Студент гр. РИМ-181226 Бабикова Евгения Витальевна

---

2. Устройство изображения. Работа с изображениями в Python

Импорт необходимых библиотек и модулей.

In [59]:

```
from skimage.io import imsave, imread, imshow
from skimage import img_as_float,img_as_ubyte
import numpy as np
%matplotlib inline
```

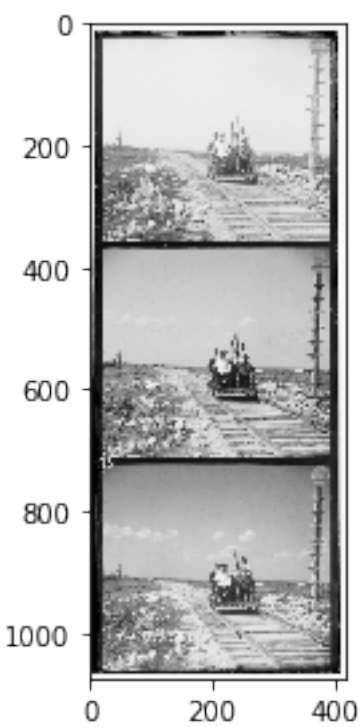
2.1. Цифровое изображение. Чтение, запись, работа с пикселями

Количество столбцов

Считывание и просмотр изображения.

In [2]:

```
img = imread('img.png')
imshow(img);
```



Определения кол-ва столбцов.

In [3]:

```
img.shape[1]
```

Out[3]:

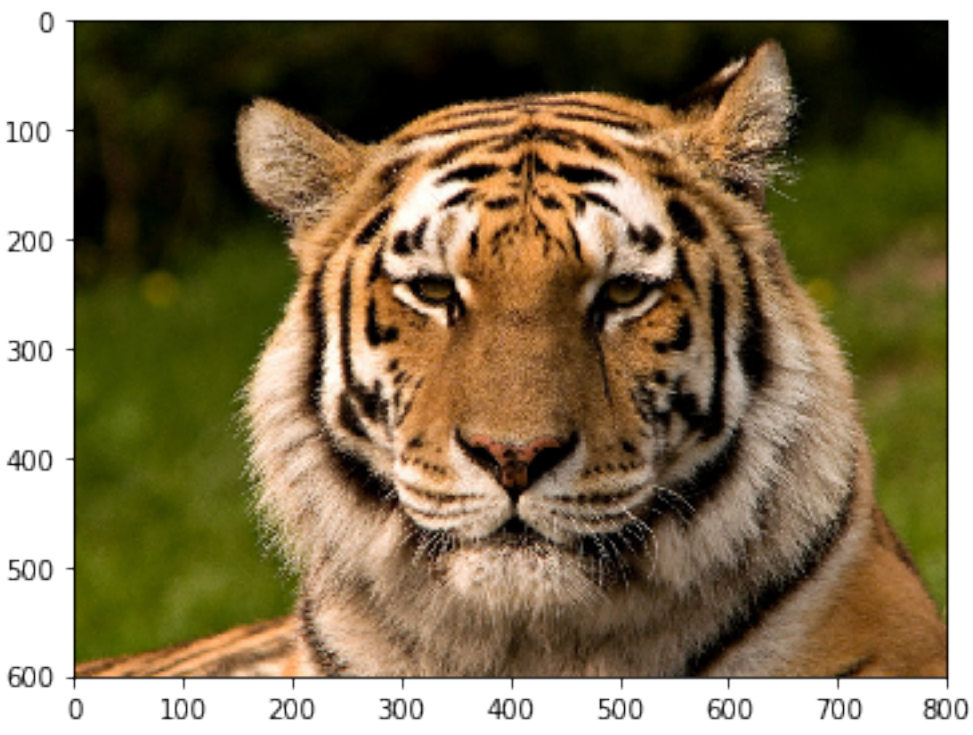
```
419
```

Изменение цвета пикселя изображения

Считывание и просмотр изображения.

In [35]:

```
img = imread('tiger-color.png')
imshow(img);
```



Определение координат центрального пикселя и смена его цвета на зеленый.

In [36]:

```
x = img.shape[1]//2
y = img.shape[0]//2
```

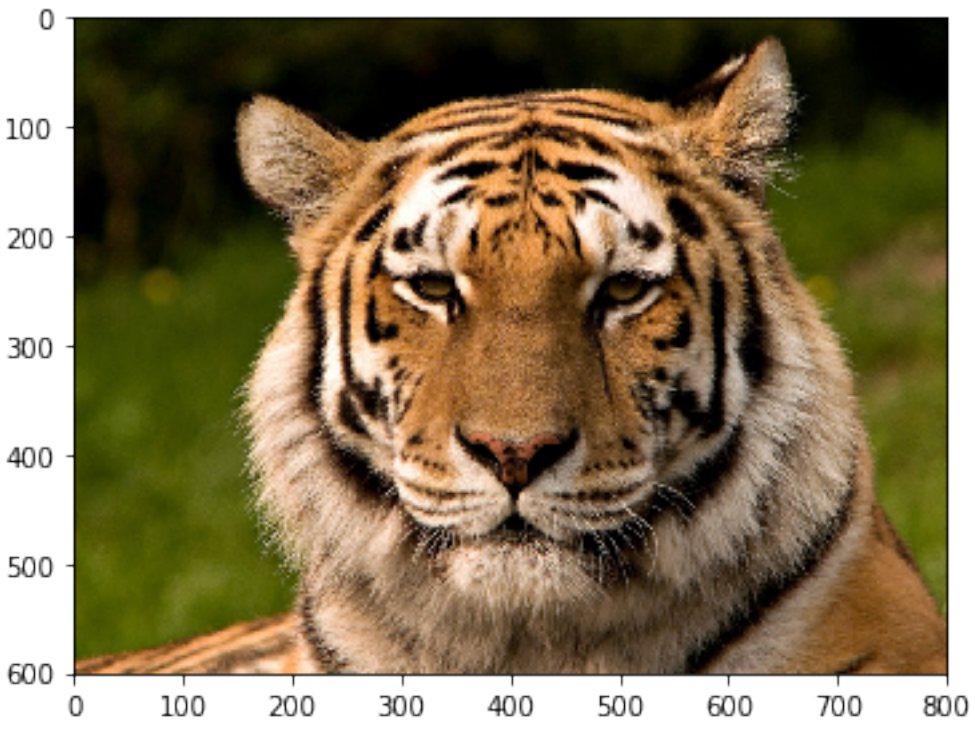
In [37]:

```
img[y,x]=[102,204,102]
```

Просмотр изображения с измененным пикселем.

In [39]:

```
imshow(img);
```



Сравнение полученного изображения и образца.

In [40]:

```
img_sample=imread('tiger-color-green-pixel.png')
np.array_equal(img, img_sample)
```

Out[40]:

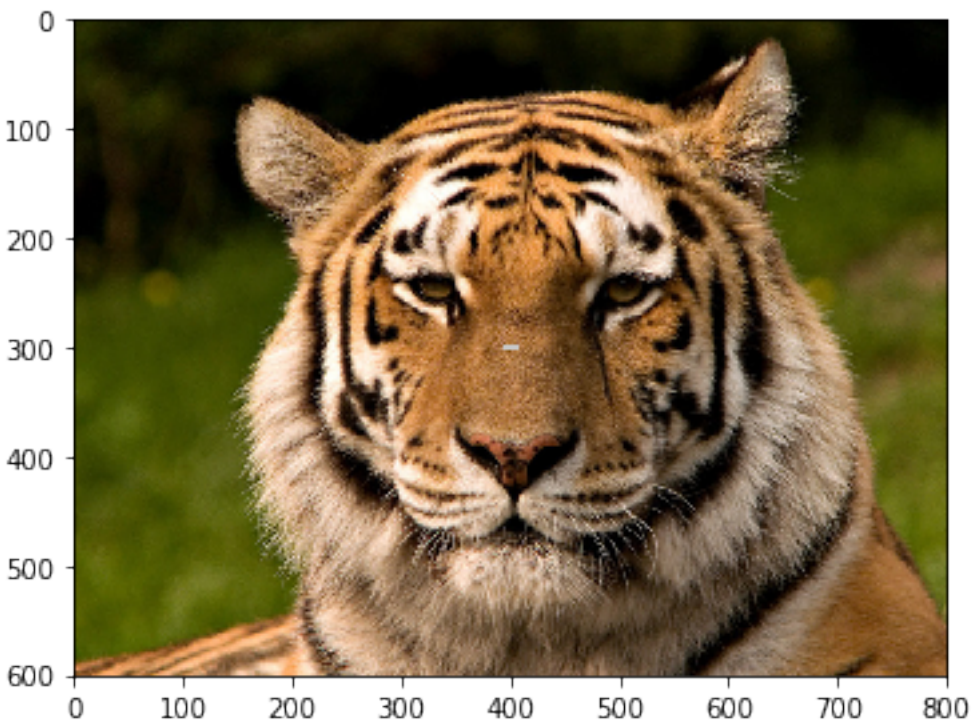
```
True
```

Изменение цвета прямоугольника

Считывание и просмотр изображения с серым прямоугольником размером 7*15 в центре.

In [42]:

```
img = imread('tiger-gray.png')
imshow(img);
```



Вычисление по определенным в предыдущем задании координатам центра координат прямоугольника и смена его цвета на розовый.

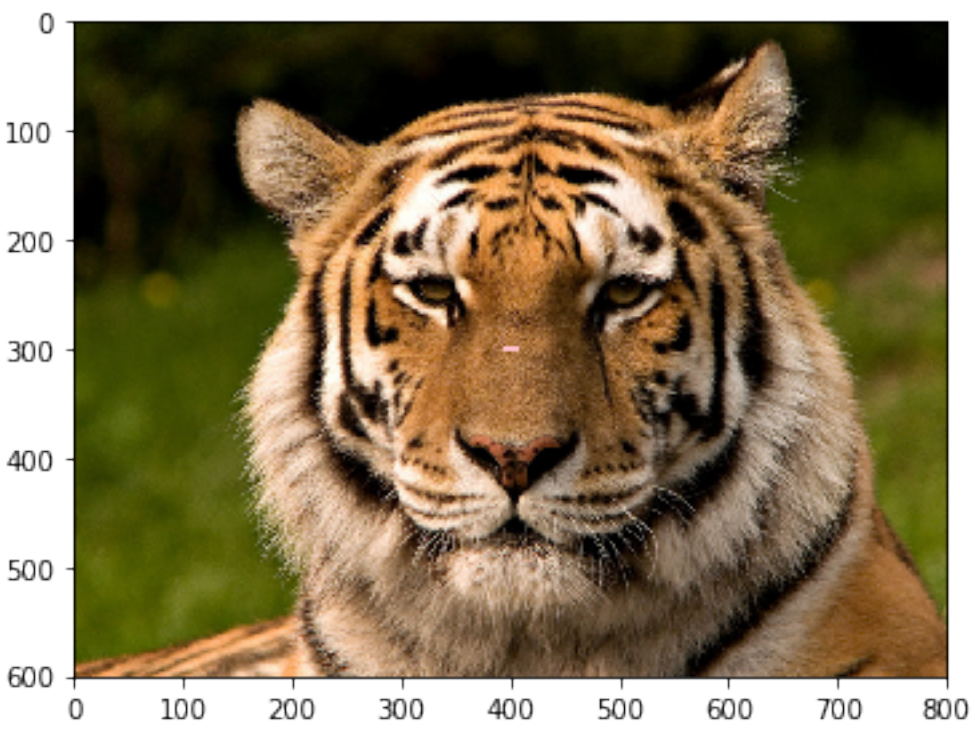
In [43]:

```
img[y-3:y+4,x-7:x+8]=[255,192,203]
```

Просмотр результата смены цвета прямоугольника.

In [44]:

```
imshow(img);
```



Сравнение полученного изображения и образца.

In [45]:

```
img_sample=imread('tiger-pink.png')
np.array_equal(img, img_sample)
```

Out[45]:

```
True
```

Определение рамки изображения

Считывание и просмотр изображения с рамкой.

In [11]:

```
img = imread('tiger-border.png')
imshow(img);
```

Определение **цвета** рамки по левому верхнему пикселю.

In [12]:

```
border_color=img[0,0]
```

Определение размера изображения и его центра.

In [13]:

```
y_size=img.shape[0]
x_size=img.shape[1]
central_y=y_size//2
central_x=x_size//2
```

Поиск ширины **левой** границы: фиксация координаты по y и перебор по оси x в прямом направлении. Сохранение в перменную `left` пикселей, соответствующих яркости пикселя из границы (определено выше). Т.о. `left` содержит кол-во пикселей в левой границе - ширина границы.

In [14]:

```
left=0
for i in range(x_size):
    if(all(border_color!=img[central_y,i])):
        break
    left+=1
```

Поиск ширины **верхней** границы: фиксация координаты по x и перебор по оси y в прямом направлении.

In [15]:

```
top=0
for i in range(y_size):
    if(all(border_color!=img[i,central_x])):
        break
    top+=1
```

Поиск ширины **правой** границы: фиксация координаты по y и перебор по оси x в обратном направлении.

In [16]:

```
right=0
for i in reversed(range(x_size)):
    if(all(border_color!=img[central_y,i])):
        break
    right+=1
```

Поиск ширины **нижней** границы: фиксация координаты по x и перебор по оси y в обратном направлении.

In [17]:

```
bottom=0
for i in reversed(range(y_size)):
    if(all(border_color!=img[i,central_x])):
        break
    bottom+=1
```

In [18]:

```
left, top, right, bottom
```

Out[18]:

```
(1, 39, 25, 7)
```

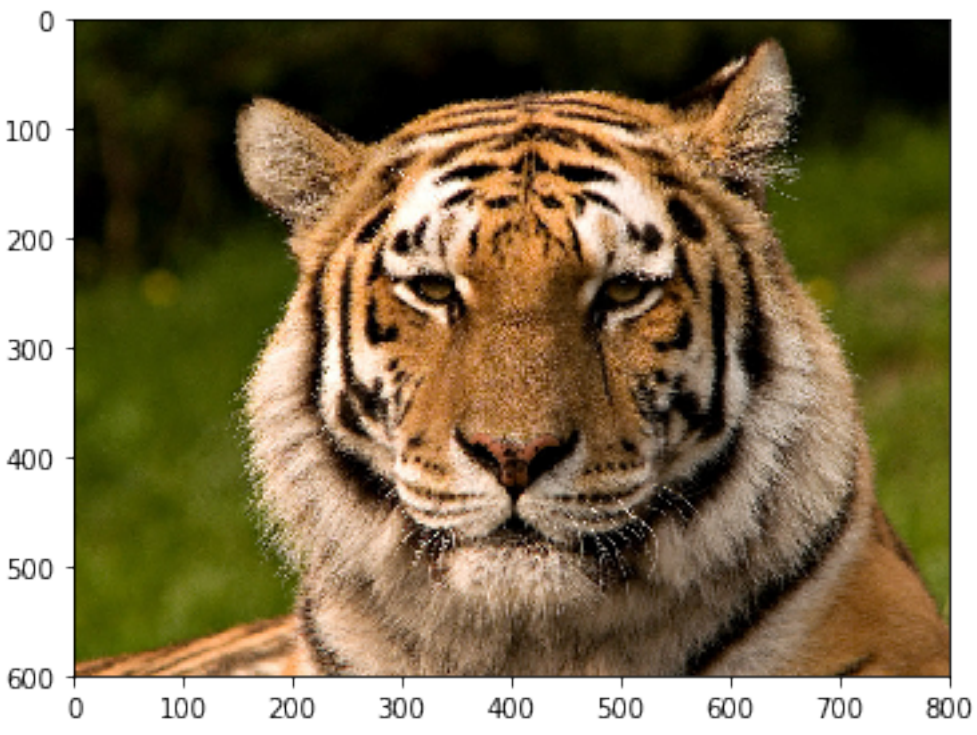
2.2. Арифметические операции. Каналы. Вычисление яркости изображения

Вычисление негатива изображения

Считывание и просмотр исходного изображения.

In [46]:

```
img = imread('tiger-color1.png')
imshow(img);
```



Получение негатива из исходного изображения и сравнение полученного изображения с образцом.

In [47]:

```
img_sample=imread('tiger-negative.png')
np.array_equal(255-img, img_sample)
```

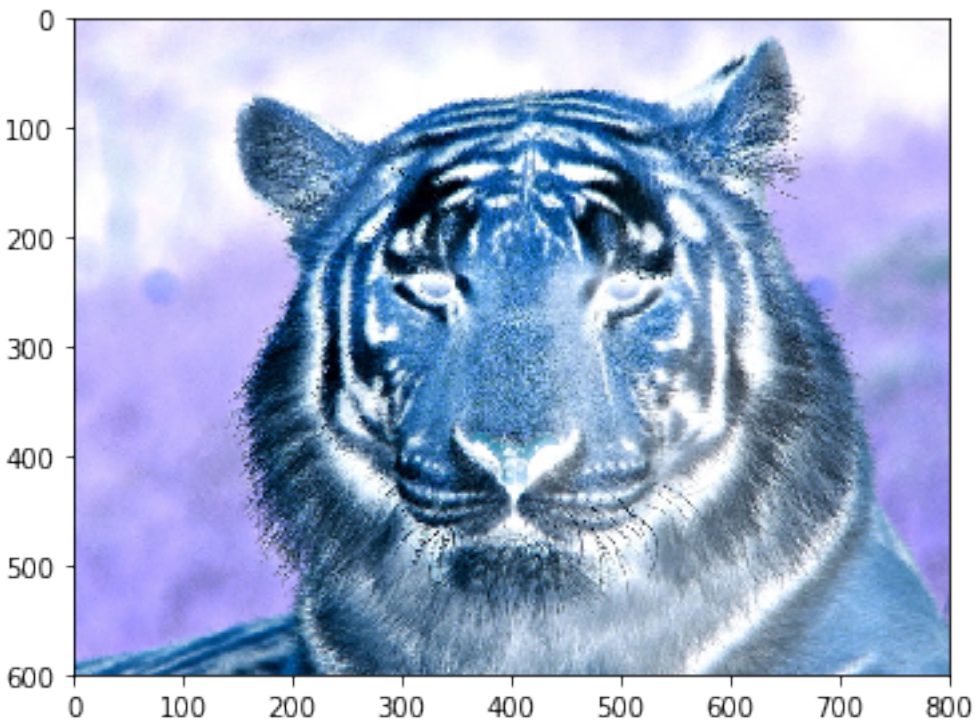
Out[47]:

```
True
```

Просмотр изображения тигра в негативе.

In [49]:

```
imshow(255-img);
```



Поменять местами каналы изображения

Выделение каналов в исходном изображении `tiger-color.png`

In [50]:

```
R = img[:,:,0]
G = img[:,:,1]
B = img[:,:,2]
```

Смена порядка каналов с RGB на BRG.

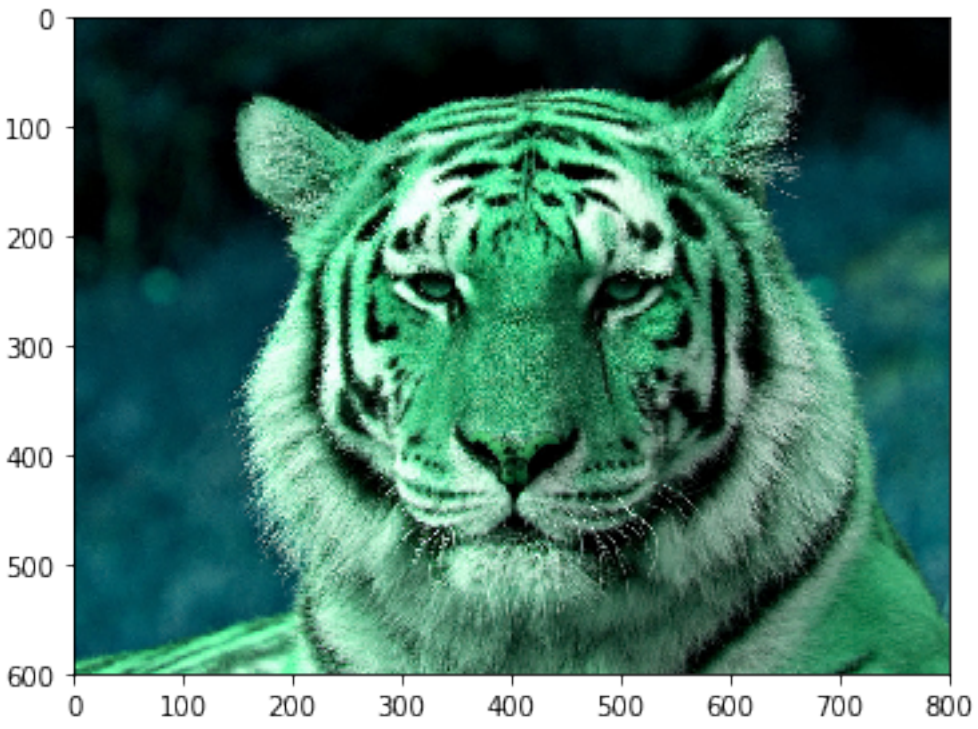
In [51]:

```
img_out=np.dstack((B,R,G))
```

Просмотр результата смены порядка каналов изображения.

In [52]:

```
imshow(img_out);
```



Сравнение полученного изображения с образцом.

In [53]:

```
img_sample=imread('tiger-brg.png')
np.array_equal(img_out, img_sample)
```

Out[53]:

```
True
```

Подсчет яркости изображения

Перевод исходного изображения в числа с плавающей запятой.

In [24]:

```
img_f=img_as_float(img)
```

Вычисление яркости.

In [25]:

```
Y=0.2126*img_f[:,:,0]+0.7152*img_f[:,:,1]+0.0722*img_f[:,:,2]
```

Приведение одноканального изображения к типу `ubyte` .

In [26]:

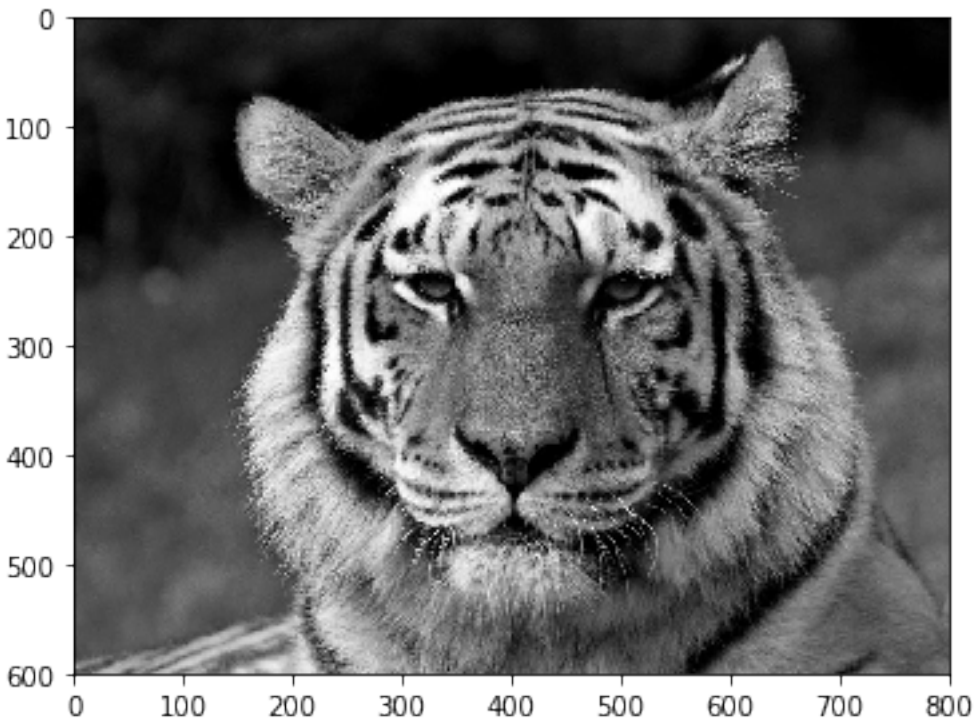
```
img_gray=img_as_ubyte(Y)
```

```
/anaconda3/lib/python3.6/site-packages/skimage/util/dtype.py:141:
UserWarning: Possible precision loss when converting from float64
to uint8
  .format(dtypeobj_in, dtypeobj_out))
```

Просмотр одноканального изображения.

In [54]:

```
imshow(img_gray);
```



Сравнение полученного одноканального изображения с образцом.

In [55]:

```
img_sample=imread('tiger-y.png')
np.array_equal(img_gray, img_sample)
```

Out[55]:

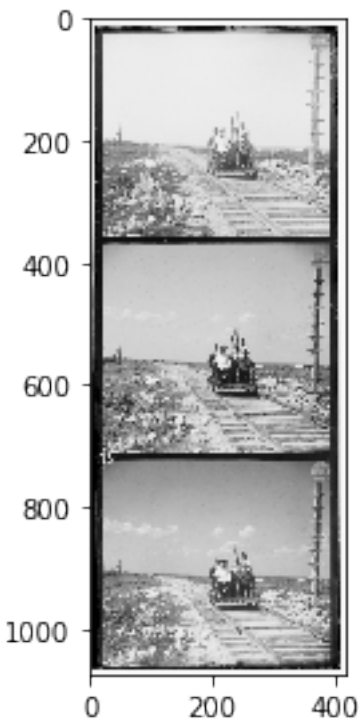
```
True
```

2.3. Сопоставление фотографий Прокудина-Горского

Считывание и просмотр фотографии Прокудина-Горского.

In [28]:

```
img=imread('00.png')
imshow(img);
```



Перевод исходного изображения в числа с плавающей запятой.

In [29]:

```
img=img_as_float(img)
```

Разделение изображения на три части по высоте и обрезка краев по 6,5% по ширине (т.к. меньшее значение оставляет край пленки).

In [30]:

```
part = int(img.shape[0]/3)
cut = int(img.shape[1]*0.065)
# Обрезка каналов
B = img[cut:part-cut, cut:img.shape[1] - cut]
G = img[part + cut:2*part - cut, cut:img.shape[1] - cut]
R = img[2 * part+cut:3*part - cut, cut:img.shape[1] - cut]
```

Фунция для сдвига каналов. Из задания – значение сдвига 15.

In [31]:

```
def shift(channel1,channel2):
    corr_result = 0 # лучшее значение корреляции каналов
    shift_y_result = 0 # лучший сдвиг канала по высоте
    shift_x_result = 0 # лучший сдвиг канала по длине
    # Сдвиг по высоте
    for y in range(-15, 15):
        shift_y = np.roll(channel1, y, 0)
        # Сдвиг по длине
        for x in range(-15, 15):
            shift_x = np.roll(shift_y, x, 1)
            # Определение "похожести" (корреляции) каналов
            corr = (shift_x * channel2).sum()
            # Поиск лучших значений
            if corr > corr_result:
                corr_result = corr
                shift_y_result = y
                shift_x_result = x
    # Сдвиг на определенное в функции значение по высоте, а затем по длине
    return np.roll(np.roll(channel1, shift_y_result, 0), shift_x_result, 1)
```

Сдвиги R и B каналов относительно фиксированного G.

In [32]:

```
shift_r = shift(R, G)
shift_b = shift(B, G)
```

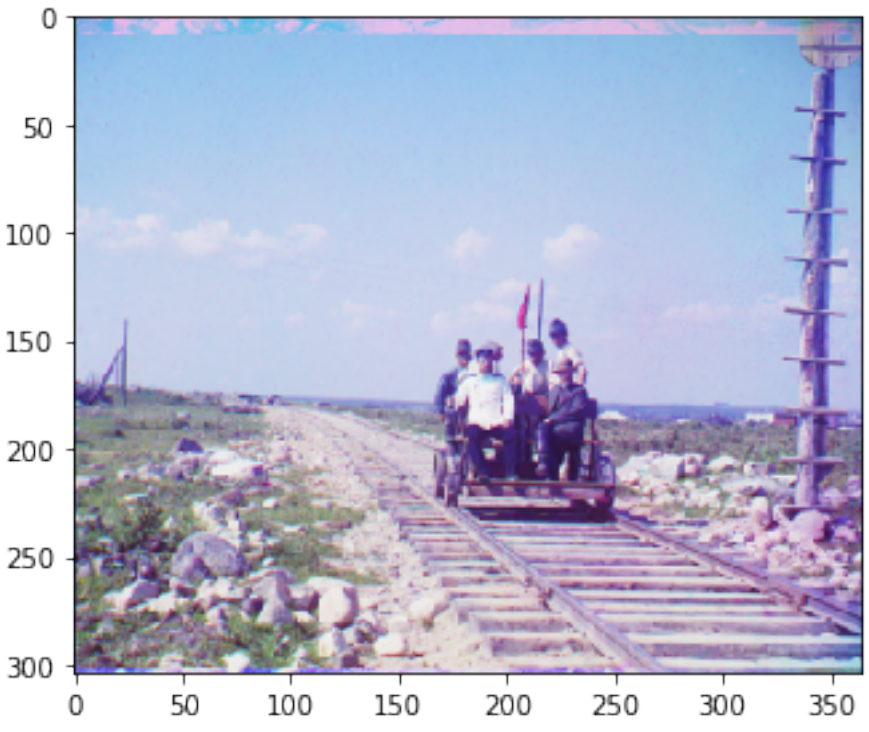
Совмещение каналов с наилучшими сдвигами.

In [33]:

```
img_out=np.dstack((shift_r, G, shift_b))
```

In [34]:

```
imshow(img_out);
```



In [ ]:

```

```