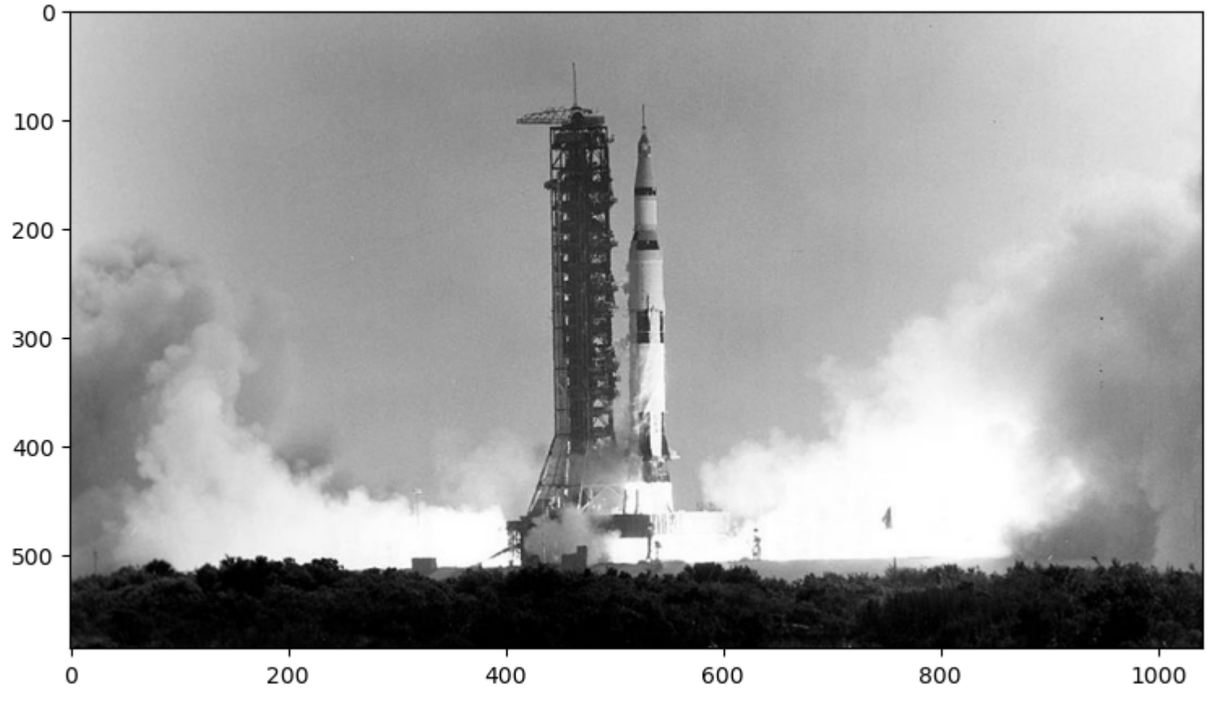
Annotating Images

In [1]:
```
#1: Draw Lines
#2: Draw Circles
#3: Draw Rectangles
#4: Add Text
```

In [2]:
```
#Import Libraries
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import cv2
from PIL import Image
matplotlib.rcParams['figure.figsize'] = (9.0, 9.0)
from IPython.display import Image
%matplotlib inline
```

In [3]:
```
#Load Image
apolloImage = cv2.imread("apollo.jpg", cv2.IMREAD_COLOR) #Or, 1 To Get Colored Image
plt.imshow(apolloImage)
```

Out[3]: <matplotlib.image.AxesImage at 0x1930184ee50>



Draw a Line

Using cv2.line() to Drawing a Line on an Image.

This Function takes 4 required arguments

In [4]:
```
#1: img: Image on which we'll Draw a line
#2: pt1: First Point (x, y) Location of the Line Segment
#3: pt2: Second Point of the Line Segment (End Point)
#4: color: Color of the Line which we'll be Drawn
```

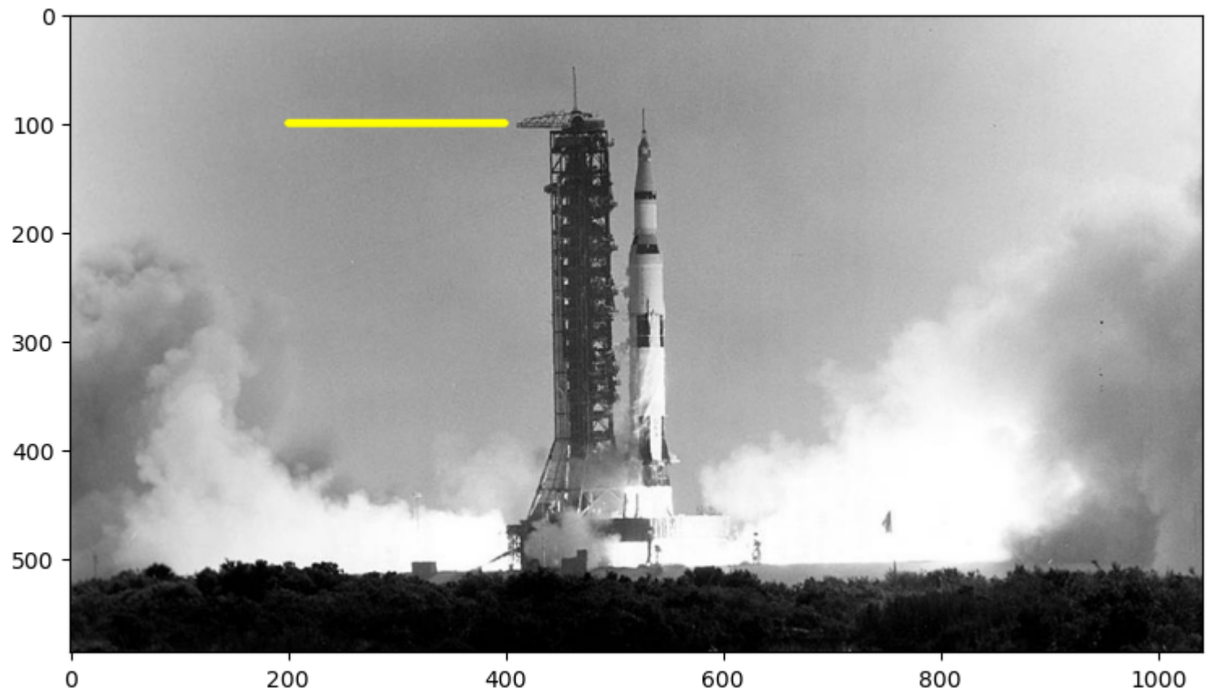
Other Optional Argument that are Important for us to Know

In [5]:
```
#1: thickness: Specifying the Line Thickness (int)
#2: lineType: Type of the Line, Default Value is 8 which stands for an 8 Connected Line.
```

Usually, cv2.LINE_AA (antialiased or smooth line) is used for the Line Type

In [6]:
```
#Make a new Image Which we're gonna draw a Line on
apolloWithLine = apolloImage.copy()

#The Line Starts From (200, 100) and ends at (400, 100). Where (200=Left, 100=Top) and (400=Left, 100=Top)
#The Color of the Line is Yellow (Recall that OpenCV uses BGR Format)
#Thickness of The Line is 5px
#Line Type is cv2.LINE_AA

#Format --> (ImageName, StartPoint, EndPoint, Color, Thickness, LineType)

cv2.line(apolloWithLine,
         (200, 100),
         (400, 100),
         (0, 255, 255),
         thickness = 5,
         lineType = cv2.LINE_AA)
plt.imshow(cv2.cvtColor(apolloWithLine, cv2.COLOR_BGR2RGB)) #Or, Use img[:,:, ::-1] to Convert Default BGR2RGB
```

Out[6]: <matplotlib.image.AxesImage at 0x19301976a90>



Draw a Circle

Using cv2.circle() to Drawing a Line on an Image.

This Function takes 4 required arguments

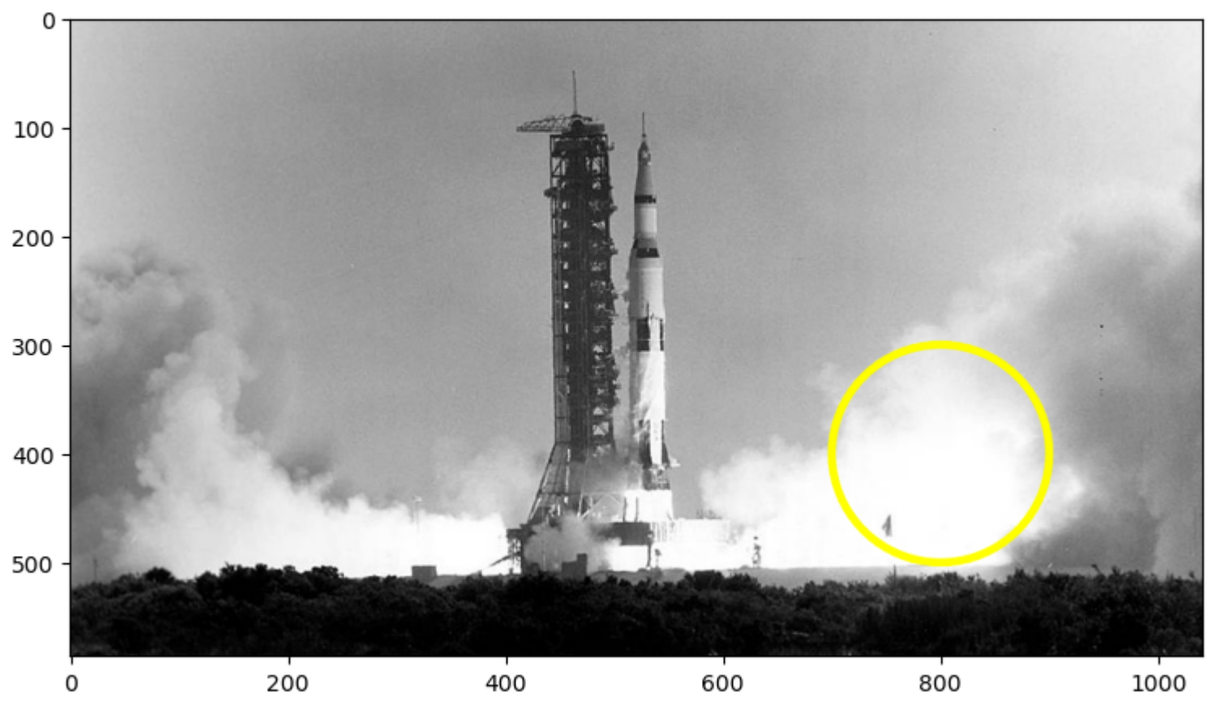
In [7]:
```
#1: img: Image on which we'll Draw a Circle
#2: center: Center of The Circle
#3: radius: Radius of The Circle
#4: color: Color of the Line which we'll be Drawn
```

Other Optional Argument that are Important for us to Know

In [8]:
```
#1: thickness: Specifying the Thickness of the Circle --> if Positive.
# If, Negative Value is Given, it'll Result in a Filled Circle.

#2: lineType: Type of the Circle Boundary, This is same as lineType argument in cv2.line
```

In [9]:
```
#Make a new Image Which we're gonna draw a Circle on
apolloWithCircle = apolloImage.copy()


#Format --> (imageName, Center, Radious, Color, Thickness, LineType)

cv2.circle(apolloWithCircle,
           (800, 400),
           100,
           (0, 255, 255),
           thickness = 5,
           lineType = cv2.LINE_AA)
plt.imshow(cv2.cvtColor(apolloWithCircle, cv2.COLOR_BGR2RGB)) #Or, Use img[:,:, ::-1] to Convert Default BGR2RGB
```

Out[9]: <matplotlib.image.AxesImage at 0x193020679d0>



Draw a Rectangle

Using cv2.rectangle() to Drawing a rectangle on an Image.

This Function takes 4 required arguments

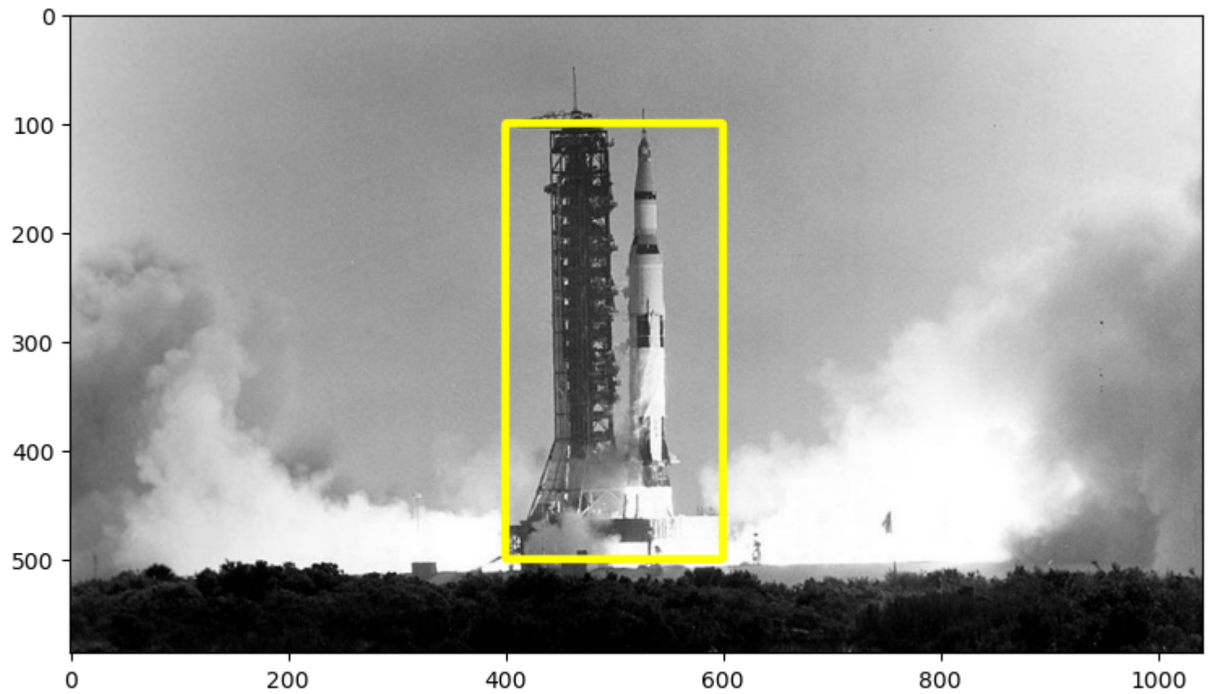
In [10]:
```
#1: img: Image on which we'll Draw a Rectangle
#2: pt1: Vertex of The Rectangle. Usuallly use top-left vertex.
#3: pt2: Vertex of The Rectangle, Opposite to pt1. Usuallly use bottom-right vertex.
#4: color: Color of the Line which we'll be Drawn
```

Other Optional Argument that are Important for us to Know

In [11]:
```
#1: thickness: Specifying the Thickness of the Rectangle --> if Positive.
# If, Negative Value is Given, it'll Result in a Filled Rectangle.

#2: lineType: Type of the Rectangle Boundary, This is same as lineType argument in cv2.line
```

In [12]:
```
#Make a new Image Which we're gonna draw a Rectangle on
apolloWithRectangle = apolloImage.copy()


#Format --> (ImageName, Left_Top, Right_Bottom, Color, Thickness, LineType)

cv2.rectangle(apolloWithRectangle,
              (400, 100),
              (600, 500),
              (0, 255, 255),
              thickness = 5,
              lineType = cv2.LINE_AA)
plt.imshow(cv2.cvtColor(apolloWithRectangle, cv2.COLOR_BGR2RGB)) #Or, Use img[:,:, ::-1] to Convert Default BGR2RGB
```

Out[12]: <matplotlib.image.AxesImage at 0x193023a61f0>



Adding Text

Using cv2.putText() to Drawing a rectangle on an Image.

This Function takes 6 required arguments

In [13]:
```
#1: img: Image on which we'll Add Text on
#2: text: Text to be Written as String
#3: org: Bottom-Left Corner of the Text String in the Image
#4: fontFace: Font Type
#5: fontScale: Font Scale Factor that is Multiplied by the font-specific base Size
#6: color: Color of Font
```

Other Optional Argument that are Important for us to Know

In [14]:
```
#1: thickness: Specifying the Thickness For the Text , Default Value is 1.
#2: lineType: Type of the Line, Default Value is 8 which stands for an 8 Connected Line. Same as cv2.LINE_AA
```

In [15]:
```
#Make a new Image Which we're gonna Add a Text on.
apolloWithText = apolloImage.copy()


#Format --> (ImageName, Text, Bottom_Left-Corner, fontFace, fontScale, Color, Thickness, LineType)

cv2.putText(apolloWithText,
            "Apollo 11 Saturn V Launch, 16July 1969",
            (200, 550),
            cv2.FONT_HERSHEY_SIMPLEX,
            1,
            (0, 255, 255),
            5,
            cv2.LINE_AA)

plt.imshow(cv2.cvtColor(apolloWithText, cv2.COLOR_BGR2RGB)) #Or, Use img[:,:, ::-1] to Convert Default BGR2RGB
```

Out[15]: <matplotlib.image.AxesImage at 0x1930291eee0>