In [39]:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
```

In [40]:

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

In [41]:

```
img.shape
```

Out[41]:

```
(168, 300, 3)
```

In [42]:

```
plt.imshow(img)
```

Out[42]:

```
<matplotlib.image.AxesImage at 0x23c75f006a0>
```

Calcul Manuel de l'histogramme

dans cette partie, nous réalisons un code de calcul manuel de l'histogramme avec des boucles imbriquées

In [43]:

```
histogram=np.zeros((3, 256))
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        for k in range(img.shape[2]):
            lineIndex=img[i,j,k]
            histogram[k,lineIndex]=histogram[k,lineIndex]+1
```

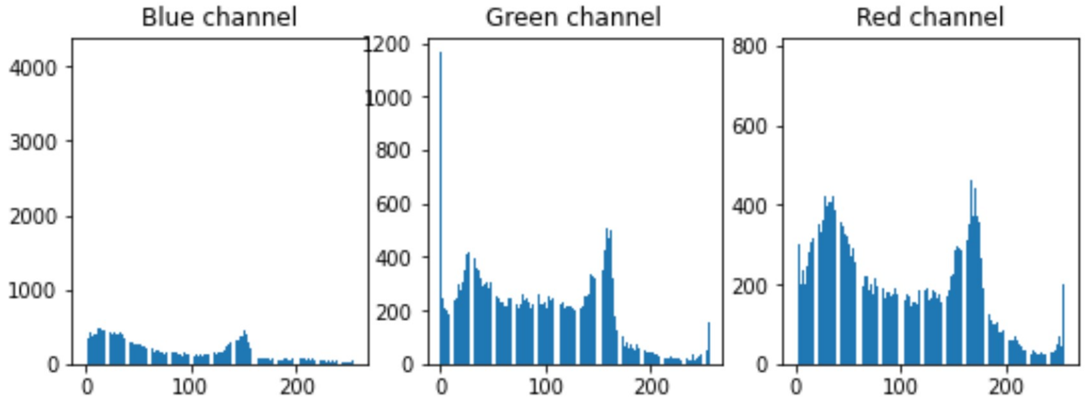
Affichage de l'histogramme

In [6]:

```
pixels = list(range(256))
values1_1 = histogram[0,:]
values2_1 = histogram[1,:]
values3_1 = histogram[2,:]

plt.figure(figsize=(9, 3))

plt.subplot(131)
plt.bar(pixels, values1_1)
plt.title('Blue channel')
plt.subplot(132)
plt.bar(pixels, values2_1)
plt.title('Green channel')
plt.subplot(133)
plt.bar(pixels, values3_1)
plt.title('Red channel')
plt.show()
```



Calcul de l'histogramme en utilisant opencv

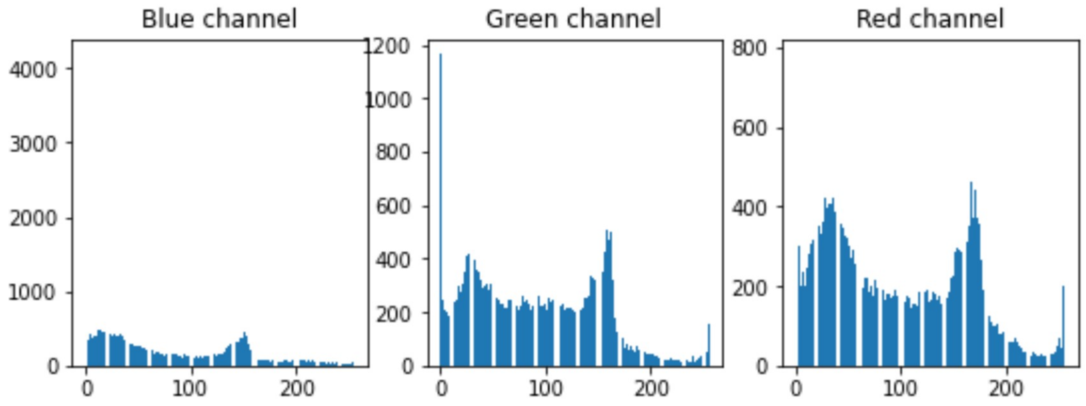
dans cette partie, nous faisons appel à opencv pour le calcul de l'histogramme

In [8]:

```
pixels = list(range(256))
hist = cv2.calcHist([img[:,:,0]],[0],None,[256],[0,256])
values1_2 = hist[:,0]
hist = cv2.calcHist([img[:,:,1]],[0],None,[256],[0,256])
values2_2 = hist[:,0]
hist = cv2.calcHist([img[:,:,2]],[0],None,[256],[0,256])
values3_2 = hist[:,0]

plt.figure(figsize=(9, 3))


plt.subplot(131)
plt.bar(pixels, values1_2)
plt.title('Blue channel')
plt.subplot(132)
plt.bar(pixels, values2_2)
plt.title('Green channel')
plt.subplot(133)
plt.bar(pixels, values3_2)
plt.title('Red channel')
plt.show()
```

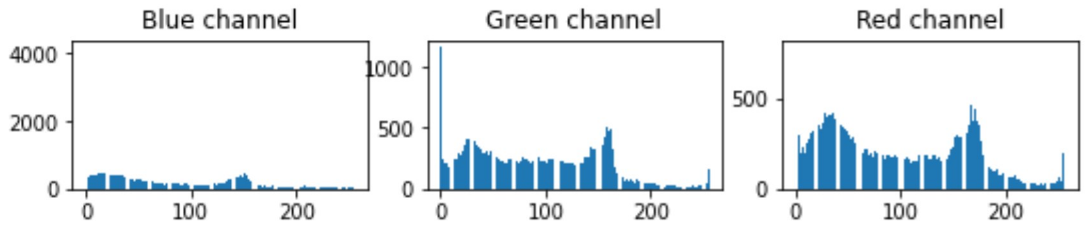


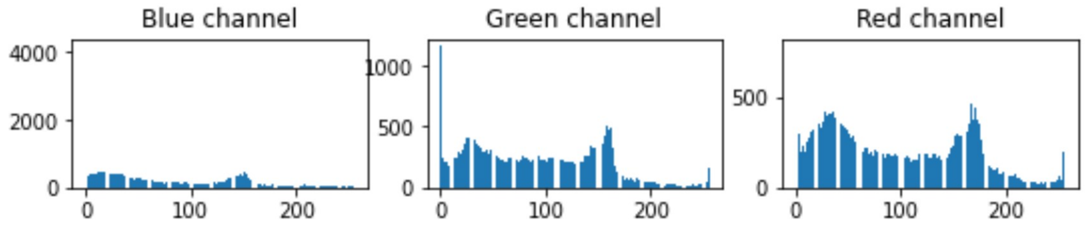
Comparaisons entre les deux histogrammes (manuel et en utilisant opencv)

In [9]:

```
plt.figure(figsize=(9, 3))

plt.subplot(231)
plt.bar(pixels, values1_1)
plt.title('Blue channel')
plt.subplot(232)
plt.bar(pixels, values2_1)
plt.title('Green channel')
plt.subplot(233)
plt.bar(pixels, values3_1)
plt.title('Red channel')
plt.figure(figsize=(9, 3))
plt.subplot(234)
plt.bar(pixels, values1_2)
plt.title('Blue channel')
plt.subplot(235)
plt.bar(pixels, values2_2)
plt.title('Green channel')
plt.subplot(236)
plt.bar(pixels, values3_2)
plt.title('Red channel')
plt.show()
```



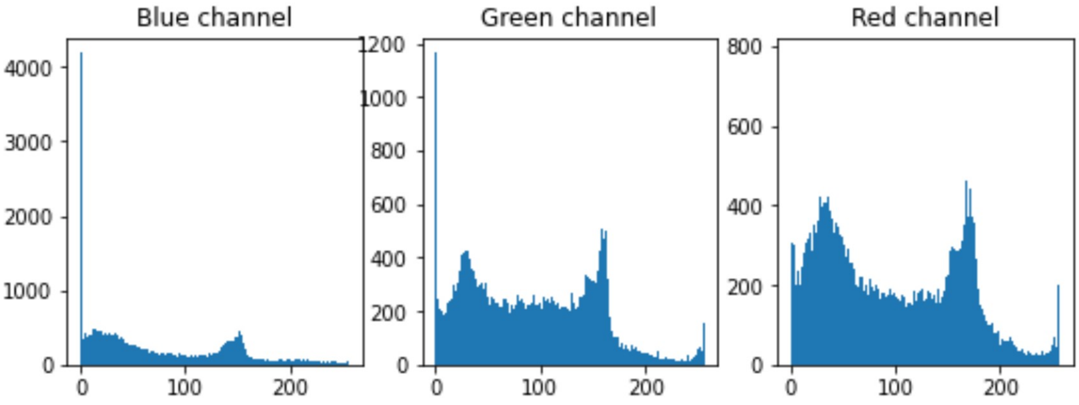


Affichage de l'histogramme en utilisant la

fonction plt.hist

In [10]:

```
plt.figure(figsize=(9, 3))
plt.subplot(131)
plt.hist(img[:,:,0].ravel(),256,[0,256]);
plt.title('Blue channel')
plt.subplot(132)
plt.hist(img[:,:,1].ravel(),256,[0,256]);
plt.title('Green channel')
plt.subplot(133)
plt.hist(img[:,:,2].ravel(),256,[0,256]);
plt.title('Red channel')
plt.show()
```



Explorer d'autres fonctions de calcul

In [51]:

```
hist,bins = np.histogram(img.ravel(),256,[0,256])
```

In [52]:

```
hist = cv2.calcHist([img[:,:,0]],[0],None,[256],[0,256])
```

Difference entre le output de la methode manuelle et la methode d'opencv

In [47]:

```
hist[0:10]
```

Out[47]:

```
array([[4186.],
       [ 273.],
       [ 345.],
       [ 329.],
       [ 419.],
       [ 420.],
       [ 381.],
       [ 478.],
       [ 399.],
       [ 435.]], dtype=float32)
```

In [48]:

```
histogram[0,0:10]
```

Out[48]:

```
array([4186.,  273.,  345.,  329.,  419.,  420.,  381.,  478.,  399.,
        435.])
```

Histogramme cumulé

histogramm cumulé valeurs

In [15]:

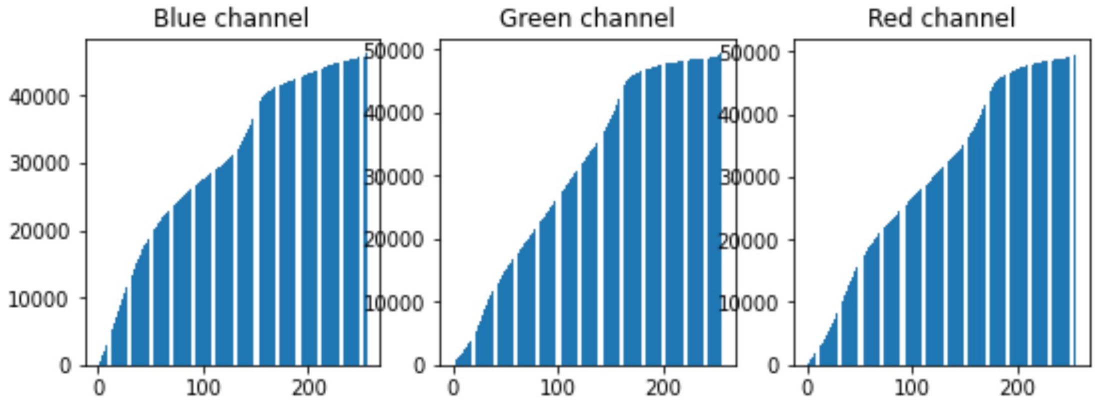
```
histogramC=np.zeros((3, 256))
histogramC[0:2,1]=histogram[0:2,1]
for i in range(histogram.shape[0]):
    for j in range(histogram.shape[1]-1):
        histogramC[i,j+1]=histogram[i,j+1]+histogramC[i,j]
```

In [16]:

```
pixels = list(range(256))
values1_3 = histogramC[0,:]
values2_3 = histogramC[1,:]
values3_3 = histogramC[2,:]

plt.figure(figsize=(9, 3))

plt.subplot(131)
plt.bar(pixels, values1_3)
plt.title('Blue channel')
plt.subplot(132)
plt.bar(pixels, values2_3)
plt.title('Green channel')
plt.subplot(133)
plt.bar(pixels, values3_3)
plt.title('Red channel')
plt.show()
```



Histogramme cumulé pourcentage

In [17]:

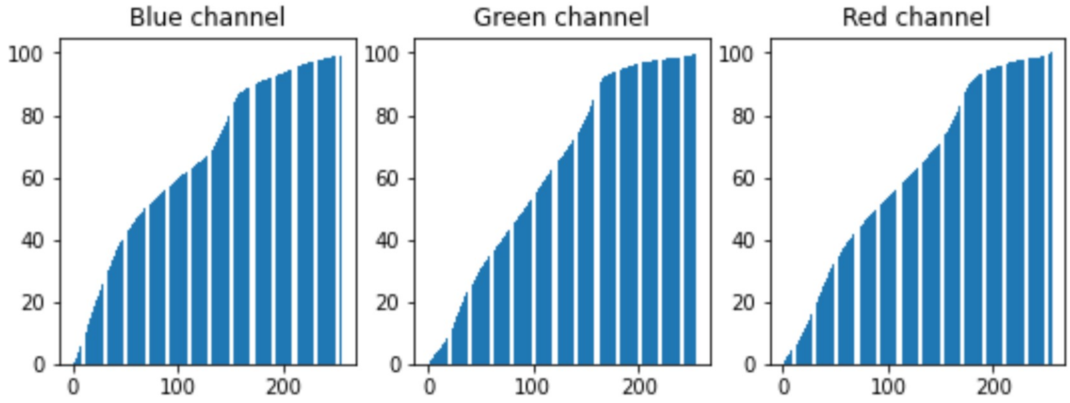
```
histogramCP=np.zeros((3, 256))
for i in range(histogramC.shape[0]):
    maxHist=histogramC[i,histogramC.shape[1]-1]
    for j in range(histogram.shape[1]):
        histogramCP[i,j]=histogramC[i,j]/maxHist*100
```

In [18]:

```
pixels = list(range(256))
values1_4 = histogramCP[0,:]
values2_4 = histogramCP[1,:]
values3_4 = histogramCP[2,:]

plt.figure(figsize=(9, 3))

plt.subplot(131)
plt.bar(pixels, values1_4)
plt.title('Blue channel')
plt.subplot(132)
plt.bar(pixels, values2_4)
plt.title('Green channel')
plt.subplot(133)
plt.bar(pixels, values3_4)
plt.title('Red channel')
plt.show()
```



Egalisation d'histogramme

Utiliser le niveau de gros de l'image

In [19]:

```
histogramG=np.zeros((256))
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
for i in range(gray.shape[0]):
    for j in range(gray.shape[1]):
        lineIndex=gray[i,j]
        histogramG[lineIndex]=histogramG[lineIndex]+1
```

In [20]:

```
pixels = list(range(256))
plt.bar(pixels, histogramG)
```

Out[20]:

```
<BarContainer object of 256 artists>
```

```
In [21]: gray.min()
```

```
Out[21]: 0
```

```
In [22]: gray.max()
```

```
Out[22]: 255
```

```
In [24]: img = cv2.imread('car.jpg',0)
         equ = cv2.equalizeHist(img)
```

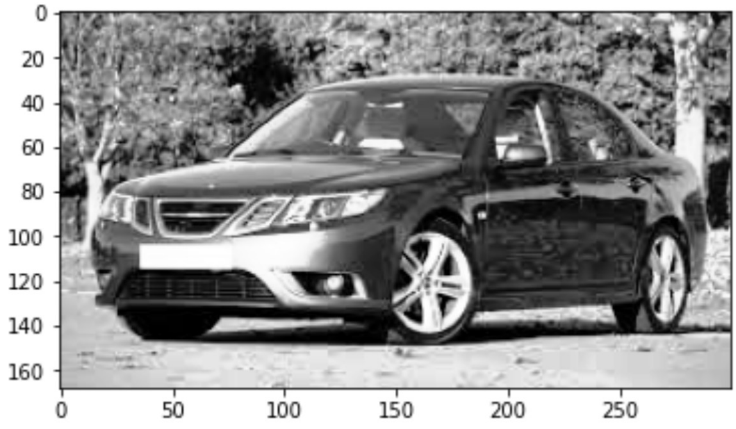
```
In [25]: # stacking images side-by-side
         res = np.hstack((img, equ))
```

```
In [28]: equ.shape
```

```
Out[28]: (168, 300)
```

```
In [35]: plt.imshow(equ, cmap='gray')
```

```
Out[35]: <matplotlib.image.AxesImage at 0x23c76cea4c0>
```



```
In [ ]:
```