In [14]:

```
import requests
from bs4 import BeautifulSoup
import os
from IPython.display import clear_output
import pandas as pd
```

In [ ]:

```
def getimgcount(path):
    path, dirs, files = next(os.walk(path))
    file_count = len(files)
    return file_count

def getallimg(webpage, savepath):

    r = requests.get(webpage)
    data = r.text
    soup = BeautifulSoup(data, "lxml")

    for link in soup.find_all('img'):
        image = link.get('data-srcset')
        if image is not None:
            url = str(image).split(',')[1][1:-2]
            image_file = requests.get(url)
            image_name = str(link.parent.parent.parent.get('data-id'))+'.jpg'
            #print('Downloading '+image_name, end='\r')
            count = getimgcount(savepath)
            print(str(count)+' images downloaded.', end='\r')
            with open(savepath+image_name, "wb") as f:
                f.write(image_file.content)
```

In [ ]:

```
webpage = "https://www.lomography.com/films/871954447-lomography-lomochrome-pu
rple-xr-100-400-35mm/photos?order=popular&page="
savepath = "./images2/"

index = 105
while getimgcount(savepath) < 5000:
    clear_output()
    print('Downloading all images from page '+str(index)+'...', end='\n')
    getallimg(webpage+str(index), savepath)
    index += 1
```

In [ ]:

```
r = requests.get(webpage)
print(webpage)
data = r.text
soup = BeautifulSoup(data, "lxml")

for link in soup.find_all('img'):
    image = link.get('data-srcset')
    if image is not None:
        url = str(image).split(',')[1][1:-2]
        image_file = requests.get(url)
        print()
```

In [ ]:

```
import cv2
import numpy as np

# read image into matrix.
m =  cv2.imread("./images/18695759.jpg")

# get image properties.
h,w,bpp = np.shape(m)

# print image properties.
print("width: " + str(w))
print("height: " + str(h))
print("bpp: " + str(bpp))
```

In [ ]:

```
all_width = []
all_height = []
path = './images/'
for filename in os.listdir(path):
    image = cv2.imread(path+filename)
    h,w,bpp = np.shape(image)
    all_width.append(w)
    all_height.append(h)
```

In [ ]:

```
dim = pd.DataFrame(all_width, all_height, columns=['width', 'height'])
```

In [ ]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter(all_width, all_height)
plt.show()
```

In [ ]:

```
plt.hist2d(all_width, all_height, bins=20)
plt.colorbar()
plt.show()
```

In [ ]:

```
np.stat(all_width)
```

In [ ]:

```
import pandas as pd
w = pd.DataFrame(all_width)
w.hist(bins=10)
w.describe()
```

In [ ]:

```
h = pd.DataFrame(all_height)
h.hist()
h.describe()
```

In [ ]:

```
dim[]
```

In [12]:

```
# central crop 512x768
path = './images2/'
newpath = './images-cropped/'
new_width = 768
new_height = 512
for filename in os.listdir(path):
    img_path = path + filename
    modified_path = newpath + filename
    resize_and_crop(img_path, modified_path, (new_width, new_height), crop_typ
e='middle')
```

In [ ]:

```
# resize to half size (to reduce memory usage)
path = './images-cropped/'
newpath = './images-small/'
new_width = 384
new_height = 256
print(new_height, new_width)
for filename in os.listdir(path):
    img_path = path + filename
    modified_path = newpath + filename
    resize_and_crop(img_path, modified_path, (new_width, new_height), crop_typ
e='middle')
    print('Converted ', len(os.listdir(newpath)), ' of ', len(os.listdir(path)
), ' images.', end='\r')
```

```
256 384
Converted  10086  of  10199  images.
```

In [15]:

```
# code taken from https://gist.github.com/sigilioso/2957026

from PIL import Image

def resize_and_crop(img_path, modified_path, size, crop_type='top'):
    """
    Resize and crop an image to fit the specified size.
    args:
        img_path: path for the image to resize.
        modified_path: path to store the modified image.
        size: `(width, height)` tuple.
        crop_type: can be 'top', 'middle' or 'bottom', depending on this
            value, the image will cropped getting the 'top/left', 'middle' or
            'bottom/right' of the image to fit the size.
    raises:
        Exception: if can not open the file in img_path of there is problems
            to save the image.
        ValueError: if an invalid `crop_type` is provided.
    """
    # If height is higher we resize vertically, if not we resize horizontally
    img = Image.open(img_path)
    # Get current and desired ratio for the images
    img_ratio = img.size[0] / float(img.size[1])
    ratio = size[0] / float(size[1])
    #The image is scaled/cropped vertically or horizontally depending on the r
atio
    if ratio > img_ratio:
        img = img.resize((size[0], round(size[0] * img.size[1] / img.size[0]))
,
                Image.ANTIALIAS)
        # Crop in the top, middle or bottom
        if crop_type == 'top':
            box = (0, 0, img.size[0], size[1])
        elif crop_type == 'middle':
            box = (0, round((img.size[1] - size[1]) / 2), img.size[0],
```

```
                round((img.size[1] + size[1]) / 2))

        elif crop_type == 'bottom':
            box = (0, img.size[1] - size[1], img.size[0], img.size[1])
        else :
            raise ValueError('ERROR: invalid value for crop_type')
        img = img.crop(box)
    elif ratio < img_ratio:
        img = img.resize((round(size[1] * img.size[0] / img.size[1]), size[1])
,
                Image.ANTIALIAS)
        # Crop in the top, middle or bottom
        if crop_type == 'top':
            box = (0, 0, size[0], img.size[1])
        elif crop_type == 'middle':
            box = (round((img.size[0] - size[0]) / 2), 0,
                   round((img.size[0] + size[0]) / 2), img.size[1])
        elif crop_type == 'bottom':
            box = (img.size[0] - size[0], 0, img.size[0], img.size[1])
        else :
            raise ValueError('ERROR: invalid value for crop_type')
        img = img.crop(box)
    else :
        img = img.resize((size[0], size[1]),
                Image.ANTIALIAS)
        # If the scale is the same, we do not need to crop
    img.save(modified_path)
```