TASK 1

Performing image arithmetic operations:

- Adding:

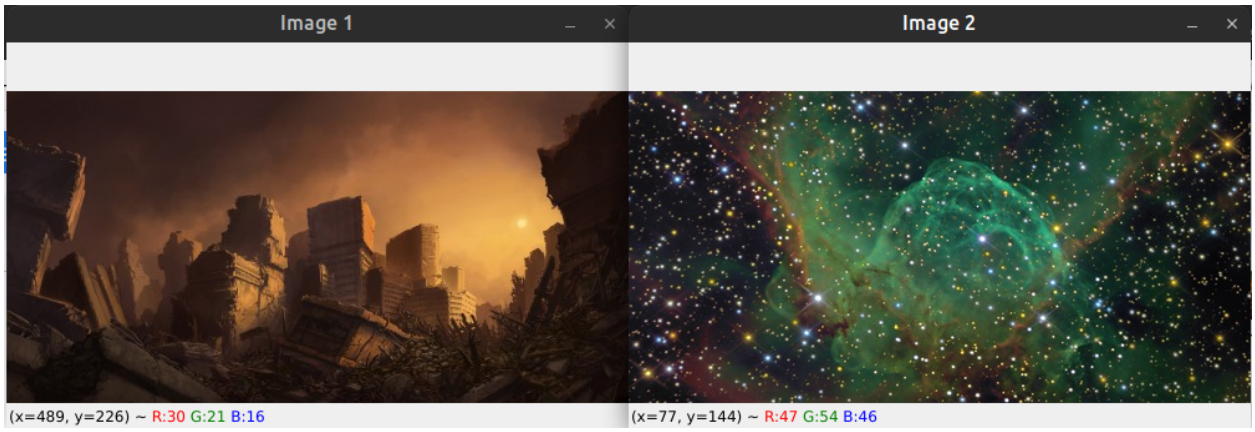
```
import cv2
import numpy as np

image1 = cv2.imread('image1.jpg')
image2 = cv2.imread('image2.jpg')
added = cv2.add(image1, image2)
added2 = cv2.addWeighted(image1, 0.5, image2, 0.4, 0)
cv2.imshow('Image 1', image1)
cv2.imshow('Image 2', image2)
cv2.imshow('add operation', added)
cv2.imshow('addWeighted', added2)
if cv2.waitKey(0):
   cv2.destroyAllWindows()
```

Output:

- Subtraction:

```
import cv2
import numpy as np
image1 = cv2.imread('image1.jpg')
image2 = cv2.imread('image2.jpg')
sub = cv2.subtract(image1, image2)
cv2.imshow('Image 1', image1)
cv2.imshow('Image 2', image2)
cv2.imshow('subtract operation', sub)

if cv2.waitKey(0):
   cv2.destroyAllWindows()
```

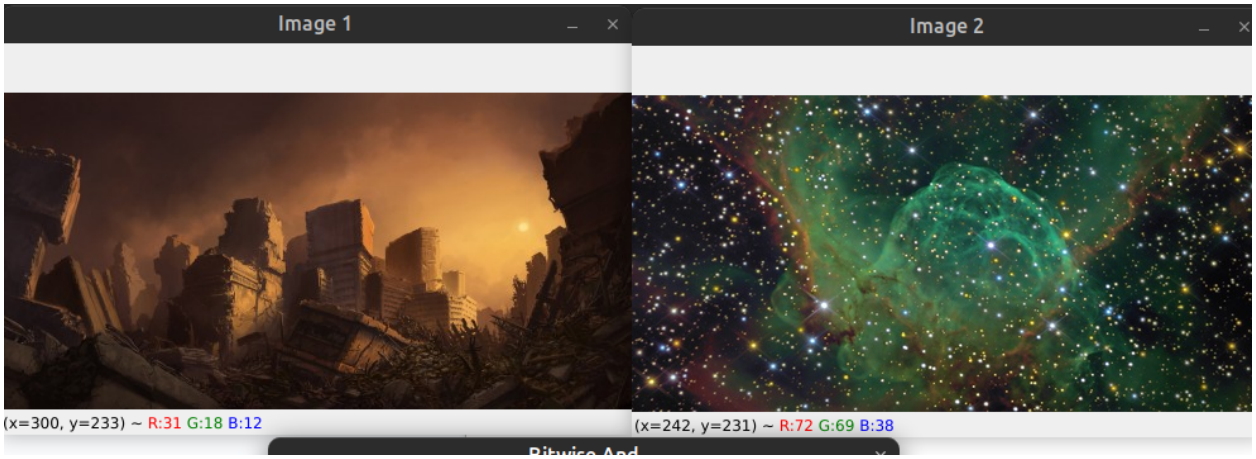
Output:

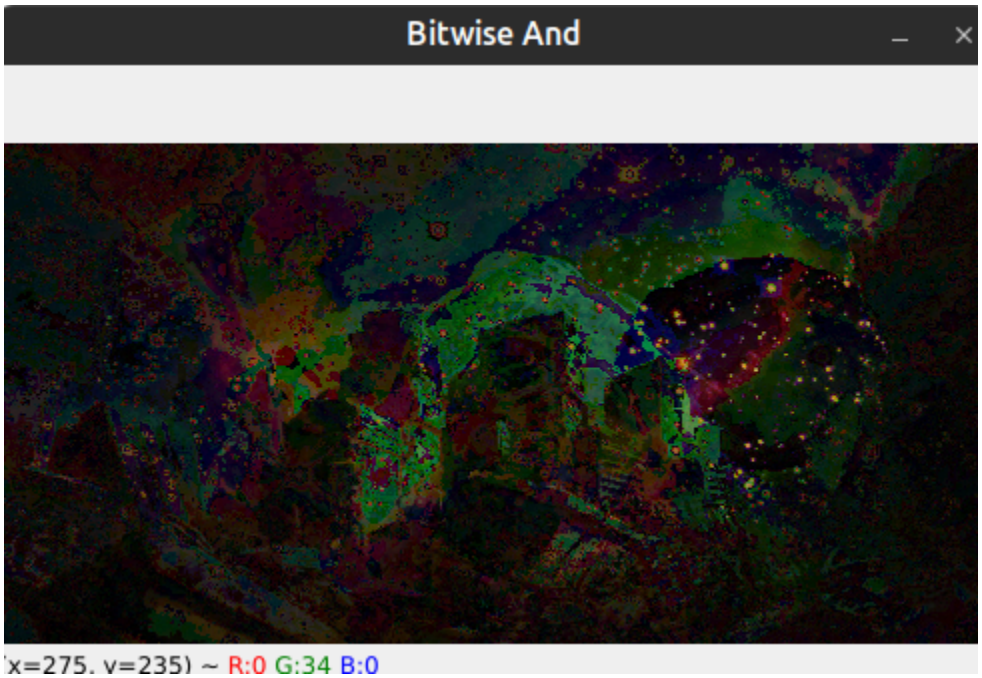


- Bitwise AND operation:

```
andOp = cv2.bitwise_and(image1, image2, mask = None)
cv2.imshow('Image 1', image1)
cv2.imshow('Image 2', image2)
cv2.imshow('Bitwise And', andOp)
cv2.waitKey(0)
if cv2.waitKey(0):
   cv2.destroyAllWindows()
```

Output:

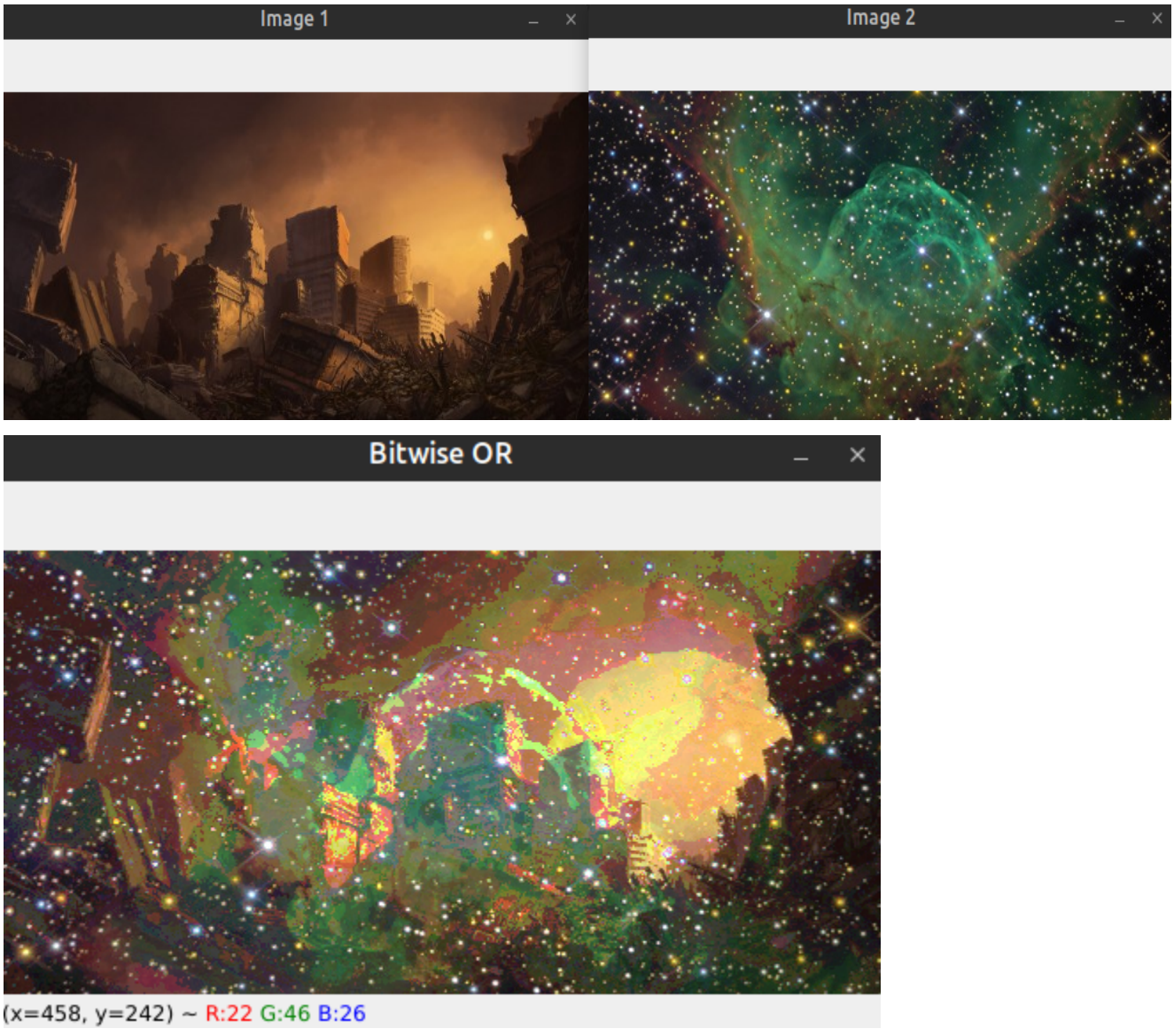




- Bitwise OR operation:

```
notOp = cv2.bitwise_not(img1, mask = None)
cv2.imshow('Image 1', image1)
cv2.imshow('Image 2', image2)
orOP = cv2.bitwise_or(img1, img2, mask = None)
cv2.imshow('Bitwise OR', orOp)
cv2.waitKey(0)
```

Output:

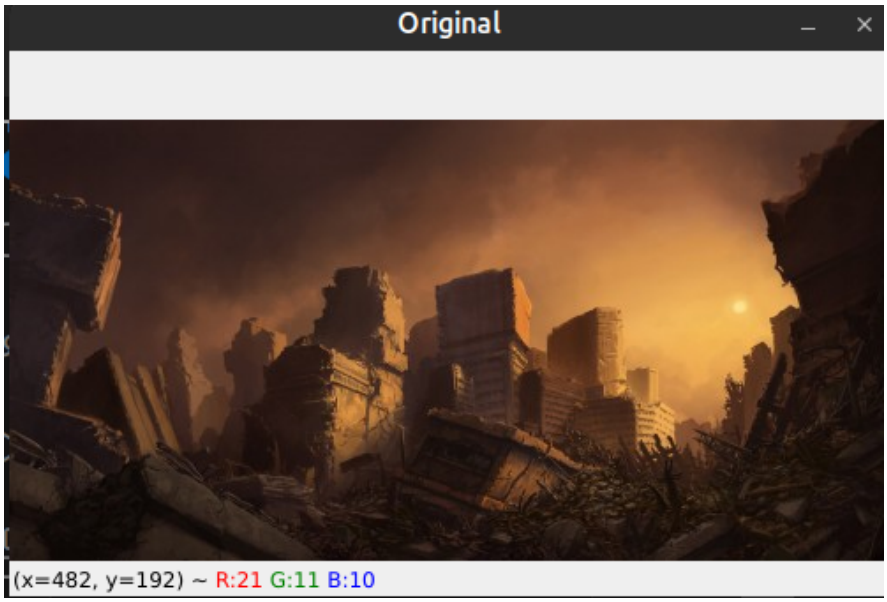


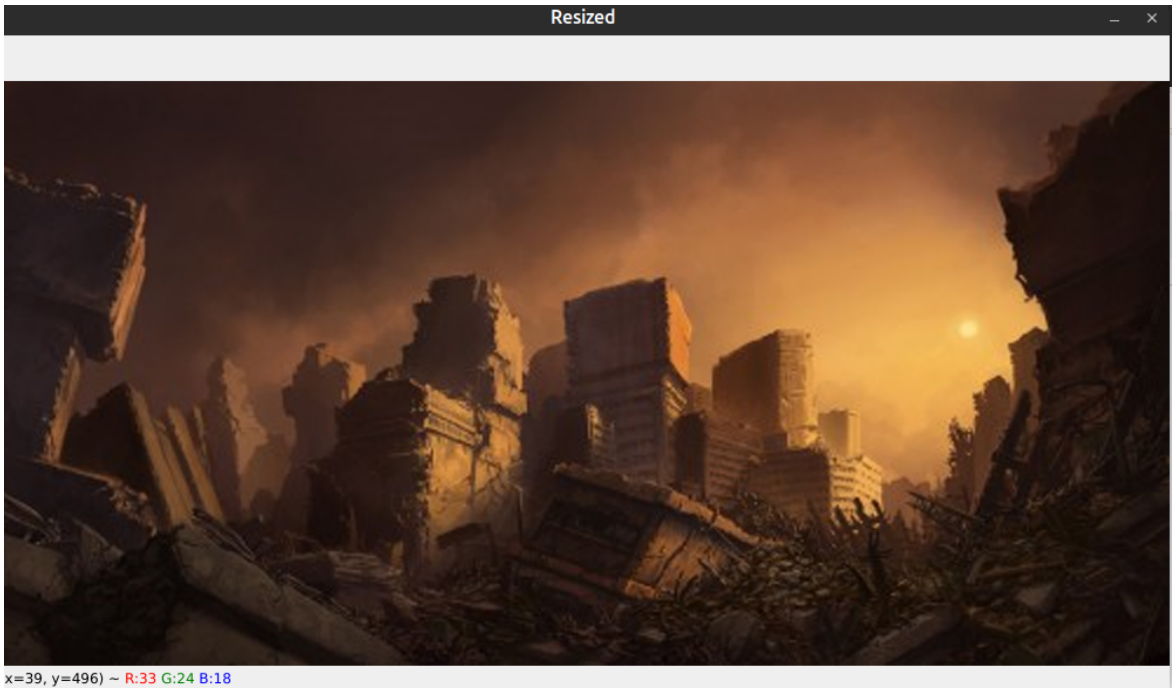
Applying geometric transformation for the image:

- Scaling:

```
import cv2
import numpy as np

img = cv2.imread('image1.jpg')
res = cv2.resize(img,None,fx=2, fy=2, interpolation =
cv2.INTER_CUBIC)

cv2.imshow("Original", img)
cv2.imshow("Resized", res)

cv2.waitKey()
cv2.destroyAllWindows()
```

Output:

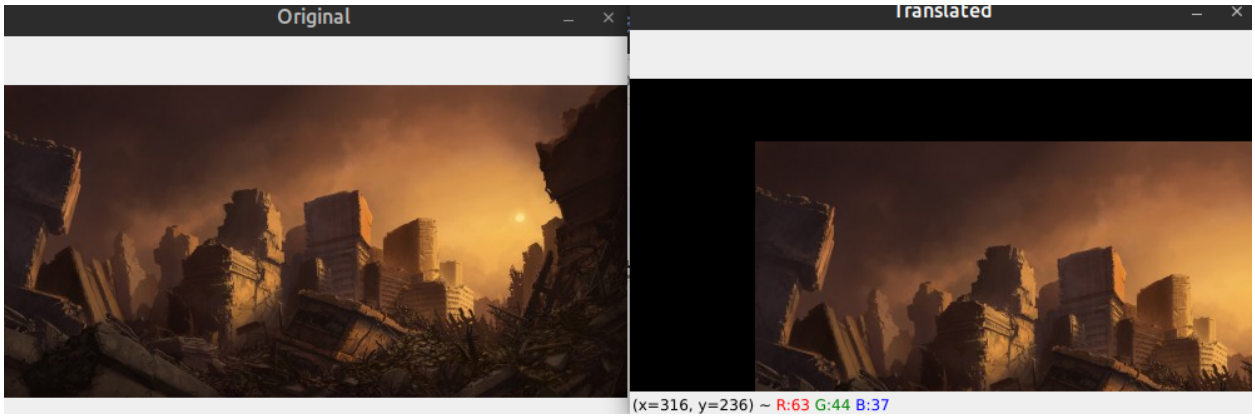




- Translation:

```
import cv2
import numpy as np

img = cv2.imread('image1.jpg')
rows,cols,ch = img.shape
M = np.float32([[1,0,100],[0,1,50]])
dst = cv2.warpAffine(img,M,(cols,rows))

cv2.imshow("Original", img)
cv2.imshow("Translated", dst)

cv2.waitKey()
cv2.destroyAllWindows()
```

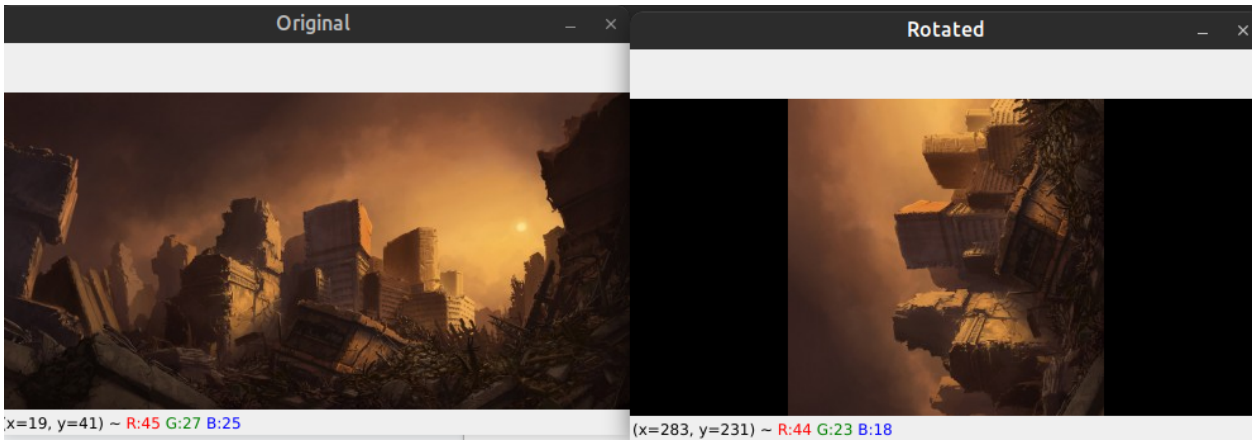
Output:



- Rotation:

```
import cv2
import numpy as np

img = cv2.imread('image1.jpg')
rows,cols,ch = img.shape
M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)
dst = cv2.warpAffine(img,M,(cols,rows))
cv2.imshow("Original", img)
cv2.imshow("Rotated", dst)
cv2.waitKey()
cv2.destroyAllWindows()
```

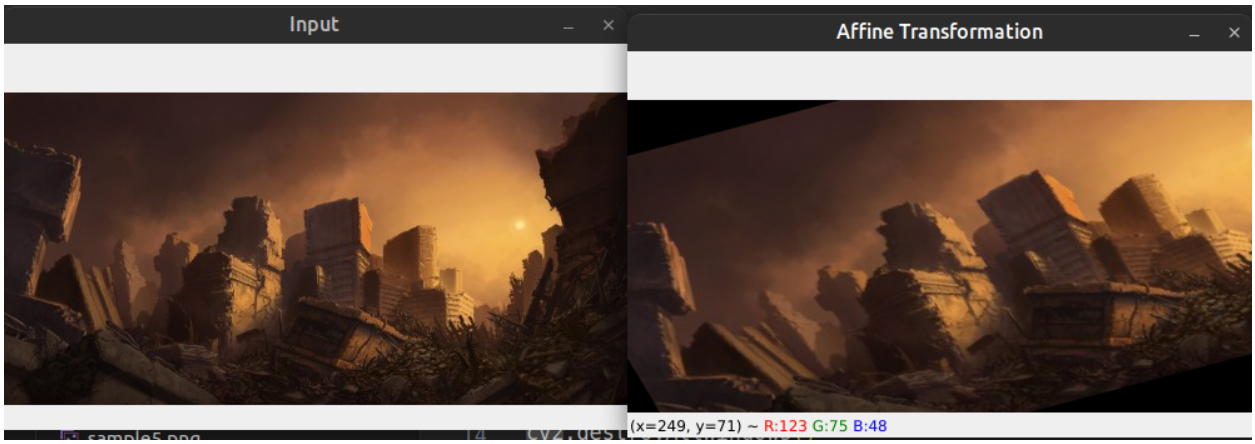
Output:



- Affine Transformation:

```
import cv2
import numpy as np

img = cv2.imread('image1.jpg')
rows, cols, ch = img.shape
pts1 = np.float32([[50, 50],
                   [200, 50],
                   [50, 200]])
pts2 = np.float32([[10, 100],
                   [200, 50],
                   [100, 250]])
M = cv2.getAffineTransform(pts1, pts2)
dst = cv2.warpAffine(img, M, (cols, rows))

cv2.imshow('Input', img)
cv2.imshow('Affine Transformation', dst)
cv2.waitKey()
cv2.destroyAllWindows()
```

Output:

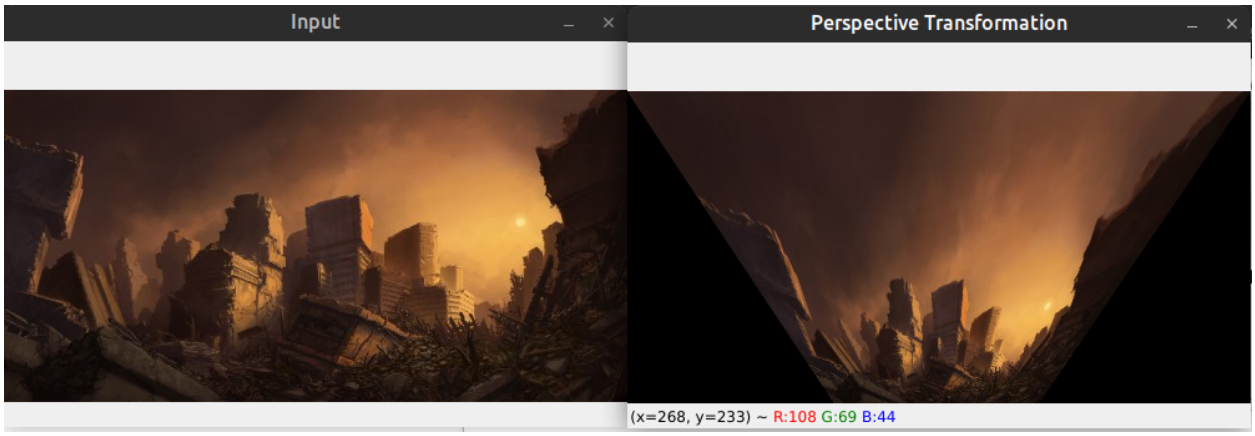


- Perspective Transformation:

```
import cv2
import numpy as np

img = cv2.imread('image1.jpg')
```

```
rows, cols = img.shape[:2]

src_points = np.float32([[0,0], [cols-1,0], [0,rows-1],
[cols-1,rows-1]])
dst_points = np.float32([[0,0], [cols-1,0], [int(0.33*cols),rows-1],
[int(0.66*cols),rows-1]])
projective_matrix = cv2.getPerspectiveTransform(src_points,
dst_points)
img_output = cv2.warpPerspective(img, projective_matrix,
(cols,rows))

cv2.imshow('Input', img)
cv2.imshow('Perspective Transformation', img_output)
cv2.waitKey()
cv2.destroyAllWindows()
```

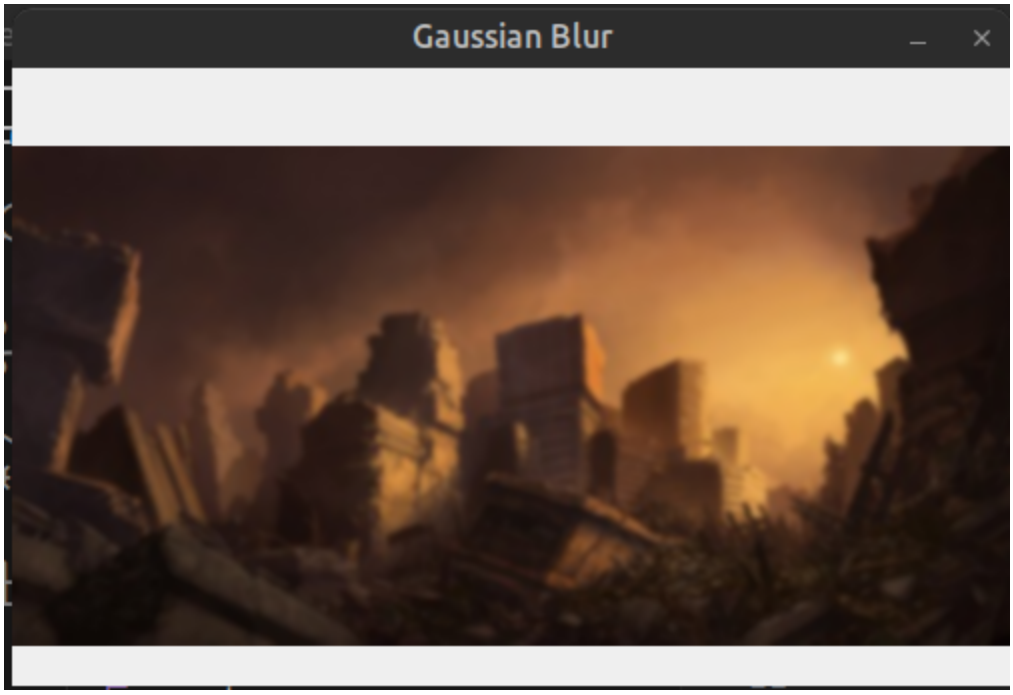
Output:



Blurring and Deblurring:

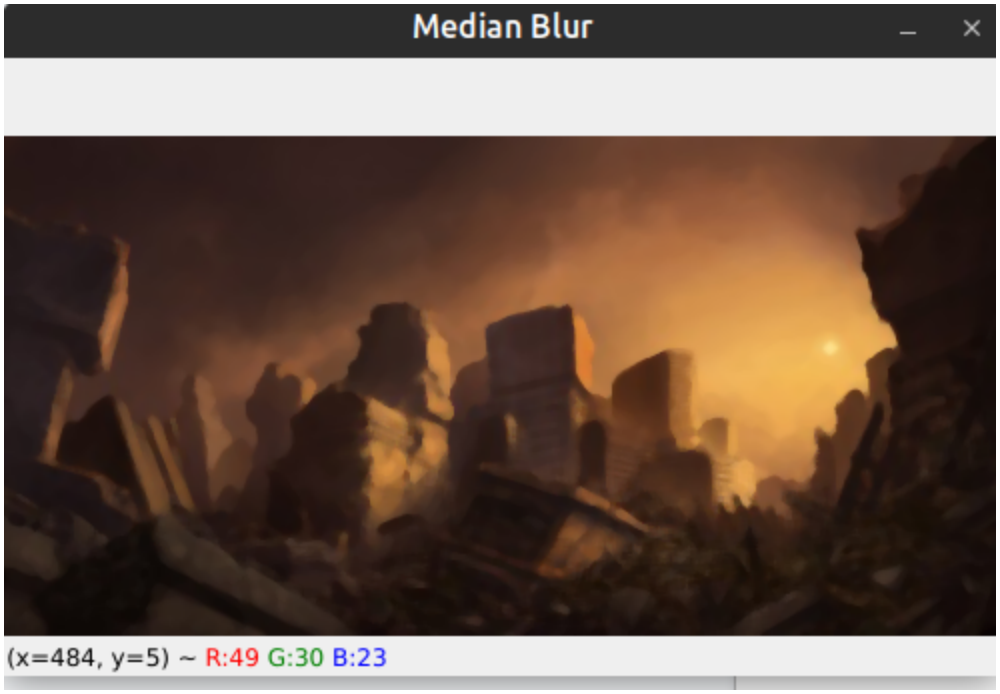
- **Gaussian Blur:**

```
import cv2
import numpy as np
 image = cv2.imread('image1.jpg')
 cv2.imshow('Original Image', image)
cv2.waitKey(0)
Gaussian = cv2.GaussianBlur(image, (7, 7), 0)
cv2.imshow('Gaussian Blur', Gaussian)
cv2.waitKey(0)
```



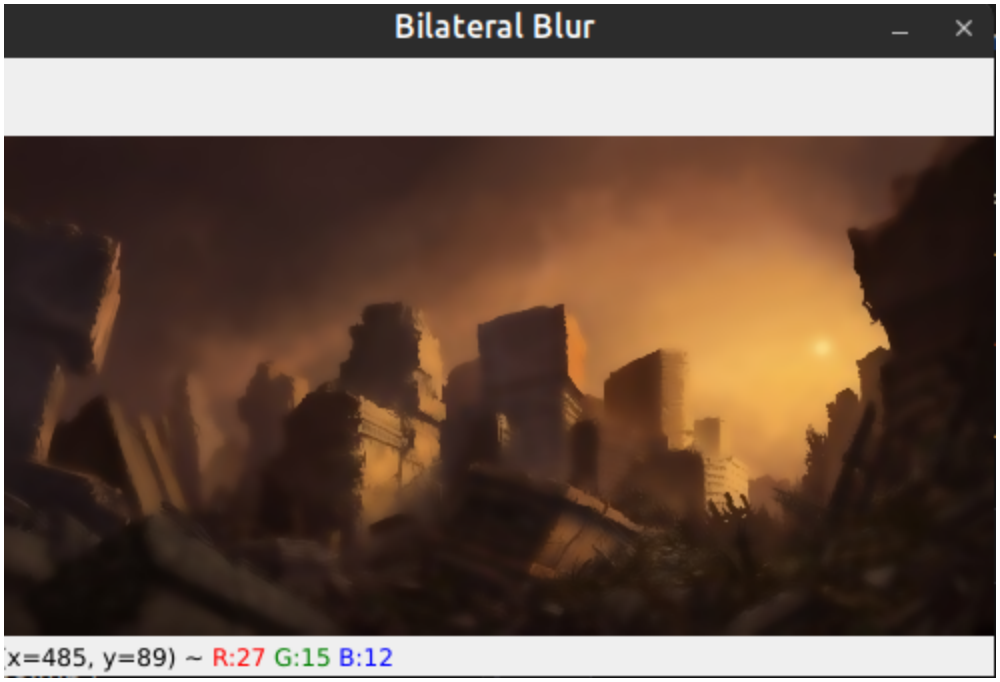
- **Median Blur:**

```
import cv2
import numpy as np
image = cv2.imread('image1.jpg')
cv2.imshow('Original Image', image)
median = cv2.medianBlur(image, 5)
cv2.imshow('Median Blur', median)
cv2.waitKey(0)
```



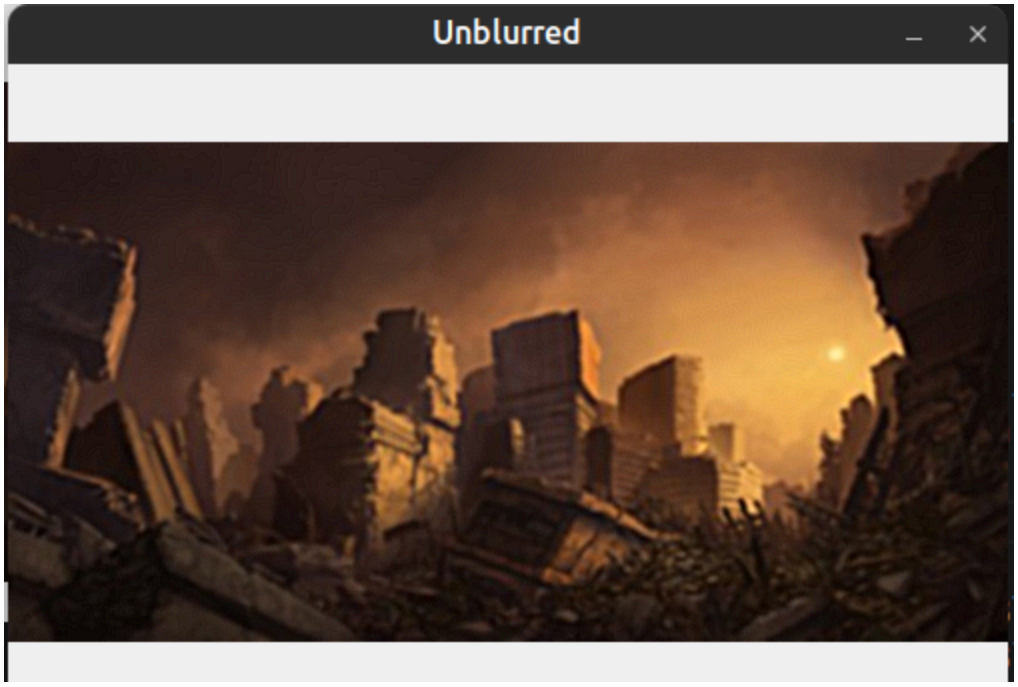
- **Bilateral Blur:**

```
import cv2
import numpy as np
image = cv2.imread('image1.jpg')
cv2.imshow('Original Image', image)
bilateral = cv2.bilateralFilter(image, 9, 75, 75)
cv2.imshow('Bilateral Blur', bilateral)
cv2.waitKey(0)
```



- **Unblurring by sharpening the image:**

```
import cv2
import numpy as np
image = cv2.imread('image1.jpg')
cv2.imshow('Original Image', image)
sharpen_kernel = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])
sharpen = cv2.filter2D(Gaussian, -1, sharpen_kernel)

cv2.imshow('Unblurred', sharpen)
cv2.waitKey()
cv2.destroyAllWindows()
```



Applying Treshold and Grayscale on each frame of a video:

```
import cv2
import numpy as np
cap = cv2.VideoCapture('video.mp4')
while (cap.isOpened()):
   ret, frame = cap.read()
   try:
       frame = cv2.resize(frame, (540, 380), fx = 0, fy = 0,
                          interpolation = cv2.INTER_CUBIC)
       cv2.imshow('Frame', frame)
       gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
       Thresh = cv2.adaptiveThreshold(gray, 255,
cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY_INV, 11, 2)
       cv2.imshow('Thresh', Thresh)
   except Exception as e:
       break;
   if cv2.waitKey(25) & 0xFF == ord('q'):
       break
cap.release()
cv2.destroyAllWindows()
```