Chapter 2: More Image Transformation and Manipulation

Problems

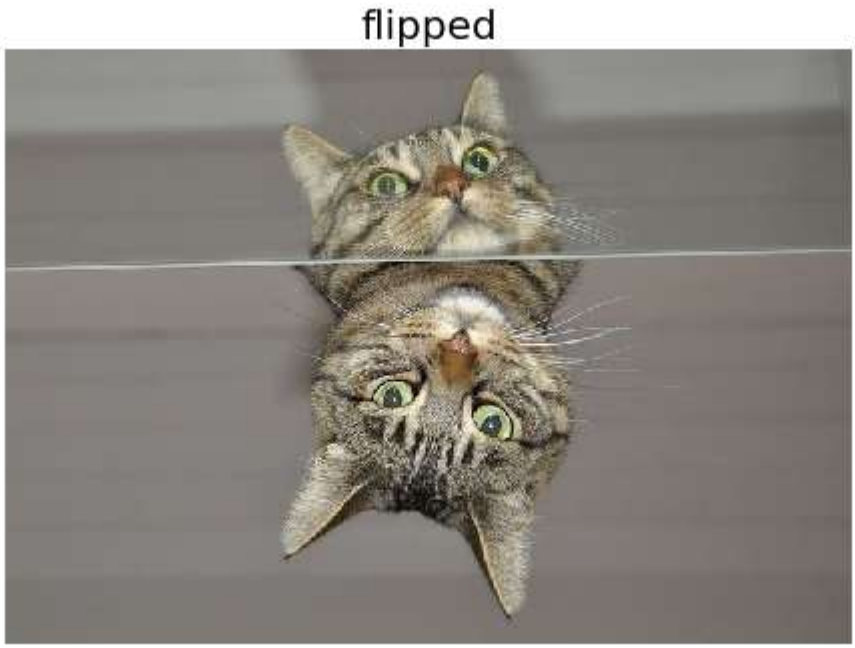
1. Applying Euclidean and Affine Transformation on an image

1.1 Rotating an image with scipy.ndimage

In [7]:
```
im = rotate(im, -45)
plt.figure(figsize=(5,5))
plt.imshow(im)
plt.axis('off') # stop showing the axes
plt.show()
```

1.2 Flipping and Flopping an image with *numpy*

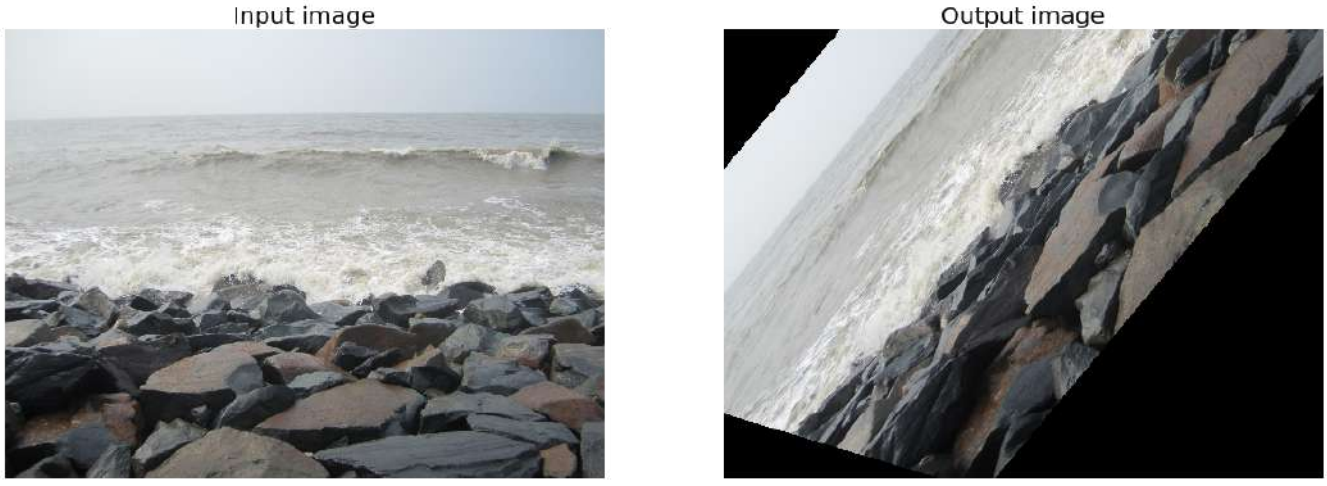
In [9]:
```
plt.figure(figsize=(10, 12))
plt.subplot(211), plt.imshow(im), plt.axis('off'), plt.title('original', size=20)
plt.subplot(212), plt.imshow(im_filpped), plt.axis('off'), plt.title('flipped', size=20)
plt.show()
```



In [14]:
```
im = plt.imread('images/Img_02_43.jpeg')
im_filpped = np.fliplr(im)
plt.figure(figsize=(15, 12))
plt.subplot(121), plt.imshow(im), plt.axis('off'), plt.title('original', size=20)
plt.subplot(122), plt.imshow(im_filpped), plt.axis('off'), plt.title('flopped', size=20)
plt.show()
```

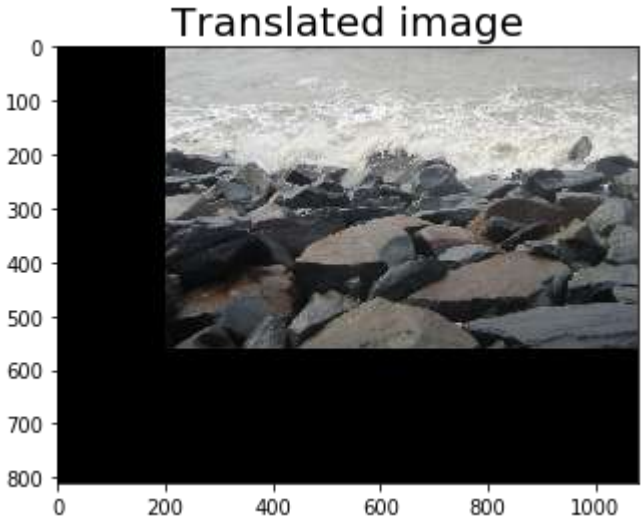
1.3 Applying Affine Transformation with *scipy.ndimage*

In [79]:
```
plt.figure(figsize=(20,10))
plt.subplot(121), plt.imshow(im), plt.axis('off'), plt.title('Input image', size=20)
plt.subplot(122), plt.imshow(transformed), plt.axis('off'), plt.title('Output image', si
plt.show()
```



2. Implement Image Transformation with Warping / Inverse Warping using scikit-image and scipy.ndimage

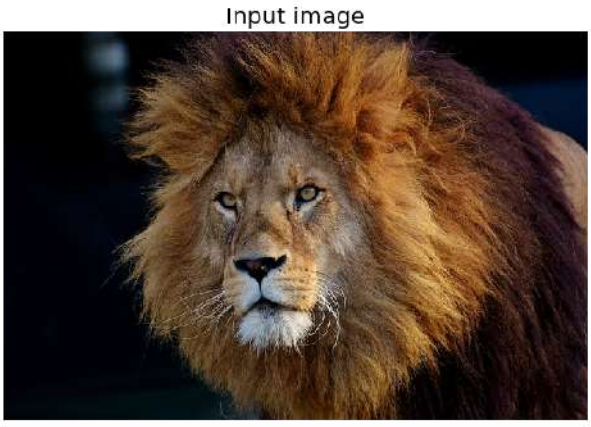
In [72]:
```
im = imread('images/Img_02_01.jpg')
im = warp(im, translate, map_args={'t_x':-250, 't_y':200}) # create a dictionary for tra
plt.imshow(im)
plt.title('Translated image', size=20)
plt.show()
```



2.2 Implementing the Swirl transformation using scikit-image warp

In [34]:
```
im = imread('images/Img_02_02.jpg')
print(im.shape)
im1 = warp(im, swirl, map_args={'x0':220, 'y0':360, 'R':650})
plt.figure(figsize=(20,10))
plt.subplot(121), plt.imshow(im), plt.axis('off'), plt.title('Input image', size=20)
plt.subplot(122), plt.imshow(im1), plt.axis('off'), plt.title('Output image', size=20)
plt.show()
```

```
(480, 720, 3)
```



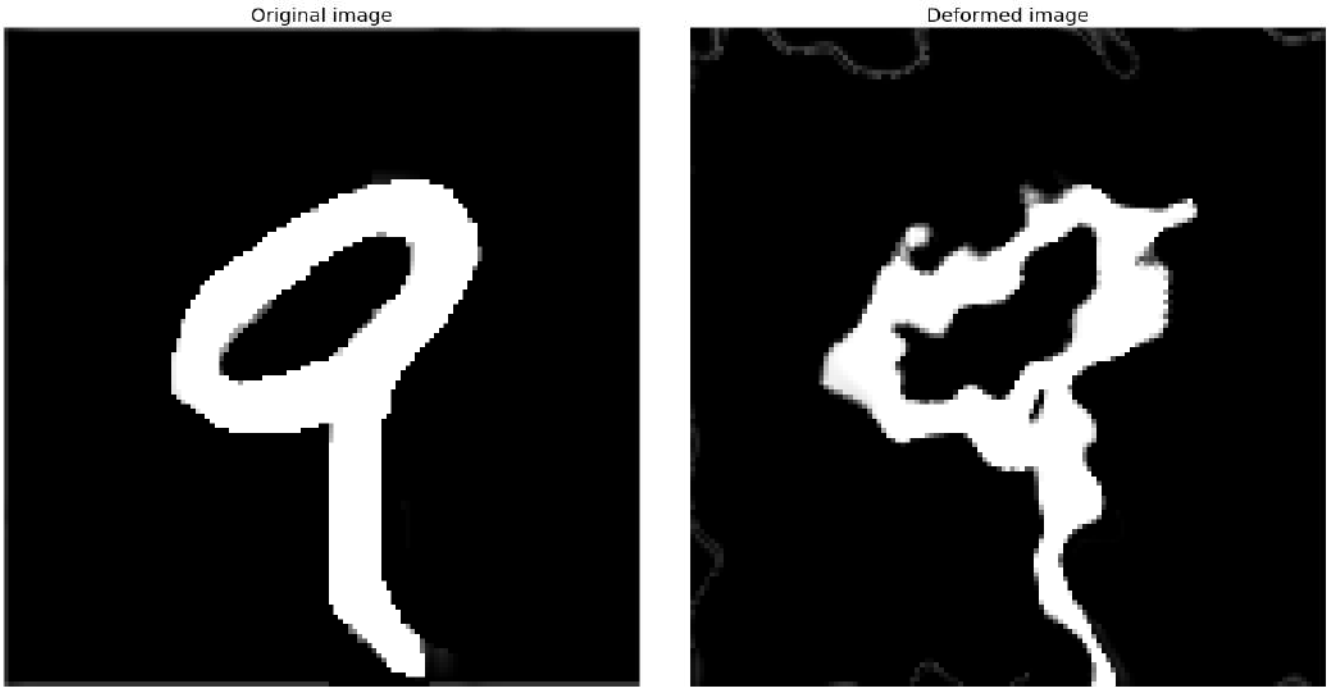
2.3 Implementing Swirl Transform using *scipy.ndimage*

In [4]:
```
im = rgb2gray(imread('images/Img_02_06.jpg'))
print(im.shape)
im1 = ndi.geometric_transform(im, apply_swirl, extra_arguments=(100, 100, 250))
plt.figure(figsize=(20,10))
plt.gray()
plt.subplot(121), plt.imshow(im), plt.axis('off'), plt.title('Input image', size=20)
plt.subplot(122), plt.imshow(im1), plt.axis('off'), plt.title('Output image', size=20)
plt.show()
```

```
(220, 220)
```

2.4 Implementing Elastic Deformation

In [15]:
```
img = rgb2gray(plt.imread('images/Img_02_22.png'))
img1 = elastic_transform(img, 100, 4)
plt.figure(figsize=(20,10))
plt.gray()
plt.subplot(121), plt.imshow(img), plt.axis('off'), plt.title('Original image', size=20)
plt.subplot(122), plt.imshow(img1), plt.axis('off'), plt.title('Deformed image', size=20)
plt.tight_layout()
plt.show()
```



3. Image Projection with Homography using scikit-image

In [1]:
```
plt.figure(figsize=(30,10))
plt.subplot(131), plt.imshow(im_src, cmap='gray'), plt.axis('off'), plt.title('Source im
plt.subplot(132), plt.imshow(im_dst, cmap='gray'), plt.axis('off'), plt.title('Destinati
plt.subplot(133), plt.imshow(im_out, cmap='gray'), plt.axis('off'), plt.title('Output im
plt.tight_layout()
plt.show()
```

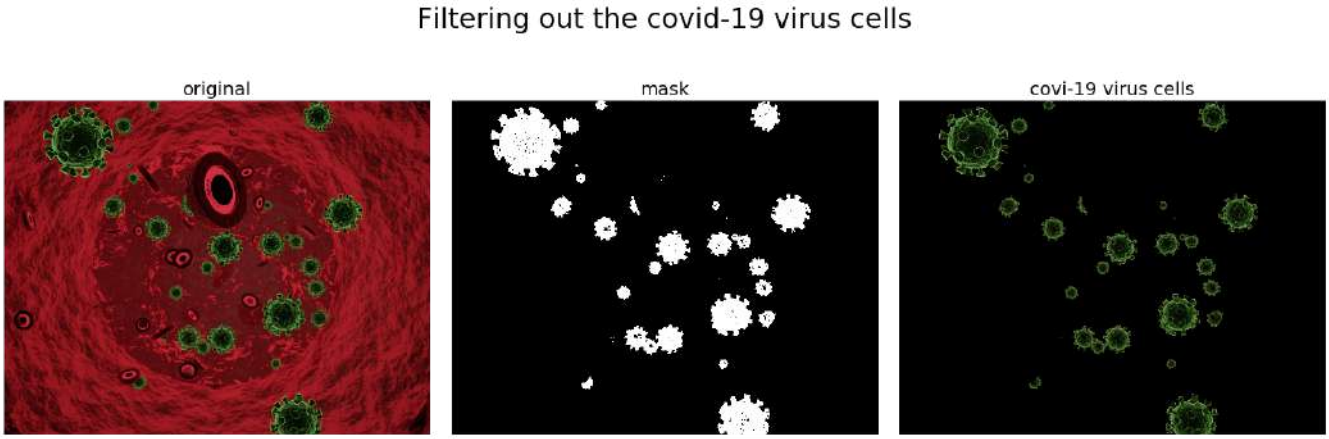
```
(379, 261, 3) (321, 450, 3)
```

4. Detecting Colors and Changing Colors of Objects with opencv-python

In [12]:
```
plt.figure(figsize=(20,10))
plt.subplots_adjust(0,0,1,0.9,0.01,0.075)
plt.subplot(131), plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)), plt.axis('off'), plt
plt.subplot(132), plt.imshow(cv2.cvtColor(brown, cv2.COLOR_BGR2RGB)), plt.axis('off'),
plt.subplot(133), plt.imshow(cv2.cvtColor(black, cv2.COLOR_BGR2RGB)), plt.axis('off'),
plt.suptitle('Detecting and changing object colors with opencv-python', size=25)
plt.show()
```

Detecting Covid-19 Virus Objects with Colors in HSV colorspace

In [53]:
```
plt.figure(figsize=(20, 8))
plt.gray()
plt.subplots_adjust(0,0,1,0.975,0.05,0.05)
plt.subplot(131), plt.imshow(img), plt.axis('off'), plt.title('original', size=20)
plt.subplot(132), plt.imshow(green_mask), plt.axis('off'), plt.title('mask', size=20)
plt.subplot(133), plt.imshow(output_img), plt.axis('off'), plt.title('covi-19 virus cel
plt.suptitle('Filtering out the covid-19 virus cells', size=30)
plt.show()
```

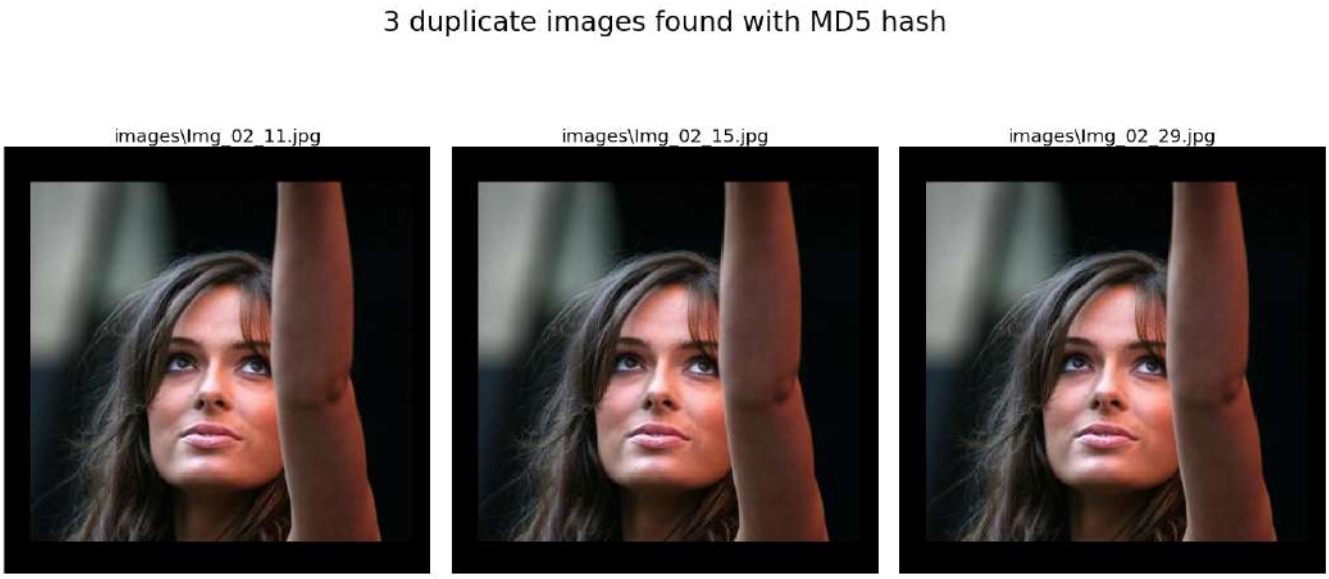


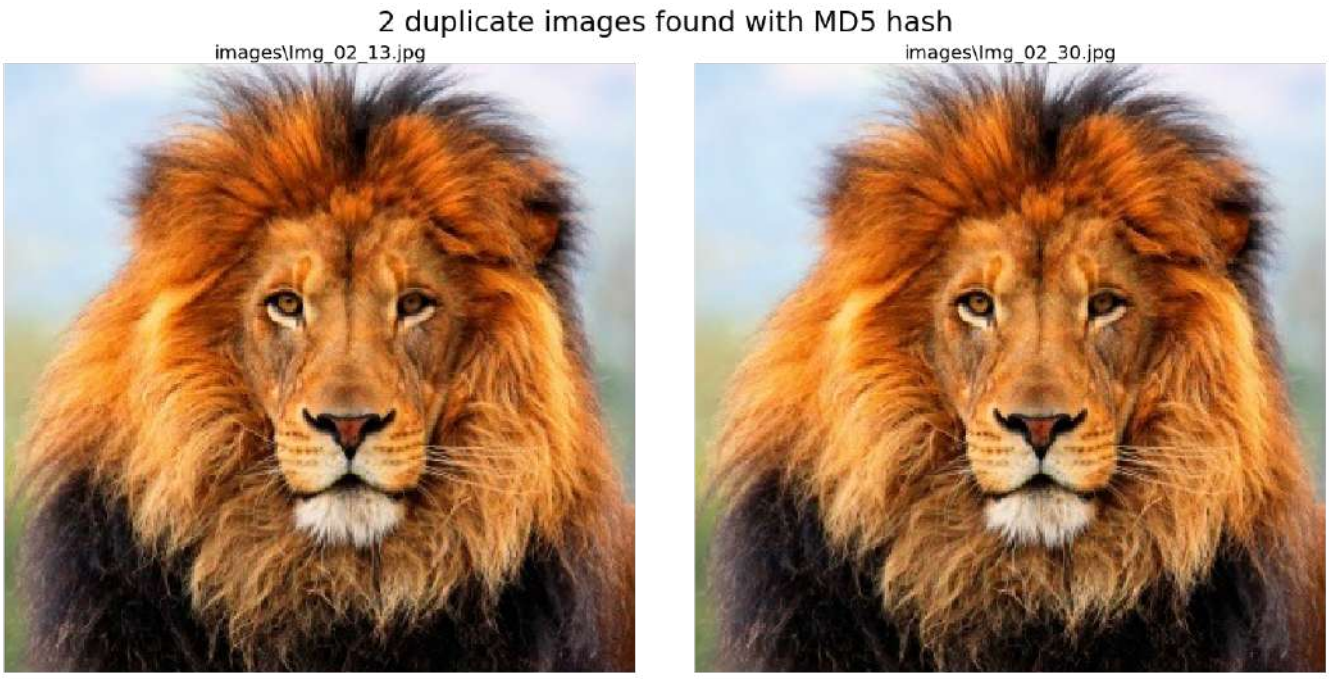
5. Finding Duplicate and Similar Images with Hashing

5.1 Using Cryptographic (MD5) Hash functions to find duplicate images with hashlib

In [31]:
```
duplicates = find_duplicates('images/*.*')
print(duplicates)
show_duplicates(duplicates)
```

```
[['images\\Img_02_11.jpg', 'images\\Img_02_15.jpg', 'images\\Img_02_29.jpg'], ['images
\\Img_02_13.jpg', 'images\\Img_02_30.jpg']]
```





5.2 Using Perceptual Hash function (phash) to find similar images using imagehash

In [45]:
```
plot_images_to_compare('images/Img_02_31.jpg', 'images/Img_02_32.jpg')
```

```
sizes of images = (300, 258), (300, 258)
```


```
hash1 = 1001101101001000011001110110011010010100100110011011001101100011 (9b4867669499
b363), length = 64 bits
hash2 = 1001101101001000011001110110011010010100100110011011001101100011 (9b4867669499
b363), length = 64 bits
hamming distance = 0
```

In [58]:
```
plot_images_to_compare('images/Img_02_31.jpg', 'images/Img_02_43.png')
```

```
sizes of images = (300, 258), (300, 258)
```


```
hash1 = 1001101101001000011001110110011010010100100110011011001101100011 (9b4867669499
b363), length = 64 bits
hash2 = 1001101101001000011001110110011010010100100111010011011000011 (9b4867669499
d363), length = 64 bits
hamming distance = 2
```

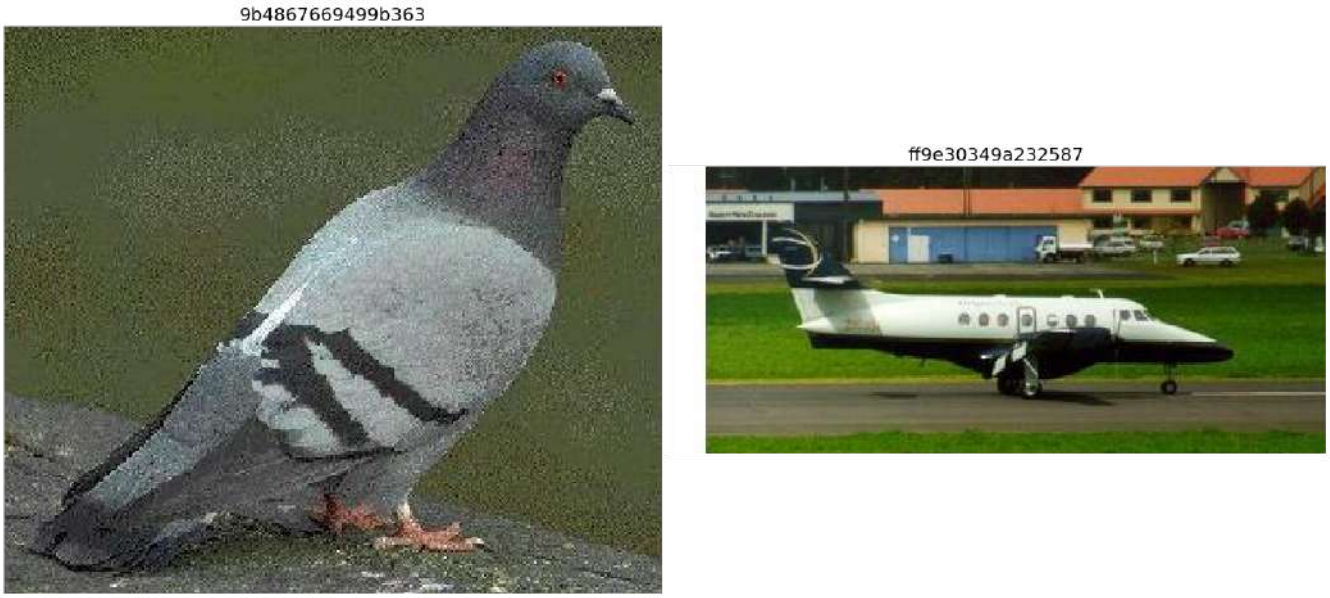
In [151]:
```
plot_images_to_compare('images/similar/Img_02_41.jpg', 'images/similar/Img_02_41.png')
```

```
sizes of images = (1024, 683), (574, 383)
```


```
hash1 = 1001010110000001011010111101001010010000111100100111001001111110 (95816bd290f2
727e)
hash2 = 1001010110000001011010101101001010010010111100100111001001111110 (95816ad292f2
727e)
hamming distance = 2
```

In [21]:
```
plot_images_to_compare('images/Img_02_31.jpg', 'images/Img_02_35.jpg')
```

```
sizes of images = (300, 258), (399, 174)
```




```
hash1 = 1001101101001000011001110110011010010100100110011011001101100011 (9b4867669499
b363)
hash2 = 1111111110011110001100000011010010011010001000110010010110000111 (ff9e30349a23
2587)
hamming distance = 32
```

In [74]:
```
plot_query_returned_images(query, found)
```