Augmenting data

Since in the end we only have about 300 images to analyse, this will definitely not be enough to train our model well. So we augment the data by using a library called imgaug. It will create more images that can be used for training the model.

Zooming

The first function is zooming, which takes in the image, and zoom in from 1 to 1.3.

```
def zoom(image):
  zoom = iaa.Affine(scale=(1, 1.3))
  image = zoom.augment_image(image)
  return image
```

We then choose a random picture to see if the process was successful, apply the zoom function and plot it besides the original image, which will show the difference.

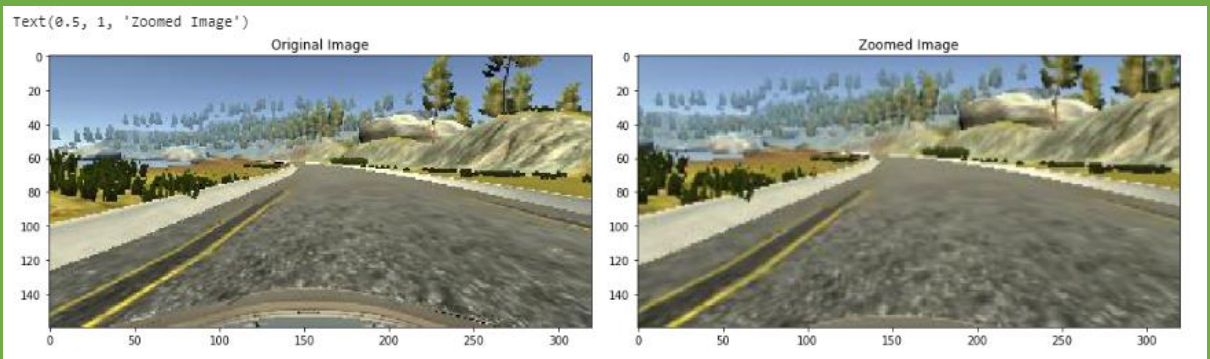
```
image = image_paths[random.randint(0, 1000)]
original_image = mpimg.imread(image)
zoomed_image = zoom(original_image)

fig, axs = plt.subplots(1, 2, figsize=(15, 10))
fig.tight_layout()

axs[0].imshow(original_image)
axs[0].set_title('Original Image')

axs[1].imshow(zoomed_image)
axs[1].set_title('Zoomed Image')
```

The image on the right side is zoomed in at some random zoom rate between 1 and 1.3. We keep the range above one so that the image can only be zoomed in not out.

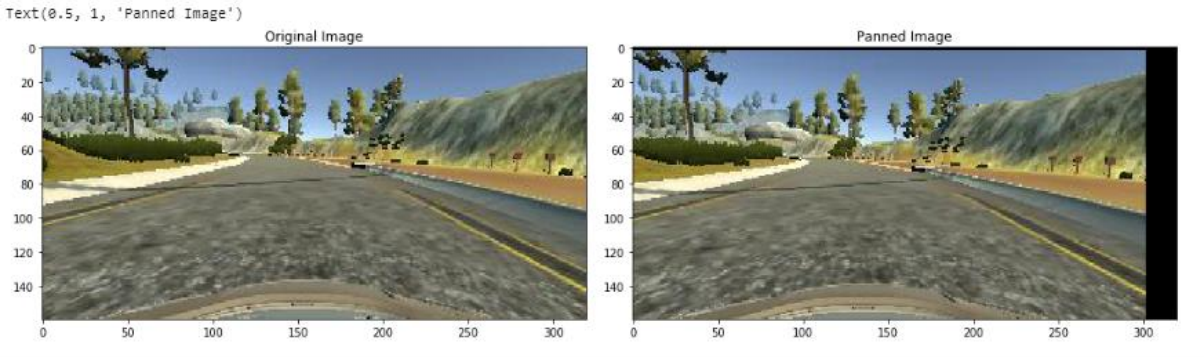


Panning

The next function we apply is panning the image. In both x and y axis we apply a panning ratio, in this case from -0.1 to 0.1.

```
def pan(image):
  pan = iaa.Affine(translate_percent= {'x': (-0.1, 0.1), 'y': (-0.1, 0.1)})
  image = pan.augment_image(image)
  return image
```

After showing the results, the difference even though small but can be noted in the corners.

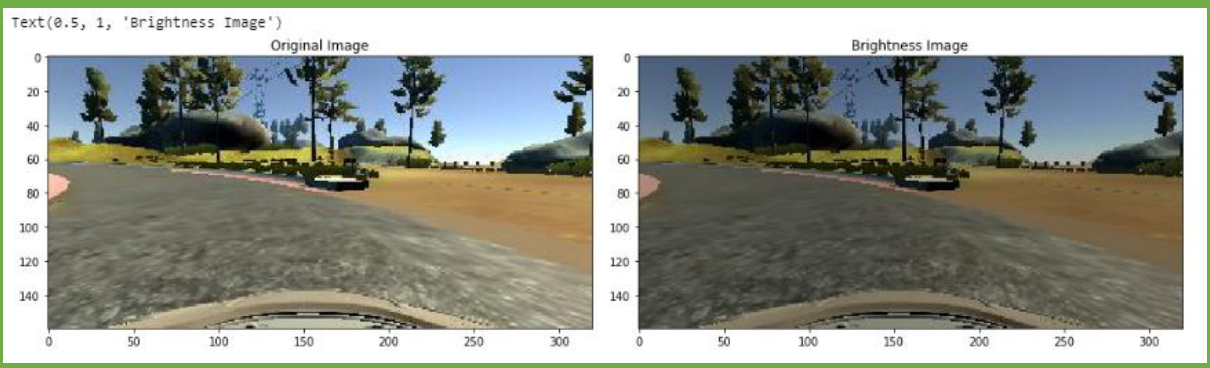


Brightness

We then change the brightness of the images, usually making it darker as dark images tend to perform well in this case. Our brightness factor is 0.2 to 1.2. Only above 1 will it become brighter than the original image.

```
def brightness(image):
  brightness = iaa.Multiply((0.2, 1.2))
  image = brightness.augment_image(image)
  return image
```

The result is clear enough, showing the original image and the edited one.

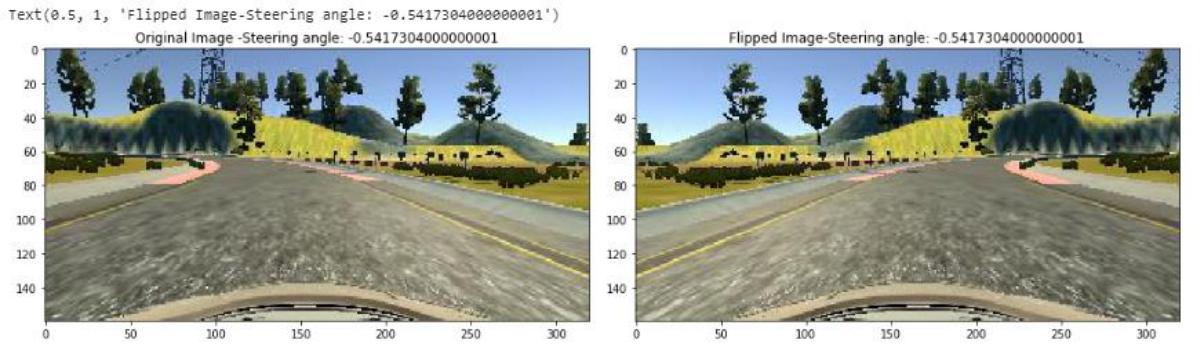


Flipping

This can be done by cv2, flipping the image horizontally (factor of 1). After the image is flipped, the angle also needs to be flipped (multiplied by 1). The other ways to flip the image would be using -1 or 0 but since we need to do it horizontally so we will use 1.

```
def img_flip(image, steering_angle):
  image = cv2.flip(image, 1)
  steering_ange = -steering_angle
  return image, steering_angle
```

This is the result of the flipped image. Original turning left and flipped image turning right, the angle also changed from (-) to (+).



Random Augmentation

The next step is to apply the augment methods to half of the images that we have. This will be done at random so as to avoid bias or errors.

```
def random_augment(image, steering_angle):
  image = mpimg.imread(image)

  if np.random.rand() < 0.5:
    image = pan(image)
  if np.random.rand() < 0.5:
    image = zoom(image)
  if np.random.rand() < 0.5:
    image = brightness(image)
  if np.random.rand() < 0.5:
    image, steering_angle = img_flip(image, steering_angle)

  return image, steering_angle
```

Batch generator

This will take all the images we have, randomly select pictures using a Boolean (istraining) to select images, augment them randomly, process them and append them to the batch image and steering arrays, thereby adding variety to our training data.

```
[83] def batch_generator(image_paths, steering_ang, batch_size, istraining):

       while True:
         batch_img = []
         batch_steering = []

         for i in range(batch_size):
           random_index = random.randint(0, len(image_paths) - 1)

           if istraining:
             im, steering = random_augment(image_paths[random_index], steering_ang[random_index])

           else:
             im = mpimg.imread(image_paths[random_index])
             steering = steering_ang[random_index]
           im = img_preprocess(im)
           batch_img.append(im)
           batch_steering.append(steering)
         yield (np.asarray(batch_img), np.asarray(batch_steering))
```

In the end we will adjust the model equation values to increase the accuracy of our model.