Chapter 6: More Image Enhancement

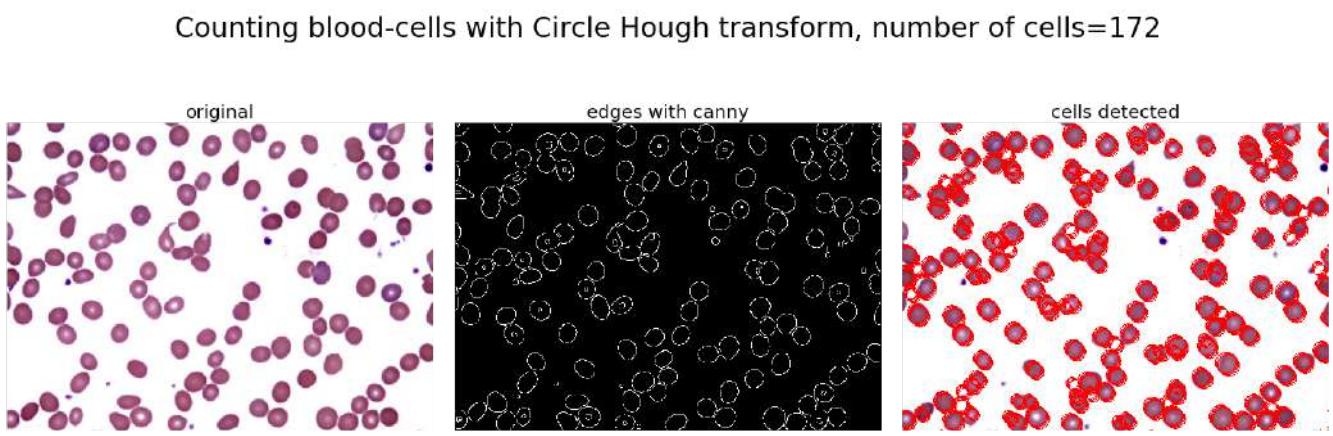
Problems

1. Object detection with Hough Transform and Colors

1.1 Counting cirular objects in an image with Circle Hough Transform

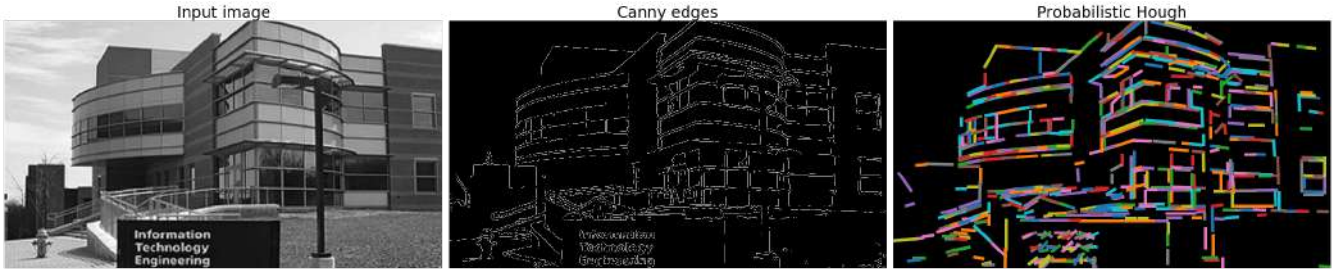
In [7]:
```
print(len(cx))

plt.figure(figsize=(20, 8))
plt.gray()
plt.subplots_adjust(0,0,1,0.975,0.05,0.05)
plt.subplot(131), plt.imshow(orig), plt.axis('off'), plt.title('original', size=20)
plt.subplot(132), plt.imshow(edges), plt.axis('off'), plt.title('edges with canny', size
plt.subplot(133), plt.imshow(image), plt.axis('off'), plt.title('cells detected', size=2
plt.suptitle('Counting blood-cells with Circle Hough transform, number of cells={}'.form
plt.show()
```

```
172
```



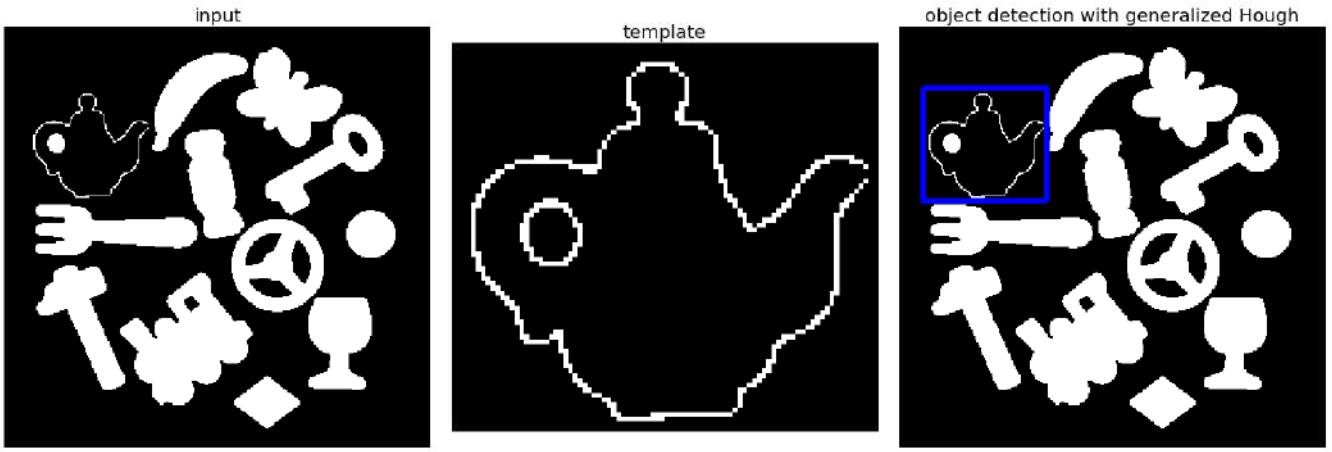
1.2 Detecting lines with Progressive Probabilistic Hough Transform

In [10]:
```
fig, axes = plt.subplots(1, 3, figsize=(30, 20), sharex=True, sharey=True)
ax = axes.ravel()
plt.gray()
ax[0].imshow(image, cmap=plt.cm.gray)
ax[0].set_title('Input image', size=25)
ax[1].imshow(edges, cmap=plt.cm.gray)
ax[1].set_title('Canny edges', size=25)
ax[2].imshow(edges * 0)
for line in lines:
 p0, p1 = line
 ax[2].plot((p0[0], p1[0]), (p0[1], p1[1]), linewidth=5)
ax[2].set_xlim((0, image.shape[1]))
ax[2].set_ylim((image.shape[0], 0))
ax[2].set_title('Probabilistic Hough', size=25)
for a in ax:
 a.set_axis_off()
plt.axis('off')
plt.tight_layout()
plt.show()
```



2.3 Detecting Objects of arbitrary shapes using Generalized Hough Transform

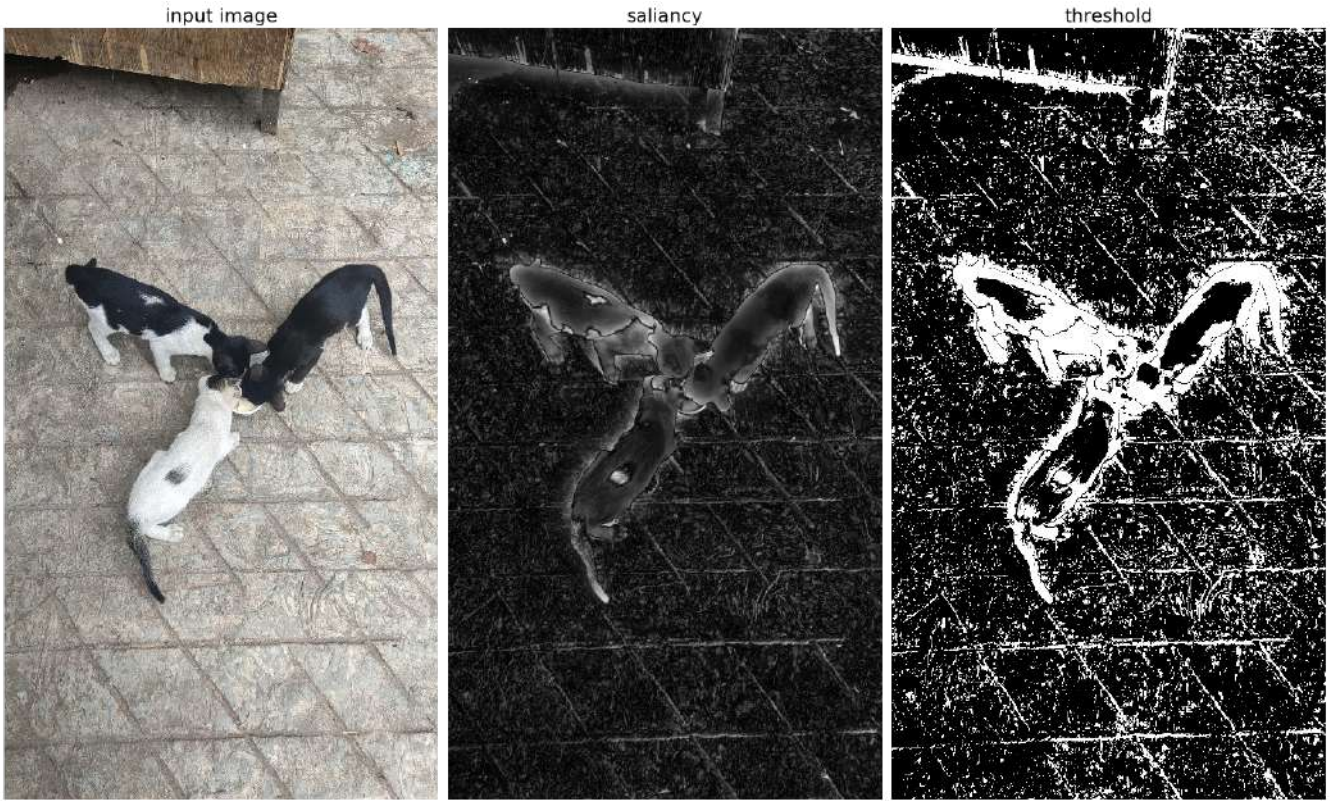
In [17]:
```
plt.figure(figsize=(20, 8))
plt.gray()
plt.subplots_adjust(0,0,1,0.975,0.05,0.05)
plt.subplot(131), plt.imshow(img), plt.axis('off'), plt.title('input', size=20)
plt.subplot(132), plt.imshow(templ), plt.axis('off'), plt.title('template', size=20)
plt.subplot(133), plt.imshow(clone), plt.axis('off'), plt.title('object detection with g
plt.show()
```



2. Object Saliency Map, Depth Map and Tone Map (HDR) with opencv-python

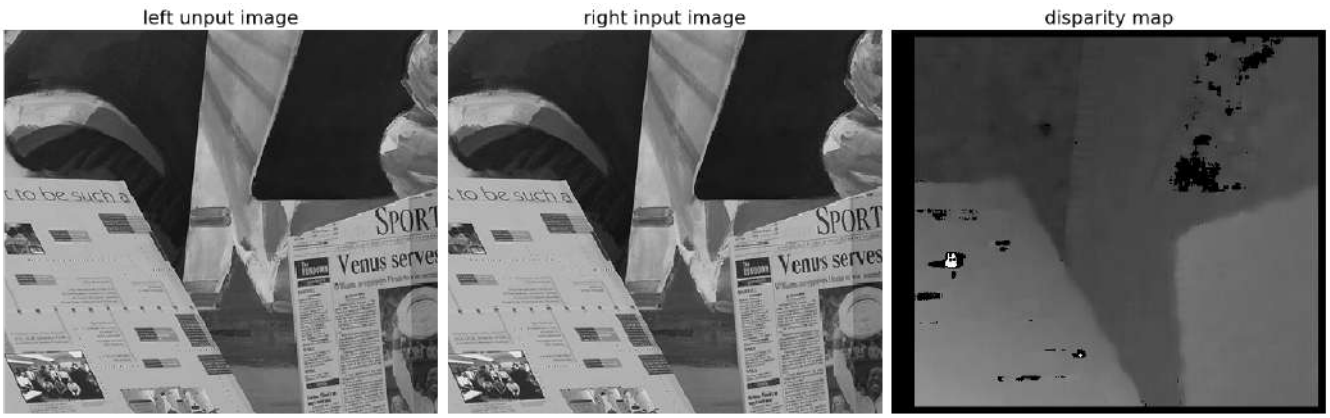
2.1 Creating Object Saliency Map

In [28]:
```
# show the images
plt.figure(figsize=(20,20))
plt.gray()
plt.subplot(131), plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB)), plt.axis('off'),
plt.subplot(132), plt.imshow(saliency_map), plt.axis('off'), plt.title('saliancy', size=
plt.subplot(133), plt.imshow(thresh_map), plt.axis('off'), plt.title('threshold', size=2
plt.tight_layout()
plt.show()
```



2.2 Creating Depth-Map from Stereo images

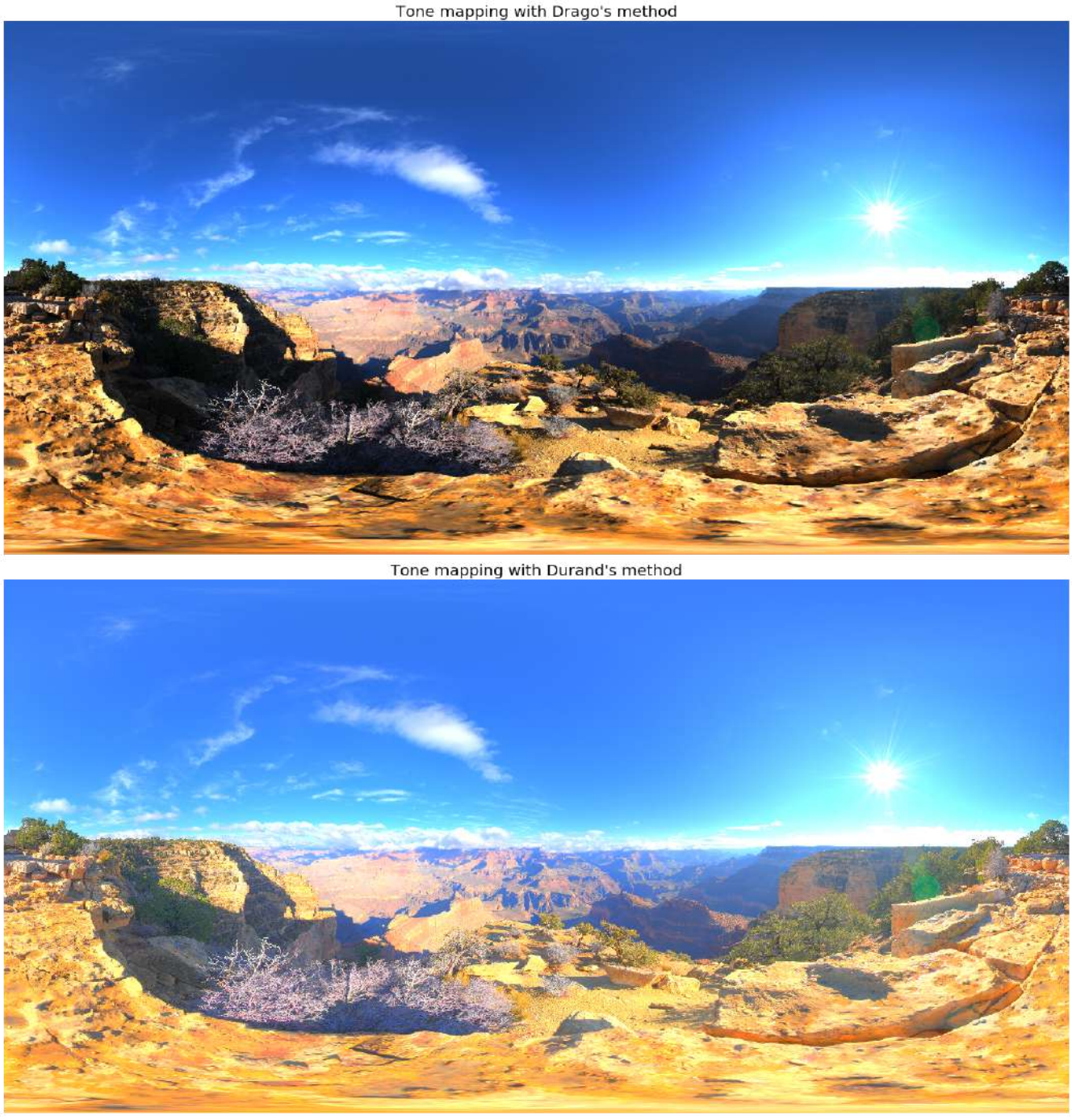
In [34]:
```
plt.figure(figsize=(20,20))
plt.gray()
plt.subplot(131), plt.imshow(img_left), plt.axis('off'), plt.title('left unput image',
plt.subplot(132), plt.imshow(img_right), plt.axis('off'), plt.title('right input image'
plt.subplot(133), plt.imshow(disparity), plt.axis('off'), plt.title('disparity map', siz
plt.tight_layout()
plt.show()
```



2.3 Tone mapping and High Dynamic Range (HDR) Imaging

In [6]:
```
plt.figure(figsize=(20,20))
plt.subplot(211), plt.imshow(ldr_drago), plt.axis('off'), plt.title('Tone mapping with D
plt.subplot(212), plt.imshow(ldr_durand), plt.axis('off'), plt.title('Tone mapping with
plt.tight_layout()
plt.show()
```

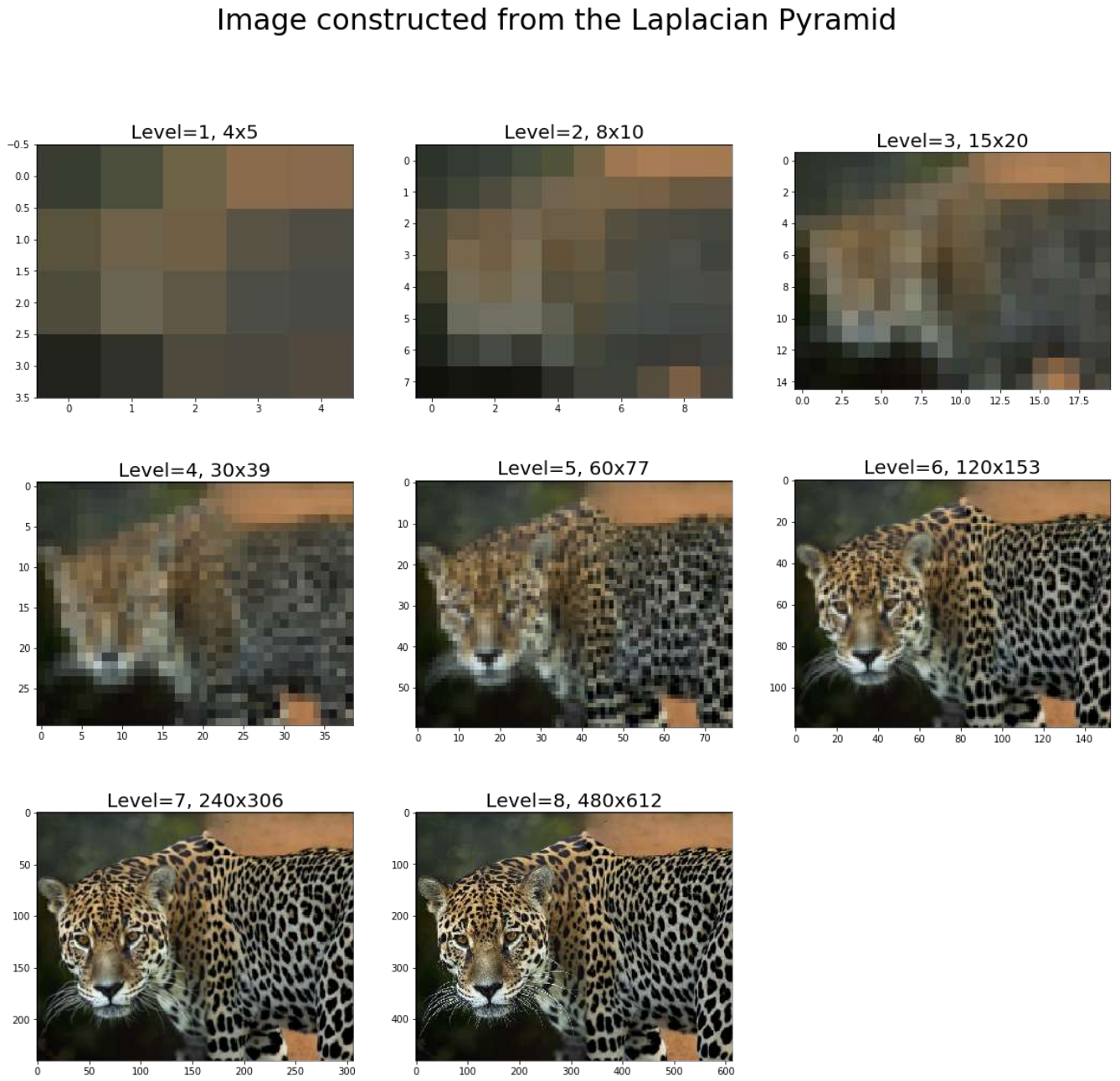
```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
```



3. Pyramid Blending

In [34]:
```
im = reconstruct_image_from_laplacian_pyramid(pyramidC)

plt.figure(figsize=(20,10))
plt.imshow(im), plt.axis('off'), plt.title('Blended output image with Pyramid', size=20)
plt.show()
```

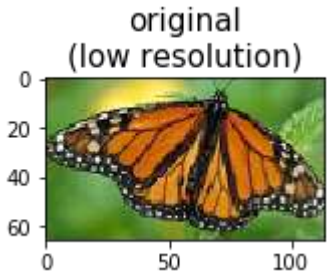
```
(4, 5, 3) (4, 5, 3)
(8, 10, 3) (8, 10, 3)
(15, 20, 3) (15, 20, 3)
(30, 39, 3) (30, 39, 3)
(60, 77, 3) (60, 77, 3)
(120, 153, 3) (120, 153, 3)
(240, 306, 3) (240, 306, 3)
(480, 612, 3) (480, 612, 3)
```

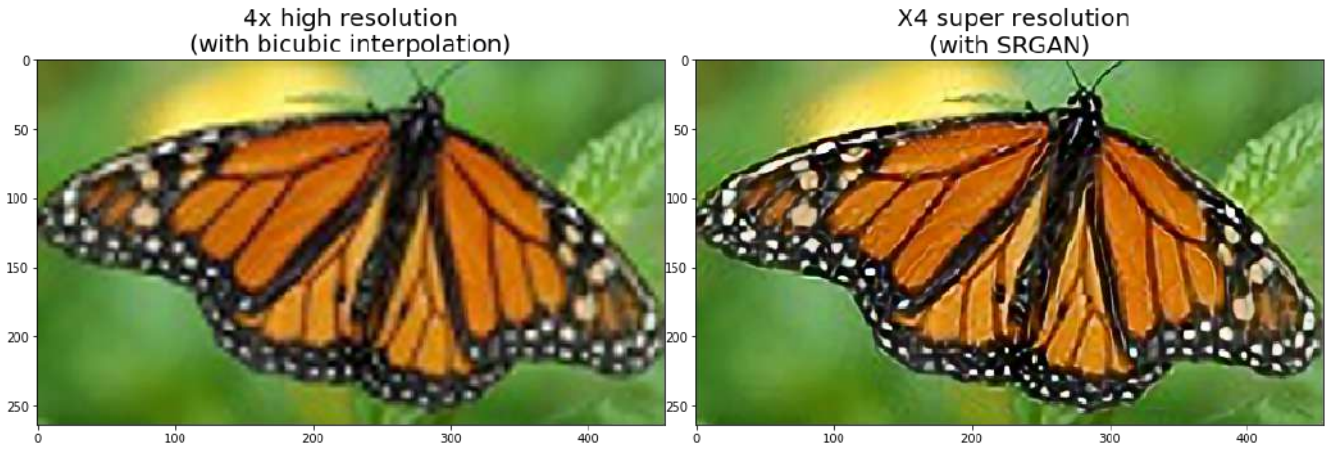


Blended output image with Pyramid

4. Image Super Resolution with deep learning model (SRGAN)

In [47]:
```
plt.figure(figsize=(2.5, 1.5))
plt.imshow(lr), plt.title('original\n(low resolution)', size=15)
plt.show()
plt.figure(figsize=(15, 9))
plt.subplots_adjust(0,0,1,1,0.05,0.05)
images = [sr, gan_sr]
titles = ['4x high resolution\n(with bicubic interpolation)', ' X4 super resolution\n(wi
positions = [1, 2]
for i, (img, title, pos) in enumerate(zip(images, titles, positions)):
    plt.subplot(1, 2, pos)
    plt.imshow(img)
    plt.title(title, size=20)
plt.show()
```





5. Low-light Image Enhancement Using CNNs

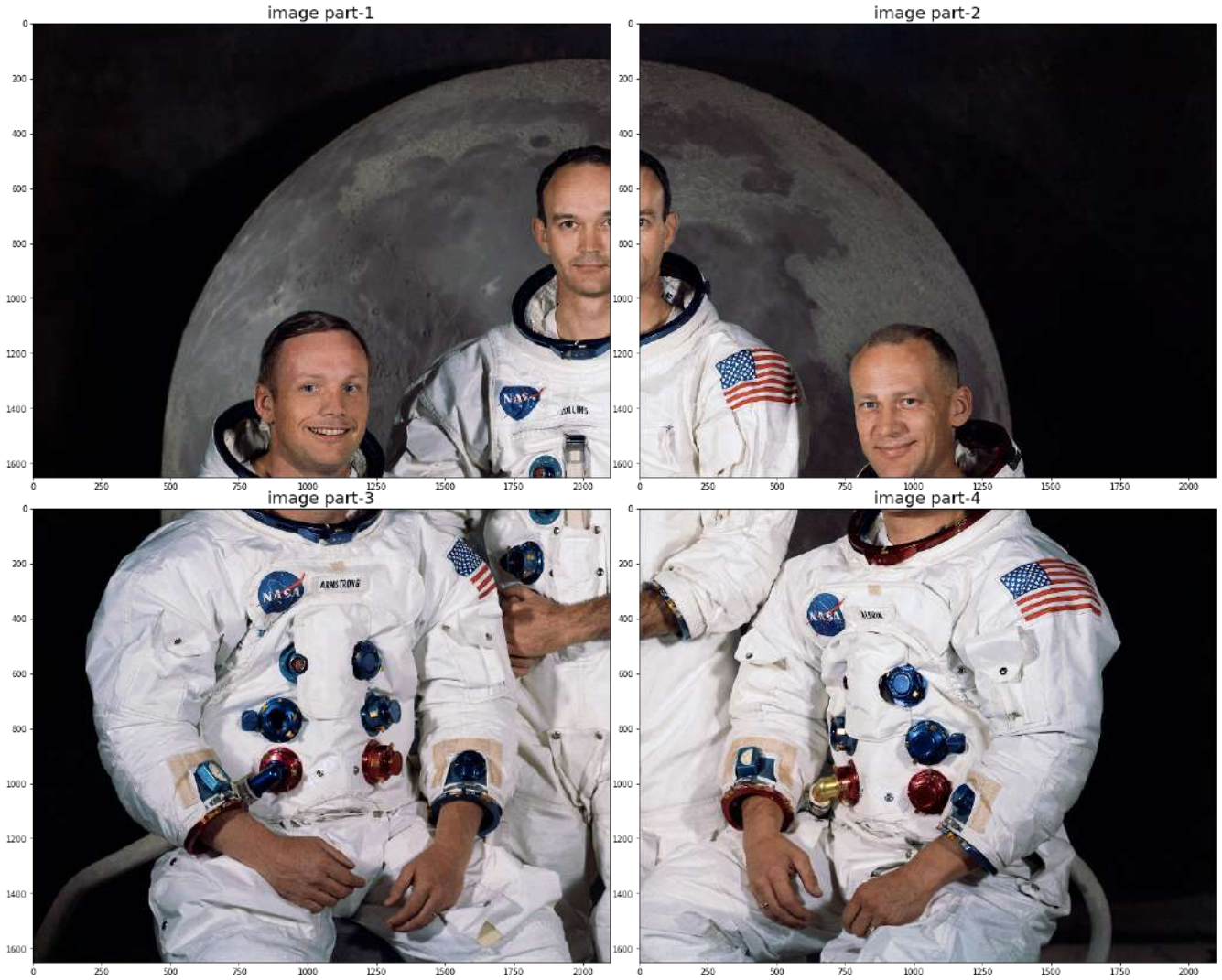
In [54]:
```
def plot_image(image, title=None, sz=20):
    plt.imshow(image)
    plt.title(title, size=sz)
    plt.axis('off')

plt.figure(figsize=(20,10))
plt.subplot(121), plot_image(img, 'low-light input')
plt.subplot(122), plot_image(np.clip(out, 0, 1), 'enhanced output')
plt.tight_layout()
plt.show()
```

6. Realistic Image Dehazing using deep neural net

In [66]:
```
plt.figure(figsize=(20,10))
plt.subplot(121), plot_image(im, 'hazed input')
plt.subplot(122), plot_image(out, 'de-hazed output')
plt.tight_layout()
plt.show()
```

7. Distributed Image Processing with Dask

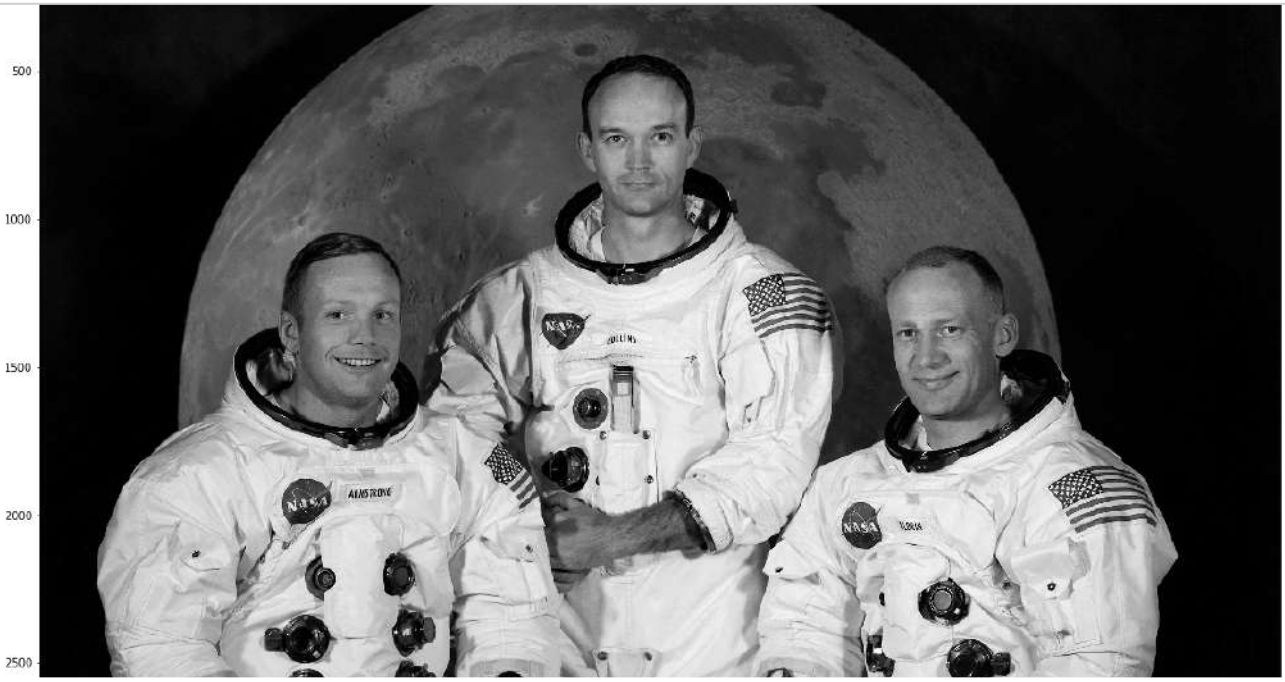
In [69]:
```
imgfile = 'images/Img_06_22.png'
partion_image(imgfile)
```



In [71]:
```
result  = (rgb2gray(partitioned_images))
print(result.shape)
plot_image(result[0])
```

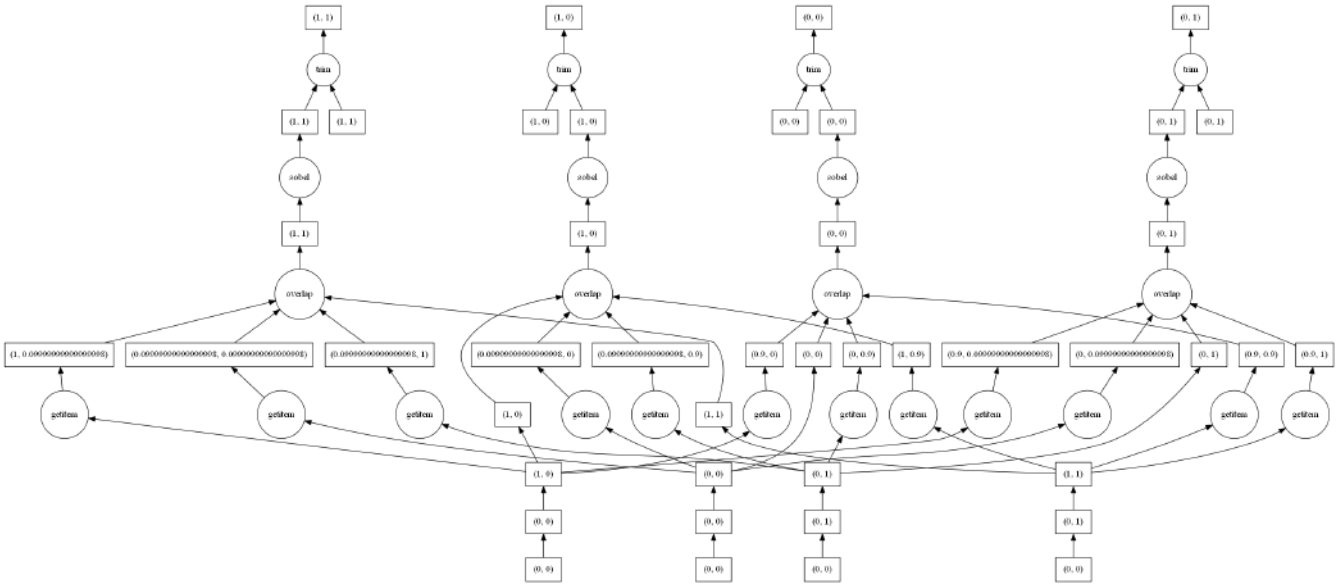
```
(4, 1650, 2100)
```

In [72]:
```
data = [result[i, ...] for i in range(result.shape[0])]
data = [data[i:i+2] for i in range(0, len(data), 2)]
combined_image = dask.array.block(data)
print(combined_image.shape)
plot_image(combined_image)
```



In [73]:
```
edges = dask_image.ndfilters.sobel(combined_image)
print(edges)
display(edges.visualize())
```

```
dask.array<_trim, shape=(3300, 4200), dtype=float64, chunksize=(1650, 2100), chunktype
=numpy.ndarray>
```



In [74]:
```
edges = np.clip(edges, 0, 1)
plot_image(edges)
```