```



[202]:
```
figure,axis = plt.subplots(1,2,figsize=(15,15))
prediction_img_number = 500

Original_Img = Train_Set[prediction_img_number]
Predict_Target = Prediction_IMG[prediction_img_number]

axis[0].imshow(Original_Img)
axis[0].set_xlabel(Original_Img.shape)
axis[0].set_ylabel(Original_Img.size)
axis[0].set_title("IMAGE")
axis[1].imshow(Predict_Target,cmap="hot")
axis[1].set_xlabel(Predict_Target.shape)
axis[1].set_ylabel(Predict_Target.size)
axis[1].set_title("PREDICTION")
```

[202]:
```
Text(0.5, 1.0, 'PREDICTION')
```



THANK YOU!