Explanation

Finding Image Disparity

In order to find the disparity in an image, the following steps and activities were taken as described below:

1. **Getting an image both from the left side and right side**
   A phone was used to obtain the image of a plastic bottle inside a room. The image was taken both from the left and right side. Please see images titled left_image.jpg, and right_image.jpg.

In other to reduce the coding line and make the code implementation functional in nature, the necessary functions were defined to implement the various steps that are to follow below.

2. **Read the left and right image**
   Once the images have been transferred from the phone to the laptop, the function below was used to read the left image and right image as shown below

```
# function that reads an image
def read_image(images):
    image = cv2.imread(images)
    return image

# Read the left sided and right sided images
left_side_color_image = read_image("left_image.jpg")
right_side_color_image = read_image("right_image.jpg")
```

3. **Converting read image to gray scale**
   The read-in image is in a BGR format and due to the sharp intensities at certain points in the image, it would be best to convert them to grayscale for proper implementation by the OpenCV. This conversion to grayscale is also implemented via a function as shown below.

```
# function to convert BGR image to Gray
def gray_image(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

left_image_gray = gray_image(left_side_color_image) # OpenCV will read it as BGR not RGB
right_image_gray = gray_image(right_side_color_image) # OpenCV will read it as BGR not RGB
```

4. **Perform Disparity.**

   Now that we have our image in the grayscale format, we are ready to perform the disparity between the two sides of the image. Although the disparity is performed using a single function, it is implemented in two steps as shown in the function below and explained subsequently.

   ```
   # function that implements disparity with inbuilt stereovision function
   def disparity_function(image_side_one, image_side_two):
      stereo_vision = cv2.StereoBM_create(numDisparities=16, blockSize=15)
      disparities = stereo_vision.compute(image_side_two, image_side_one)
      return disparities

   # Perform the disparity
   image_disparities=disparity_function(image_side_one=left_image_gray,image_side_two=right_img
   e_gray)
   ```

   First the two sides of the image are passed into the function which is then implements an OpenCV in-built function called the in-block matching or cv2.StereoBM_create() where the disparity is computed by comparing the sum of absolute differences (SAD) of each 'block' of pixels.

   ```
   stereo_vision = cv2.StereoBM_create(numDisparities=16, blockSize=15)
   ```

   Finally, the result from the first step is now used to find the disparity of both sides of the image

   ```
   disparities = stereo_vision.compute(image_side_two, image_side_one)
   ```

5. **Converting image back to RGB for Matplotlib**

   From previous conversion, OpenCV has read the image in the BGR format. This would not be suitable for matplotlib to use in order to plot the disparity and image itself, thus a re-conversion is needed. This conversion from BGR to RGB is done by the function shown below

```
# function that converts BGR to RGB

def RGB_to_BGR(image):

   return cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Covert images to format that can be displayed by matplotlib

# Recall OpenCV reads image in BGR, so we need to convert it back to RGB

left_side_RGB_image = RGB_to_BGR(image=left_side_color_image)

right_side_RGB_image = RGB_to_BGR(image=right_side_color_image)
```

6. **Plotting the Image and Disparity**

Finally the image is plotted for both sides as well as the disparity image using the functions below

```
# function to plot image
def plot_image(image):
    plt.figure()
    plt.imshow(image)
    plt.show()

# function to plot disparity
def plot_disparity(disparity, color_type):
    plt.figure()
    plt.imshow(disparity,color_type)
    plt.show()

# Plot the disparities and image
plot_image(left_side_RGB_image)
plot_image(right_side_RGB_image)
plot_disparity(image_disparities, "gray")
```

7. **Saving the plots**

In order to view the images outside of the coding environment, the plotted images are saved using the function shown below.

```
def save_image(image_name_format, image):
    """This function saves any image to a given path"""
    cv2.imwrite(image_name_format, image)

# saving images
save_image('left_side_RGB_image.jpg', left_side_RGB_image)
save_image('right_side_RGB_image.jpg', right_side_RGB_image)
save_image('image_disparities.jpg', image_disparities)
```