Final Dataset Generation Using Annotations

January 30, 2023

1 Foreground Separation using COCO API

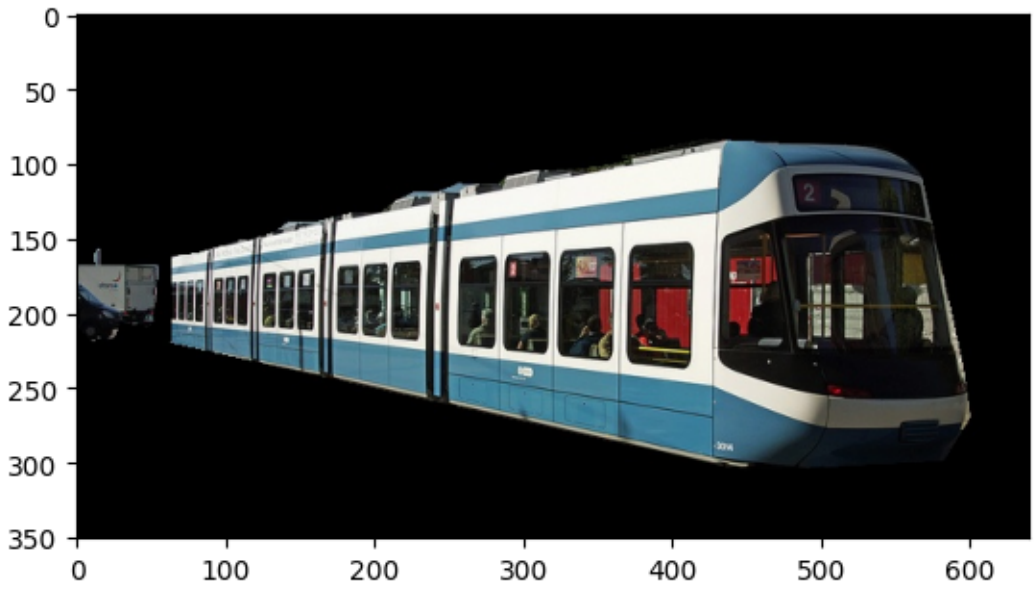
[ ]:
```
# Foreground separation using annotations - Working

import json
from pycocotools.coco import COCO
import cv2
import numpy as np
import os

# Load the COCO annotations
ann_file = "C:/Users/mukun/Desktop/Workshop Image Dataset Generation/DATASET/
 ↪annotations_trainval2017/annotations/instances_train2017.json"
coco = COCO(ann_file)

# Load the image
img_dir = "C:/Users/mukun/Desktop/Workshop Image Dataset Generation/train2017/
 ↪train2017"

# Get all image IDs in the dataset
img_ids = coco.imgs.keys()

# Iterate over the image IDs
for img_id in img_ids:
    # Load the image
    img = cv2.imread(os.path.join(img_dir, '%012d.jpg' % img_id))

    # Get the object instances for the image
    ann_ids = coco.getAnnIds(imgIds=img_id)
    anns = coco.loadAnns(ann_ids)

    # Create a binary mask for the background
    mask = np.ones(img.shape[:2], np.uint8)
    for ann in anns:
        mask[coco.annToMask(ann)==1] = 0

    # Subtract the image with the binary mask to extract the background
```

```
    img = cv2.subtract(img, img * mask[:,:,np.newaxis])

    # Save the output image
    cv2.imwrite(os.path.join("C:/Users/mukun/Desktop/Workshop Image Dataset␣
 ↪Generation/train fg separation", '%012d.jpg' % img_id), img)
```

1.1 Examples

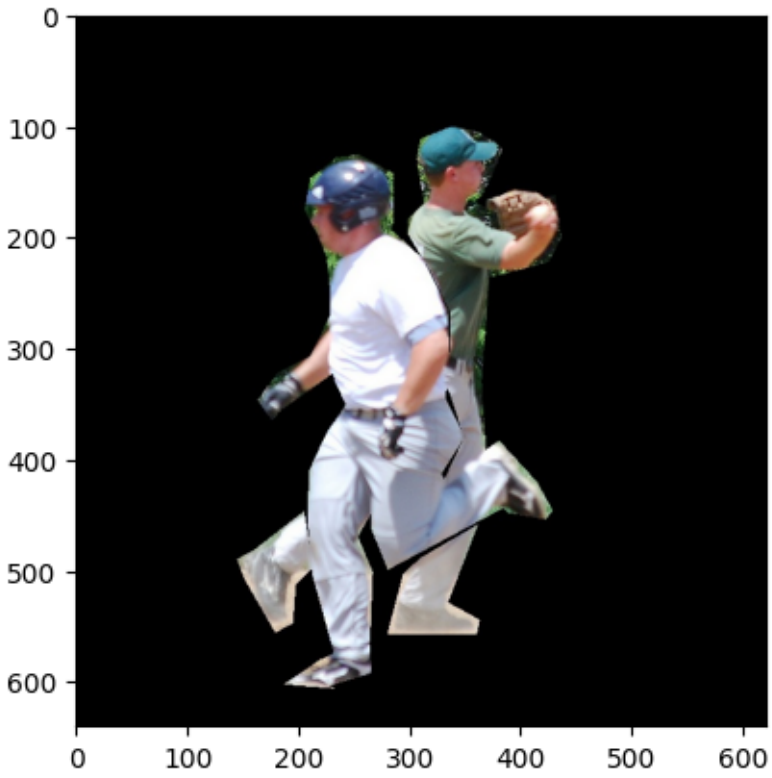
[1]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val fg␣
 ↪separation/000000006040.jpg')
plt.imshow(original_img)
```

[1]:
```
<matplotlib.image.AxesImage at 0x7f9c12980ac0>
```



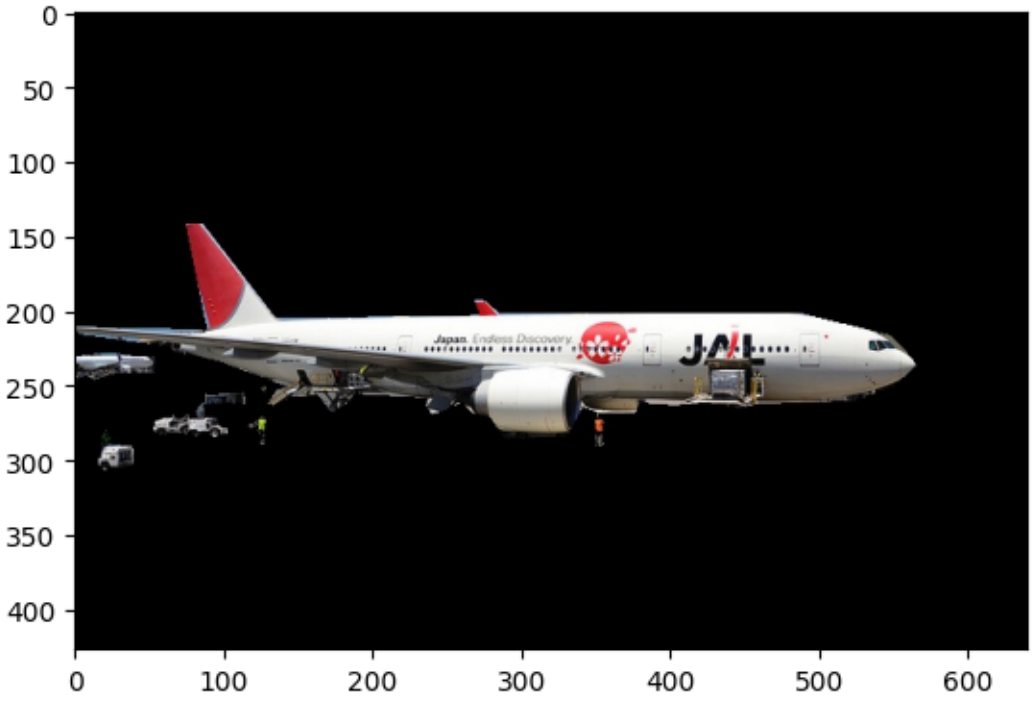
[2]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val fg␣
 ↪separation/000000000872.jpg')
plt.imshow(original_img)
```

[2]:
```
<matplotlib.image.AxesImage at 0x7f9c107f8850>
```



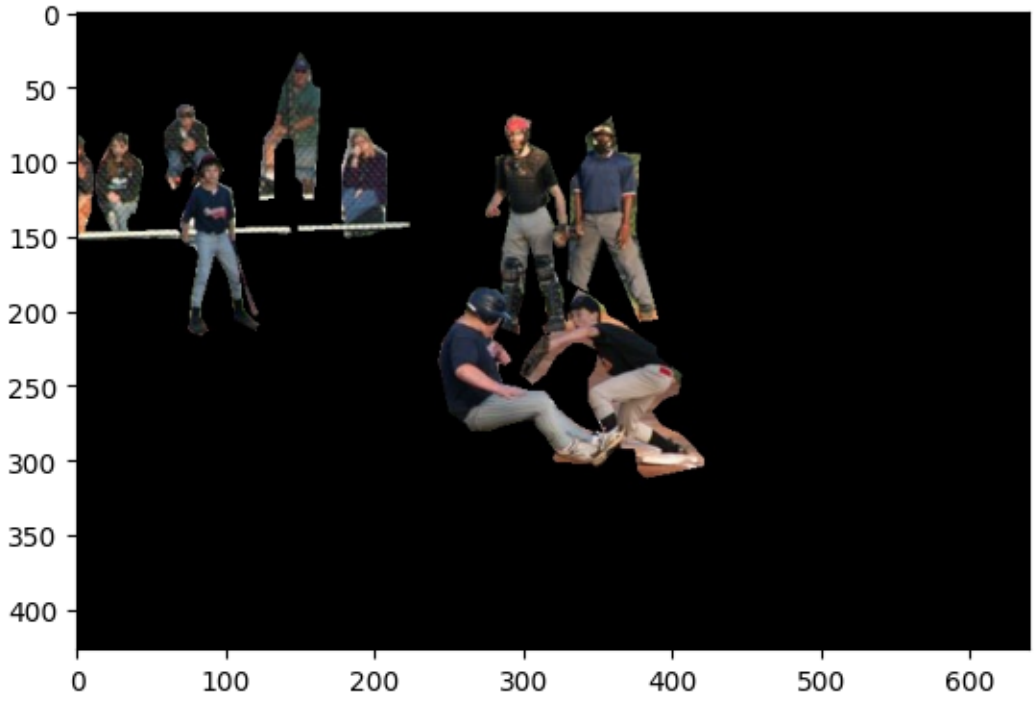
[3]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val fg␣
 ↪separation/000000013348.jpg')
plt.imshow(original_img)
```

[3]: <matplotlib.image.AxesImage at 0x7f9c10862410>



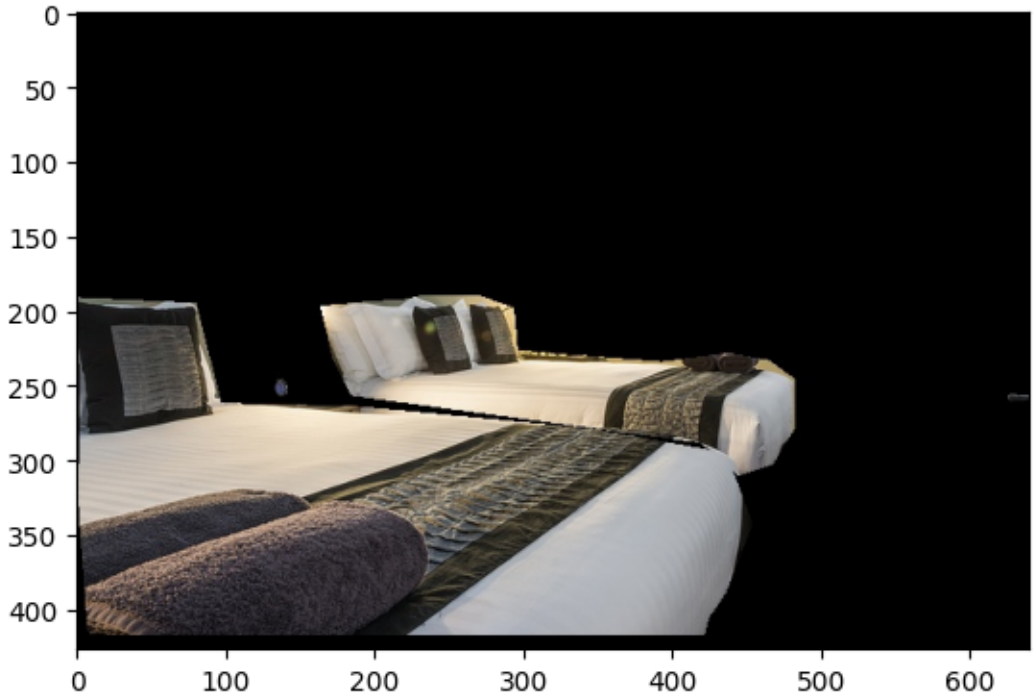
[4]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val fg␣
 ↪separation/000000018491.jpg')
plt.imshow(original_img)
```

[4]:
```
<matplotlib.image.AxesImage at 0x7f9c0eeed6c0>
```



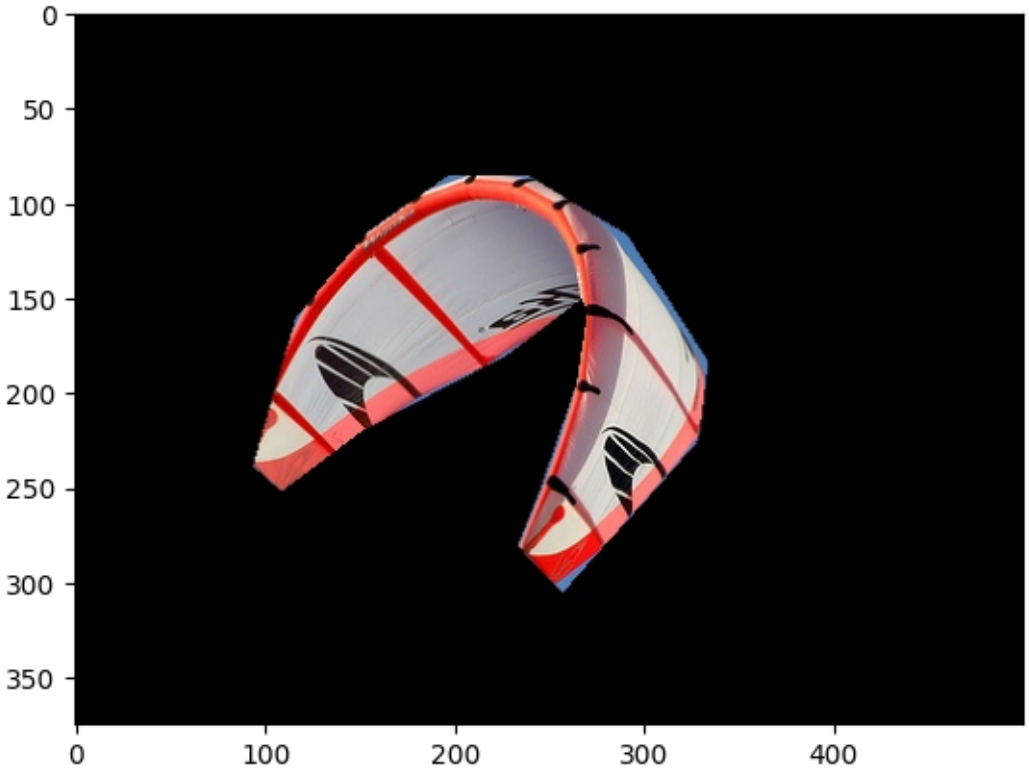
[5]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val fg␣
 ↪separation/000000007795.jpg')
plt.imshow(original_img)
```

[5]:
```
<matplotlib.image.AxesImage at 0x7f9c0ef84f10>
```



[6]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val fg␣
 ↪separation/000000007784.jpg')
plt.imshow(original_img)
```

[6]: <matplotlib.image.AxesImage at 0x7f9c0ee18400>



2 Background Separation Using COCO API

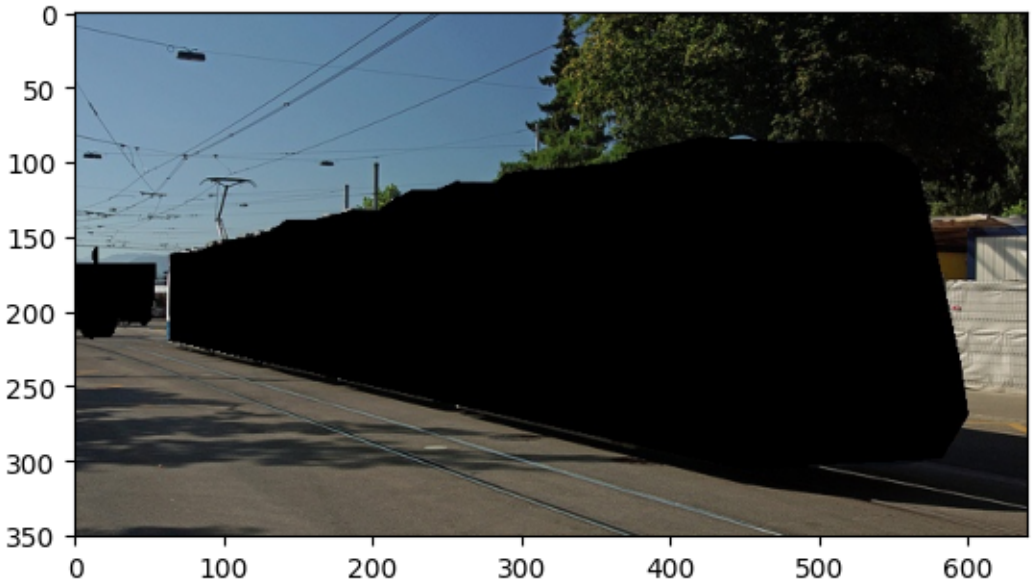
[ ]:
```
# Background separation using annotations - Working

import json
from pycocotools.coco import COCO
import cv2
import numpy as np
import os

# Load the COCO annotations
ann_file = "C:/Users/mukun/Desktop/Workshop Image Dataset Generation/DATASET/
 ↪annotations_trainval2017/annotations/instances_train2017.json"
coco = COCO(ann_file)

# Load the image
img_dir = "C:/Users/mukun/Desktop/Workshop Image Dataset Generation/train2017/
 ↪train2017"

# Get all image IDs in the dataset
img_ids = coco.imgs.keys()
```

```
# Iterate over the image IDs
for img_id in img_ids:
    # Load the image
    img = cv2.imread(os.path.join(img_dir, '%012d.jpg' % img_id))

    # Get the object instances for the image
    ann_ids = coco.getAnnIds(imgIds=img_id)
    anns = coco.loadAnns(ann_ids)

    # Create a binary mask for the foreground
    mask = np.zeros(img.shape[:2], np.uint8)
    for ann in anns:
        mask[coco.annToMask(ann)==1] = 1

    # Subtract the binary mask from the image to extract the background
    img = cv2.subtract(img, img * mask[:,:,np.newaxis])

    # Save the output image
    cv2.imwrite(os.path.join("C:/Users/mukun/Desktop/Workshop Image Dataset␣
 ↪Generation/train bg separation", '%012d.jpg' % img_id), img)
```

2.1 Examples

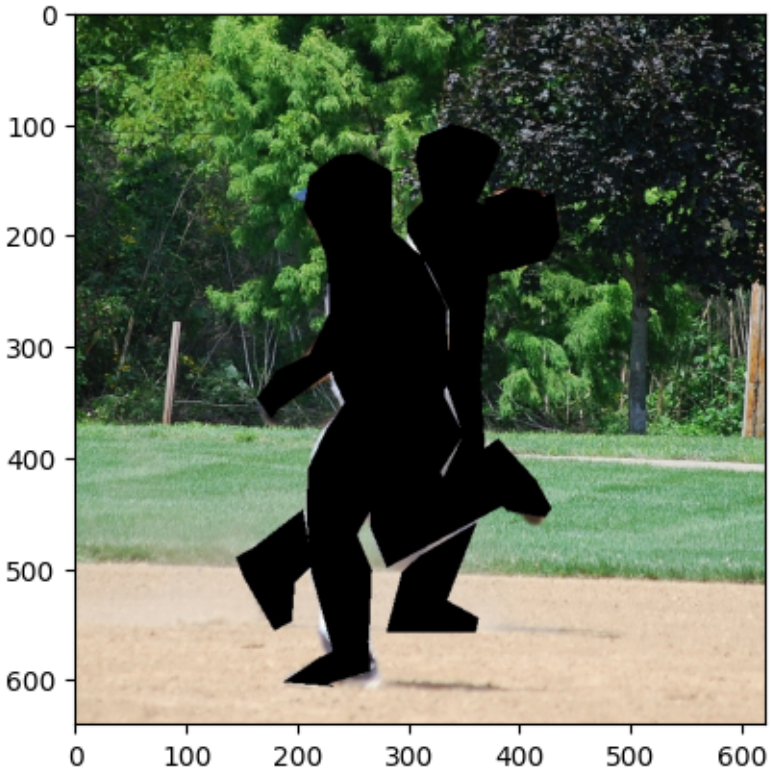
[8]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val bg␣
 ↪separation/000000006040.jpg')
plt.imshow(original_img)
```

[8]:
```
<matplotlib.image.AxesImage at 0x7f9c0ece7370>
```



[9]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val bg␣
 ↪separation/000000000872.jpg')
plt.imshow(original_img)
```

[9]: <matplotlib.image.AxesImage at 0x7f9c0ed7de10>



[10]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val bg␣
 ↪separation/000000013348.jpg')
plt.imshow(original_img)
```

[10]: <matplotlib.image.AxesImage at 0x7f9c0ed38a90>

[11]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val bg␣
 ↪separation/000000018491.jpg')
plt.imshow(original_img)
```

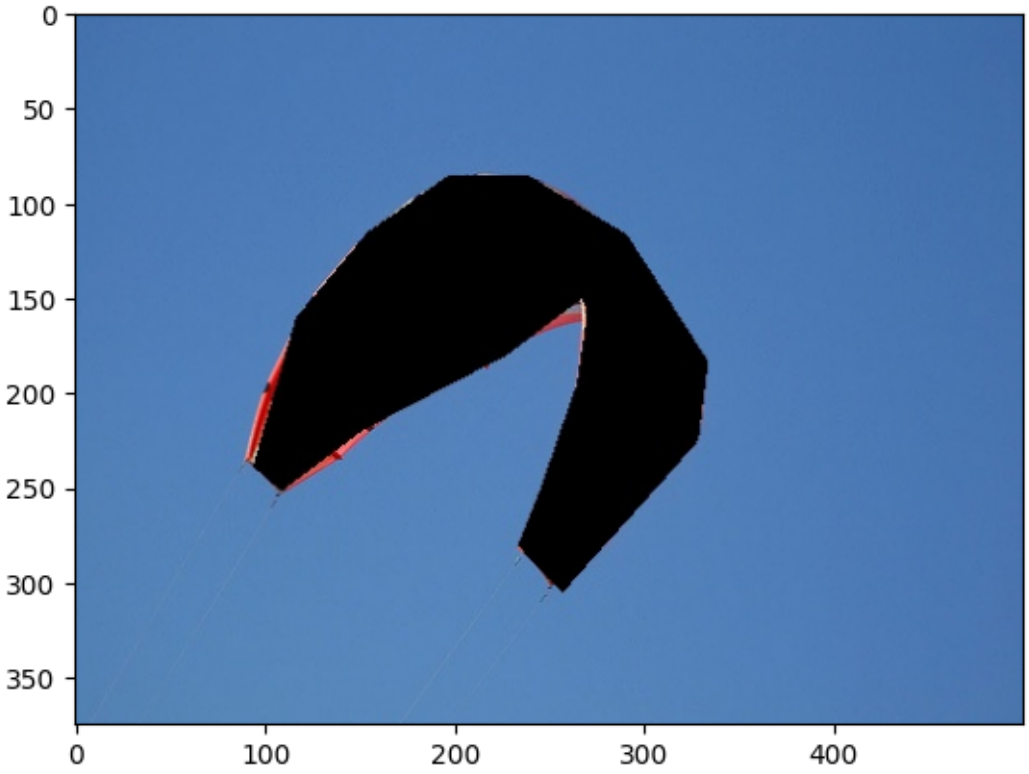
[11]: <matplotlib.image.AxesImage at 0x7f9c0ec5ec20>

[12]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val bg␣
 ↪separation/000000007795.jpg')
plt.imshow(original_img)
```

[12]: <matplotlib.image.AxesImage at 0x7f9c0eaea410>

[13]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val bg␣
 ↪separation/000000007784.jpg')
plt.imshow(original_img)
```

[13]:
```
<matplotlib.image.AxesImage at 0x7f9c0eb79b40>
```



3 Foreground Blurring using Randomized Motion Blur Kernel

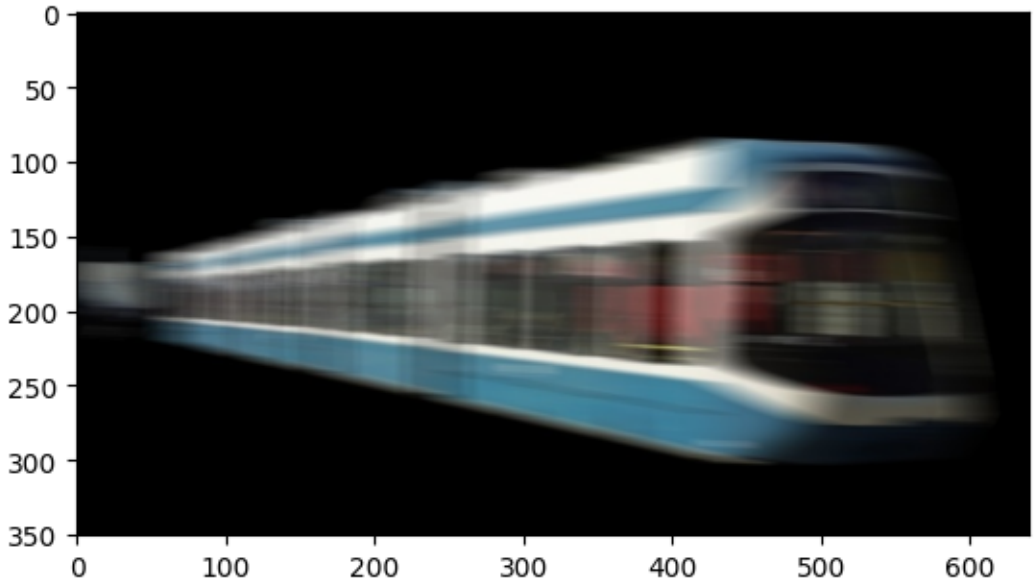
[ ]:
```
# Using the motion blur kernel from the MTech Thesis

import numpy as np
import cv2
# from google.colab.patches import cv2_imshow
import os
import pandas as pd
import re
import matplotlib.pyplot as plt
# import random
from random import randrange,random,choice
import time

def blurkernel(i): # takes parameters for blurring as input as an array 'i'.
  ker_size=2*i[0]+1;  # The first parameter is (kernel_size-1)/2.
  blur=np.zeros((ker_size,ker_size))
  # blur[i[0],i[0]:ker_size] = (1/(i[0]+1))*np.ones(i[0]+1)

  #implementing non-symmetric blurring for realistic blurring.
```

```
  blur[i[0],:]= (1/(ker_size))*np.ones(ker_size)

  # blur[i[0],0:ker_size+1]=(1/ker_size)*np.ones(ker_size) #symmetric blurring

  if i[1]!=0:      # the second parameter is blur angle.
    (h, w) = blur.shape
    (cX, cY) = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D((cX, cY), i[1], 1.0)
    rotated = cv2.warpAffine(blur, M, (w, h))
  else:
    rotated=blur
  rotated=rotated/rotated.sum()
  return rotated

def motionblur(image,i):
  ker_size=2*i[0]+1;
  blur=np.zeros((ker_size,ker_size))
  # blur[i[0],i[0]:ker_size] = (1/(i[0]+1))*np.ones(i[0]+1)
  blur[i[0],:]= (1/(ker_size))*np.ones(ker_size)
  # blur[i[0],0:ker_size+1]=(1/ker_size)*np.ones(ker_size) #symmetric blurring
  if i[1]!=0:
    (h, w) = blur.shape
    (cX, cY) = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D((cX, cY), i[1], 1.0)
    rotated = cv2.warpAffine(blur, M, (w, h))
  else:
    rotated=blur
  rotated=rotated/rotated.sum()
  # plt.figure()
  # plt.imshow(rotated,cmap='gray')
  # plt.title("blur applied")
  output = cv2.filter2D(image,-1,rotated,borderType=cv2.BORDER_CONSTANT  )
  return output

  # Iterating through the entire validation dataset foreground images to induce␣
 ↪random motion blur

img_dir = "C:/Users/mukun/Desktop/Workshop Image Dataset Generation/GOOD␣
 ↪RESULTS/TRAIN/train fg separation"
blurred_img_dir = "C:/Users/mukun/Desktop/Workshop Image Dataset Generation/
 ↪train fg blurred"
for filename in os.listdir(img_dir):
    img = cv2.imread(os.path.join(img_dir, filename))
    ker_size = randrange(2,11)*2+1
    angle = choice([0,randrange(1,360)])
    output_img = motionblur(img,[ker_size,angle])
    cv2.imwrite(os.path.join(blurred_img_dir, filename), output_img)
```

3.1 Examples

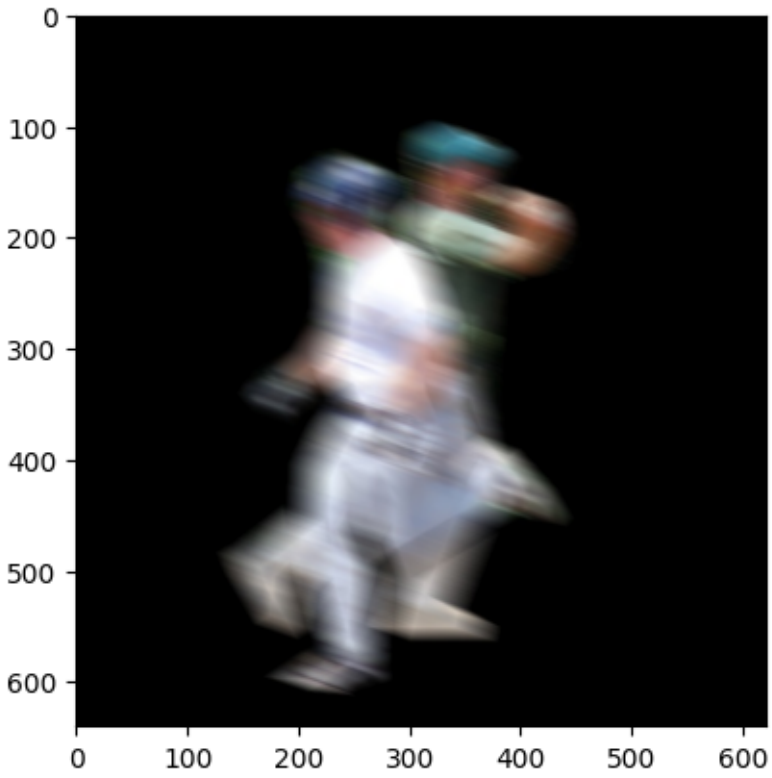
[14]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val fg␣
 ↪blurred/000000006040.jpg')
plt.imshow(original_img)
```

[14]:
```
<matplotlib.image.AxesImage at 0x7f9c0e9eb940>
```



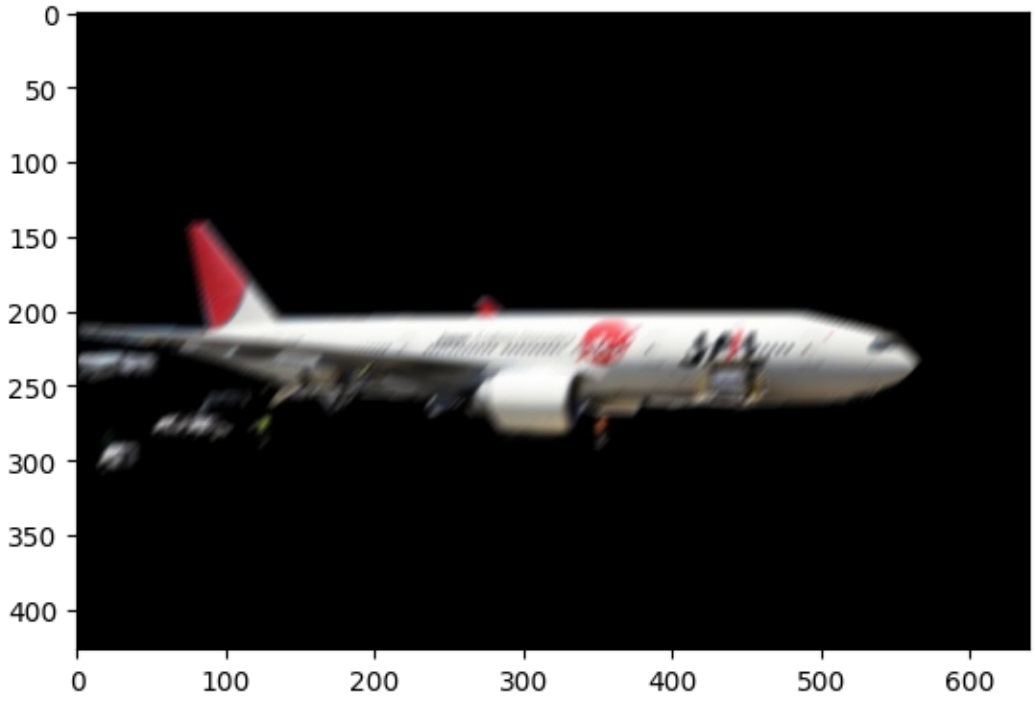
[15]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val fg␣
 ↪blurred/000000000872.jpg')
plt.imshow(original_img)
```

[15]:
```
<matplotlib.image.AxesImage at 0x7f9c0ea7a0b0>
```



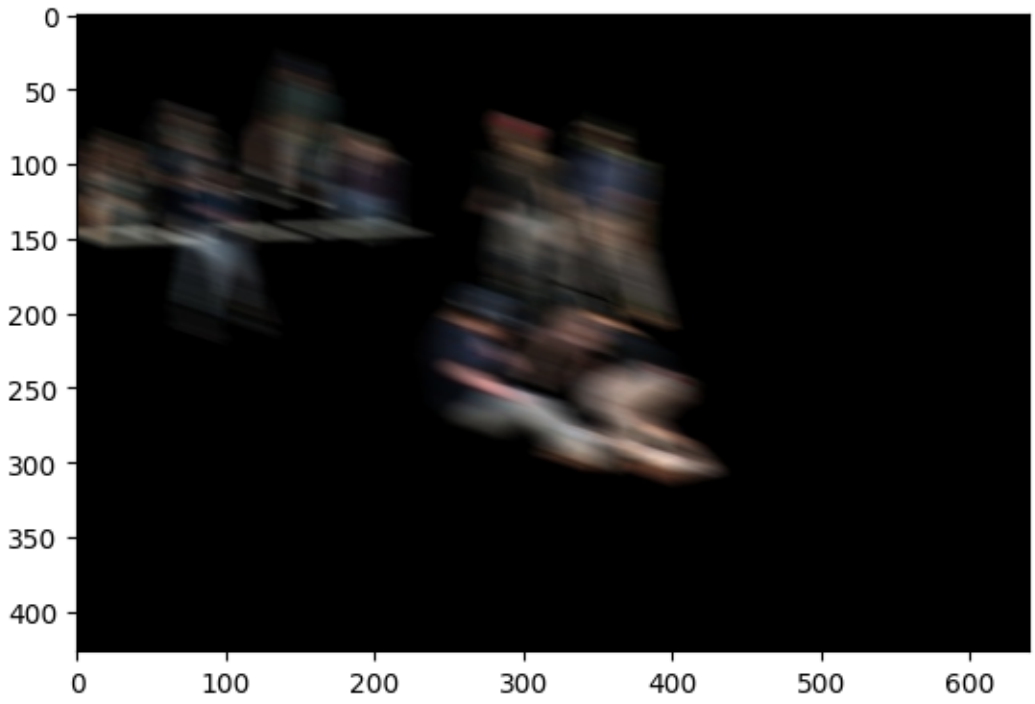
[16]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val fg␣
 ↪blurred/000000013348.jpg')
plt.imshow(original_img)
```

[16]: <matplotlib.image.AxesImage at 0x7f9c0e8e7a90>



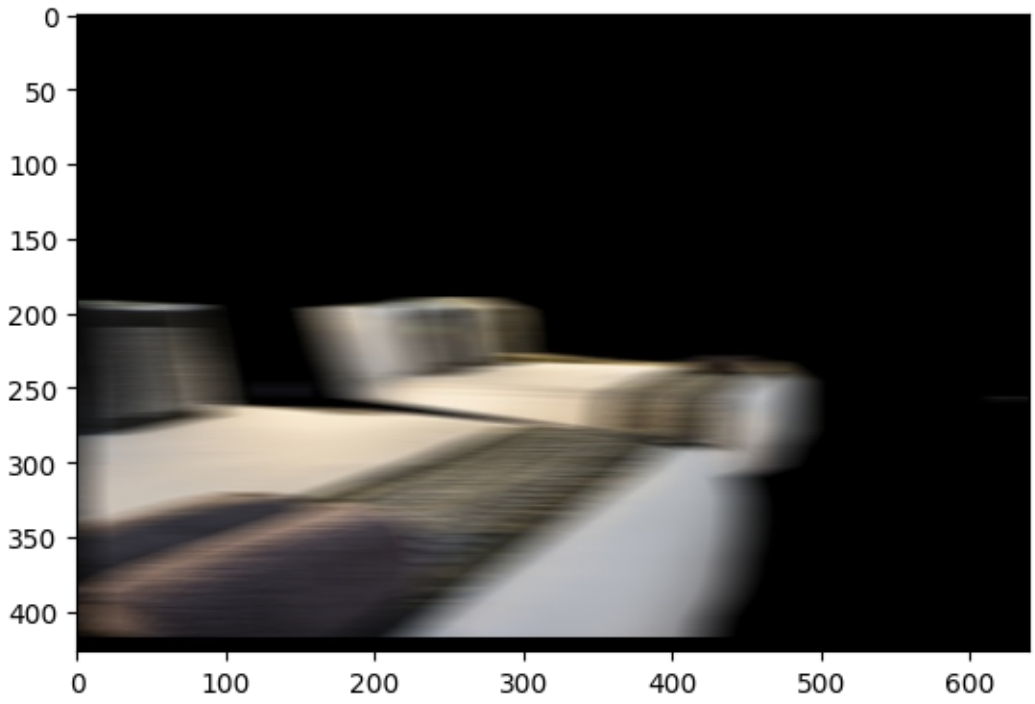
[17]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val fg␣
↪blurred/000000018491.jpg')
plt.imshow(original_img)
```

[17]: <matplotlib.image.AxesImage at 0x7f9c0e96b430>



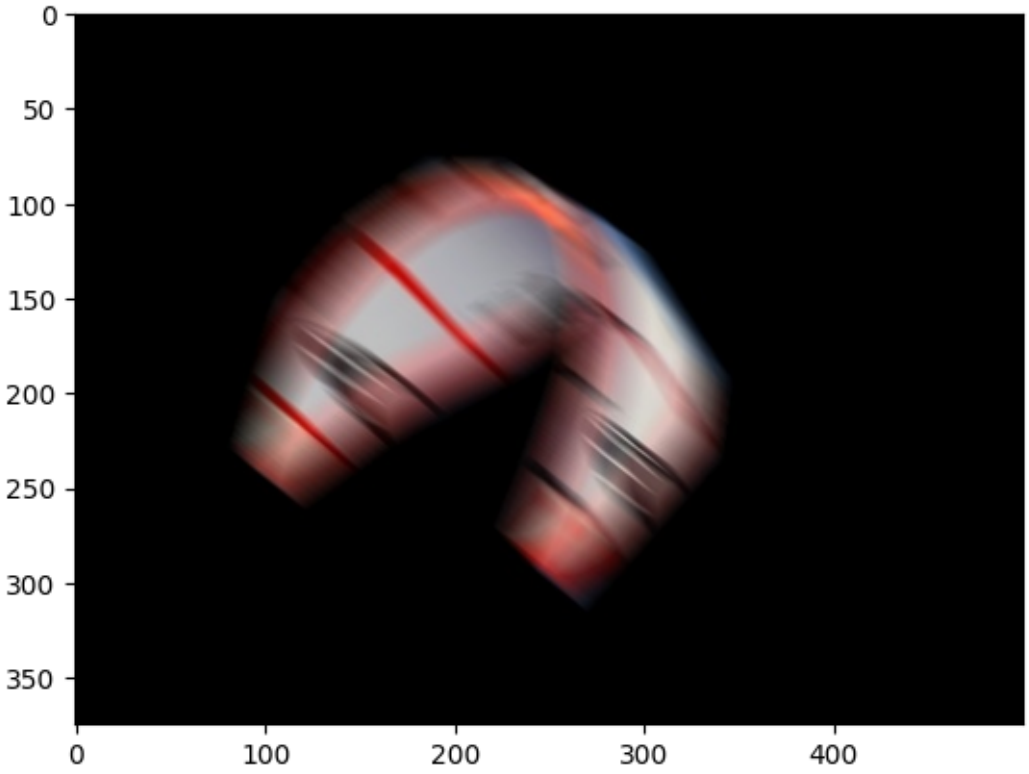
[18]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val fg␣
 ↪blurred/000000007795.jpg')
plt.imshow(original_img)
```

[18]: <matplotlib.image.AxesImage at 0x7f9c0e8028f0>



[19]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/val fg␣
 ↪blurred/000000007784.jpg')
plt.imshow(original_img)
```

[19]: <matplotlib.image.AxesImage at 0x7f9c0e874730>



4 Matching the blurred foreground with its corresponding background

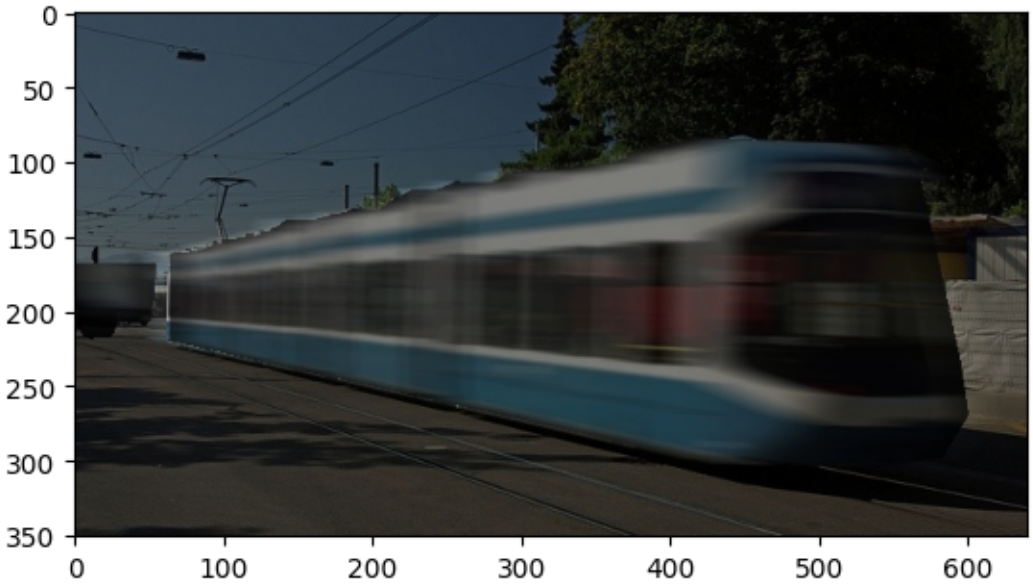
[ ]:
```
import json
from pycocotools.coco import COCO
import cv2
import numpy as np
import os

# Load the COCO annotations
ann_file = "C:/Users/mukun/Desktop/Workshop Image Dataset Generation/DATASET/
 ↪annotations_trainval2017/annotations/instances_train2017.json"
coco = COCO(ann_file)

# Load the image
img_dir = "C:/Users/mukun/Desktop/Workshop Image Dataset Generation/train2017/
 ↪train2017"
fg_dir = "C:/Users/mukun/Desktop/Workshop Image Dataset Generation/GOOD RESULTS/
 ↪TRAIN/train fg blurred"
```

```
bg_dir = "C:/Users/mukun/Desktop/Workshop Image Dataset Generation/GOOD RESULTS/
 ↪TRAIN/train bg separation"

# Get all image IDs in the dataset
img_ids = coco.imgs.keys()

# Iterate over the image IDs
for img_id in img_ids:
    # Load the foreground image
    fg_img = cv2.imread(os.path.join(fg_dir, '%012d.jpg' % img_id))

    # Load the background image
    bg_img = cv2.imread(os.path.join(bg_dir, '%012d.jpg' % img_id))

    # Superimpose the foreground image on the background image
    result = cv2.addWeighted(fg_img, 0.5, bg_img, 0.5, 0)

    # Save the output image
    cv2.imwrite(os.path.join("C:/Users/mukun/Desktop/Workshop Image Dataset␣
 ↪Generation/completed train 2017", '%012d.jpg' % img_id), result)
```

4.1 Examples

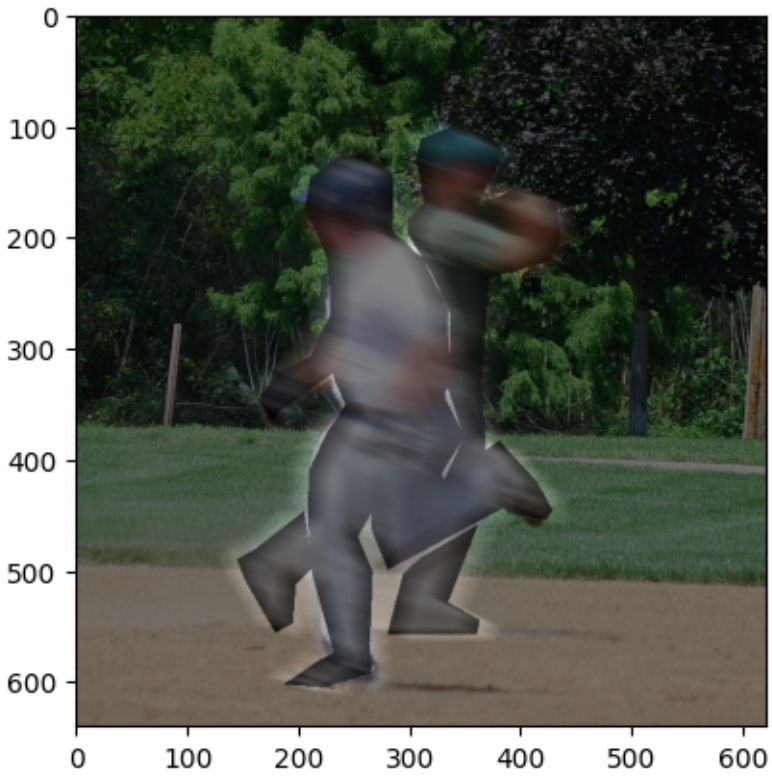
[20]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/completed␣
 ↪val 2017/000000006040.jpg')
plt.imshow(original_img)
```

[20]:
```
<matplotlib.image.AxesImage at 0x7f9c0e6fafe0>
```



[21]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/completed␣
 ↪val 2017/000000000872.jpg')
plt.imshow(original_img)
```

[21]: <matplotlib.image.AxesImage at 0x7f9c0e772f80>



[22]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/completed␣
 ↪val 2017/000000013348.jpg')
plt.imshow(original_img)
```

[22]: <matplotlib.image.AxesImage at 0x7f9c0e5fad10>

[23]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/completed␣
 ↪val 2017/000000018491.jpg')
plt.imshow(original_img)
```

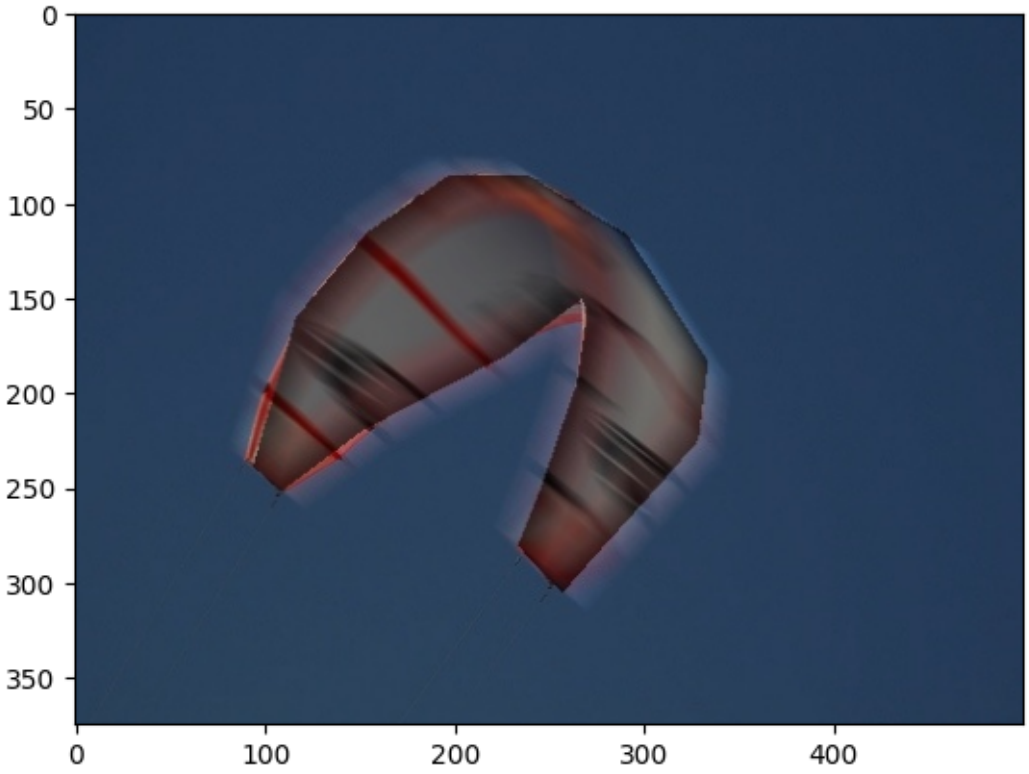
[23]:
```
<matplotlib.image.AxesImage at 0x7f9c0e67f340>
```

[24]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/completed␣
 ↪val 2017/000000007795.jpg')
plt.imshow(original_img)
```

[24]:
```
<matplotlib.image.AxesImage at 0x7f9c0e528040>
```

[25]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

original_img = mpimg.imread('/media/hrishikesh/Elements/VALIDATION/completed␣
 ↪val 2017/000000007784.jpg')
plt.imshow(original_img)
```

[25]: <matplotlib.image.AxesImage at 0x7f9c0e3b44c0>


0
50
100
150
200
250
300
350
0
100
200
300
400

[ ]: