Object segmentation using Colour Thresholding and solid colour background replacement

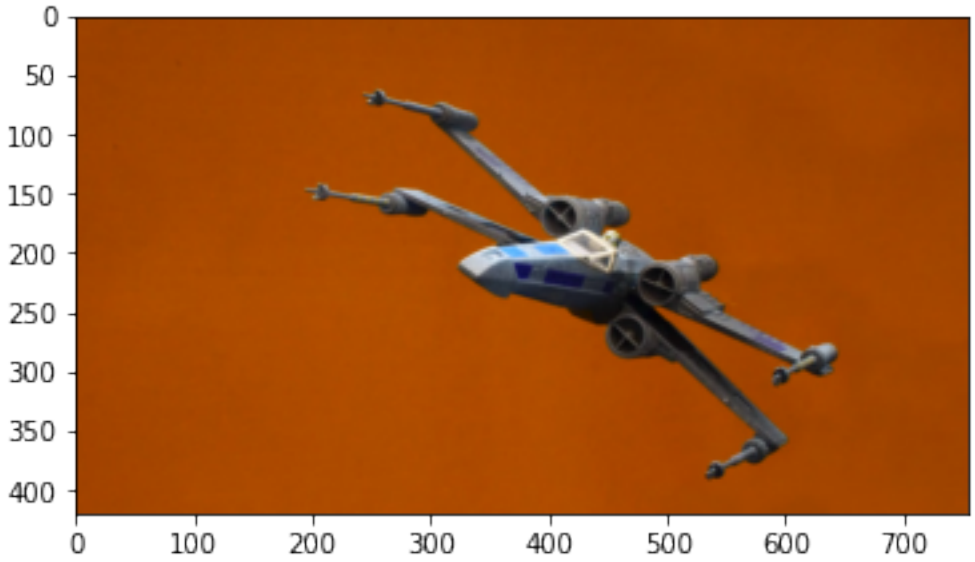
July 23, 2021

[ ]:
```
"""  https://github.com/Shivam1795  """
```

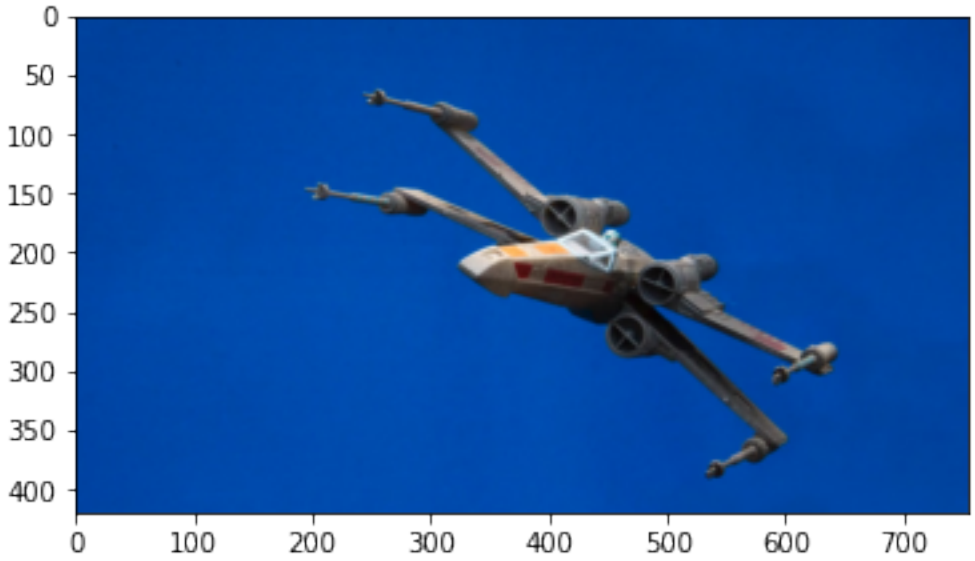
[1]:
```
## Import all the required libraries !!
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
## Read image !!
image = cv2.imread('images/spaceship_blueScreen.jpg')

## Display the dtype and dimensions of the image array !!
print('image <dtype> and (dimensions):', type(image), image.shape)

## By default cv2 uses BGR format while other libraries use RGB, so the image here will look different (No blue screen) !!
plt.imshow(image);
```

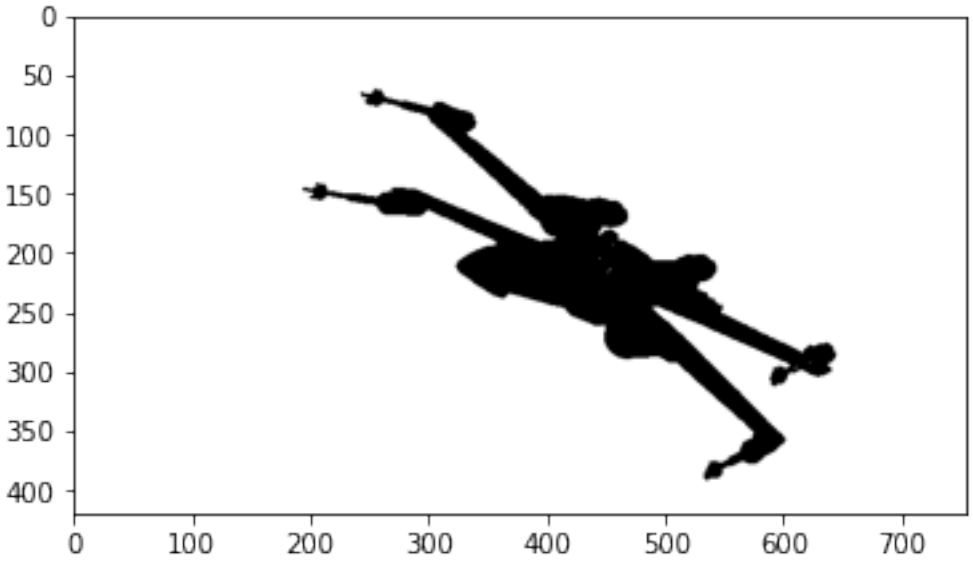
```
image <dtype> and (dimensions): <class 'numpy.ndarray'> (420, 755, 3)
```



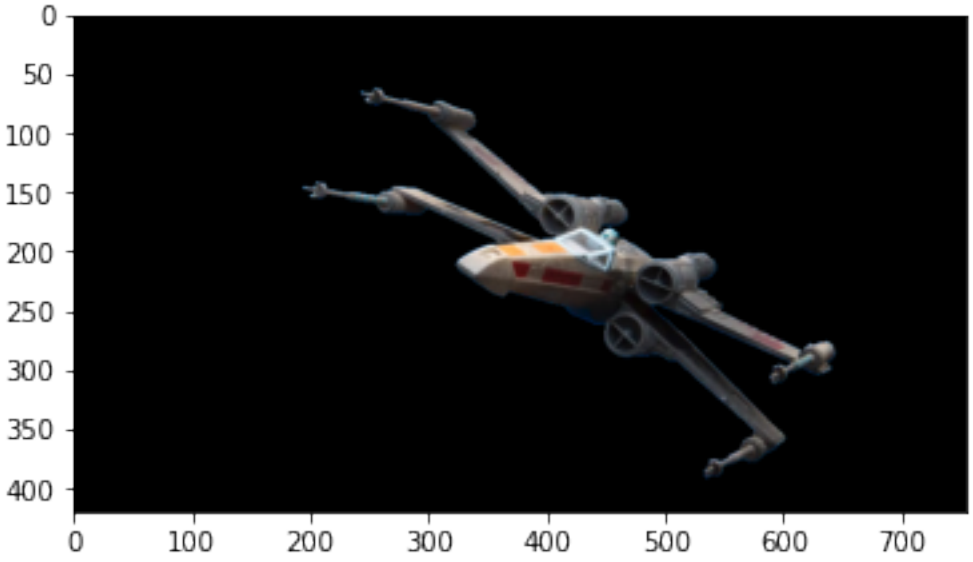
[3]:
```
## convert a copy of the original BGR image to RGB  !!
img_copy = cv2.cvtColor(np.copy(image), cv2.COLOR_BGR2RGB)

## This image will look normal (With a blue screen)  !!
plt.imshow(img_copy);
```



[4]:
```
## Define boundaries to select blue background in RGB image (lower and upper␣
 ↪limits) !!

## Just play with these values to find correct upper and lower threshold values␣
 ↪!!
lower_lim = np.array([0, 50, 100])
upper_lim = np.array([80, 100, 255])

## Uncomment these lines for hp.jpg image
#lower_lim = np.array([0, 0, 85])
#upper_lim = np.array([100, 100, 255])

## Create a mask !!
mask = cv2.inRange(img_copy, lower_lim, upper_lim)

## Display mask !!
plt.imshow(mask, cmap='gray');
```

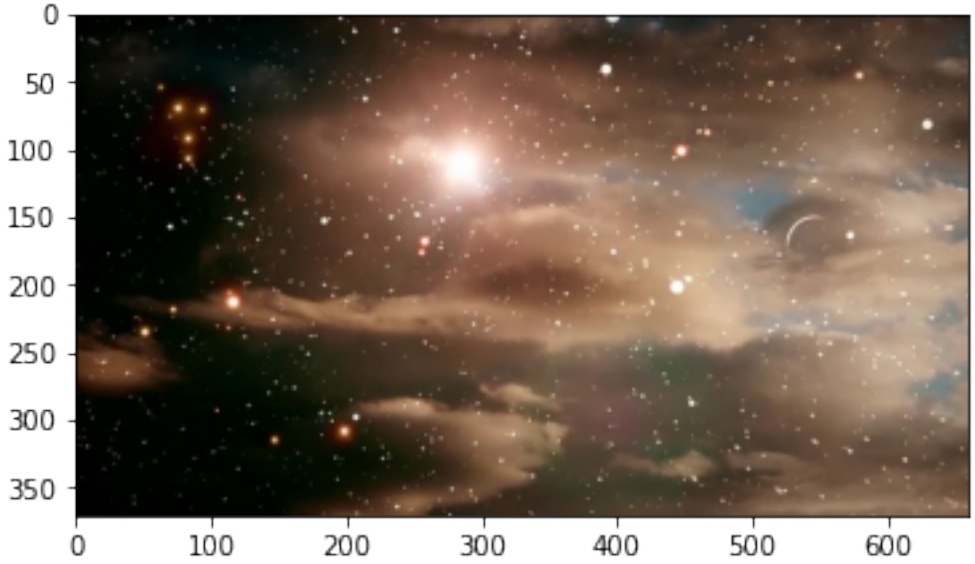


[5]:
```
## Generate a masked image using new copy !!
masked_img = np.copy(img_copy)

## In a copy of the original image replace the pixel values with vector [0,0,0]
 ↪at those locations, wherein the corresponding masked image pixels are
 ↪non-zero !!
masked_img[mask != 0] = [0, 0, 0]

## Display masked image
plt.imshow(masked_img);
```



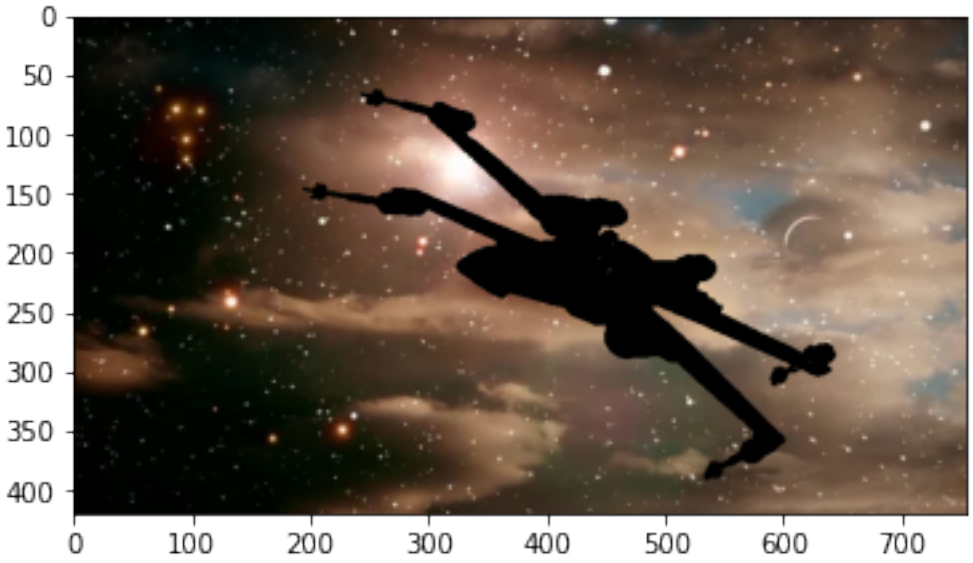
[6]:
```
## Read background image and convert to RGB from BGR !!
background = cv2.cvtColor(cv2.imread('images/space.jpg'), cv2.COLOR_BGR2RGB)

## Check the shape of the background !!
print('Original Shape :',background.shape)

## Display original background !!
plt.imshow(background);
```

```
Original Shape : (371, 660, 3)
```

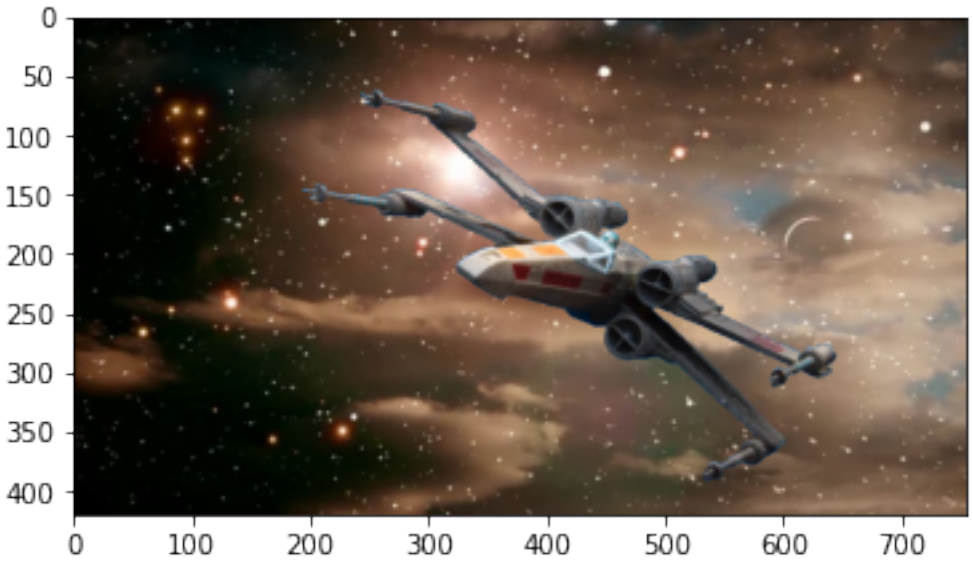


[7]:
```
## Resize background image to our original image size !!
resized_background = cv2.resize(background, (image.shape[1], image.shape[0]),␣
 ↪interpolation = cv2.INTER_AREA)

## Check the shape of the background !!
print('Shape after resizing :',resized_background.shape)

## Crop background to fit our masked image !!
resized_background[mask == 0] = [0, 0, 0]

## Display resized_background !!
plt.imshow(resized_background);
```

```
Shape after resizing : (420, 755, 3)
```



[8]:
```
## Add two images to generate the final image !!
final_img = resized_background + masked_img

## Display final image !!
plt.imshow(final_img);
```



[9]:
```
## Thanks !!
```