Нормальный отчет без юмора шуток и т. п.

Мы часто сталкиваемся в жизни со смазанными картинками, испорченными звуковыми сигналами и т. п. Все это можно представить как операция деконволюции

Если рассматривать размытие изображений, имеет место такая формула

$$g = f \otimes h + n$$

где f – исходное изображение, h – матрица размытия (PSF), n – шум.

$$h(x,y) \otimes f(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} h(i,j) f(x+i, y \cdot$$

и x,y пробегают по всему изображению

Теорема о свертке

Есть теорема о свертке, в которой говорится, что операция свертки в пространственной области эквивалентна поэлементному умножению в частотной

$$h(x,y) \otimes f(x,y) = H(u,v) F(u,v)$$

где H(u,v) и F(u,v) -Фурье-образы соответствующих функций

Модели искажения

Всякий смаз можно свести к какой-то модели. Здесь мы рассмотрим некоторые из них

1. Гауссово размытие.

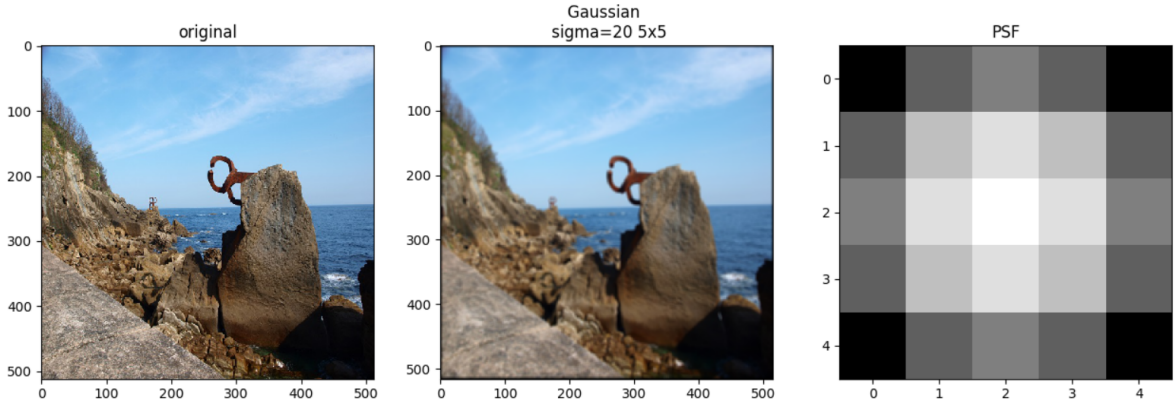
В этом случае ядро смаза (то самое h) формируется по формуле нормального распределения $N(0, \sigma^2)$

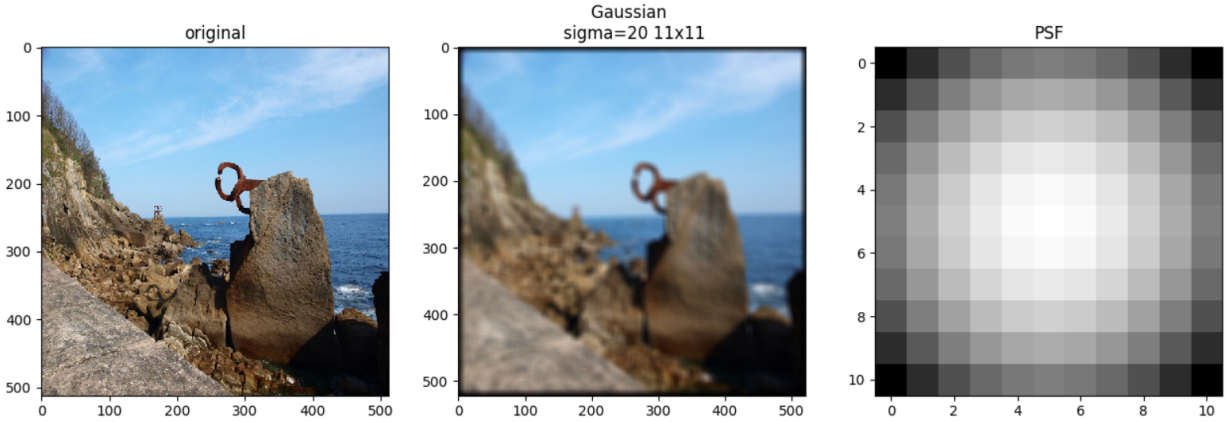
$$h(m,n) = \frac{1}{2\pi\sigma^2} \sum_{u,v} e^{\frac{-(u^2+v^2)}{2\sigma^2}}$$

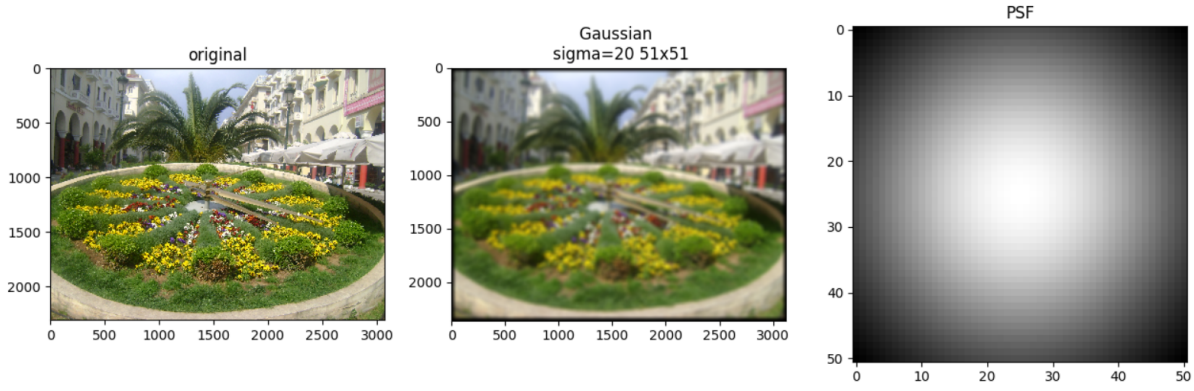
Попытка реализации на Python

```
def gauss(x,y,sigma):
    twoPi = math.pi * 2
    return (1/(twoPi*sigma*sigma))*math.exp(-(x*x+y*y)/float(2*sigma*sigma))
def gaussian(sigma,n,m):
    f=np.array([[gauss(i,j,sigma) for j in range (-(m-1)//2, (m+1)//2)] for i in
range(-(n-1)//2, (n+1)//2)])
    f = f / np.sum(f)
    return f
```

Примеры размытия:







2. Motion blur

Возникает при движении камеры или объектов на изображении (или можно даже сказать сцене)

Честно переписал с MatLab-а на Python

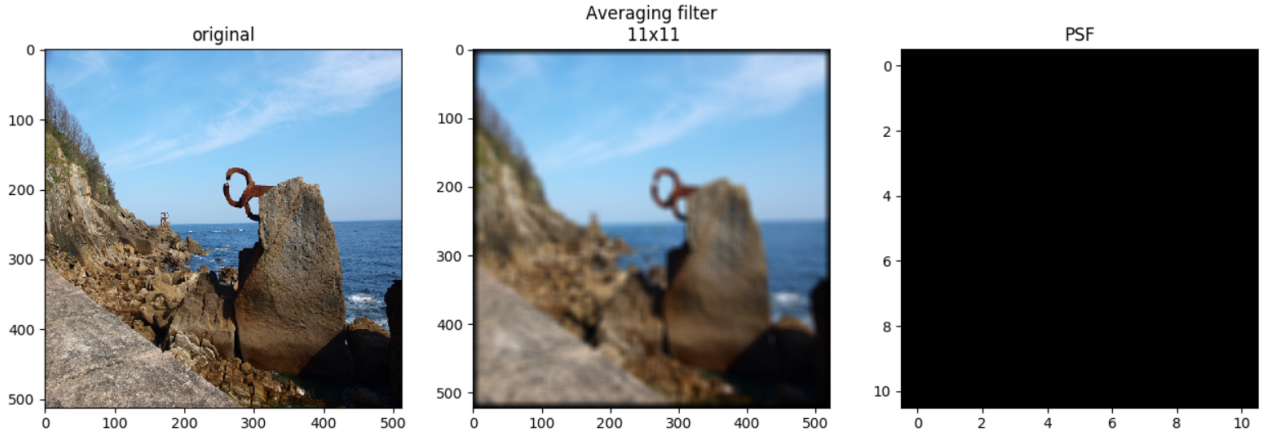
```
def motion_blur(len, ang):

    eps=2.220446049250313e-016
    len=max(1, len)
    half=(len-1)/2
    phi=(ang%180)/180.*math.pi
    cosphi=math.cos(phi)
    sinphi=math.sin(phi)
    xsign=np.sign(cosphi)
    linewdth=1
    sx=int(half*cosphi+linewdth*xsign-len*eps)
    sy=int(half*sinphi+linewdth-len*eps)
    xx=np.arange(0,sx+xsign, xsign)
    yy=np.arange(0, sy+1)
    x,y=np.meshgrid(xx,yy)
    dist2line=(y*cosphi-x*sinphi)
    rad=np.sqrt(x*x+y*y)
    indexs1=np.where(rad>=half)
    indexs2=np.where(np.fabs(dist2line)<=linewdth)
    indexs1=np.array(indexs1)
    indexs1=indexs1.transpose()
    indexs2=np.array(indexs2)
    indexs2=indexs2.transpose()
    indexs=find(indexs1,indexs2)
    x2lastpix=half-np.fabs((x[indexs[:,0], indexs[:,1]]
+dist2line[indexs[:,0], indexs[:,1]]*sinphi)/cosphi)
    dist2line[indexs[:,0], indexs[:,1]]=np.sqrt(dist2line[indexs[:,0],
indexs[:,1]]**2+x2lastpix**2)
    dist2line=linewdth+eps-abs(dist2line)
    dist2line[np.where(dist2line < 0)] = 0
    h=np.rot90(dist2line,2)
    old_h=np.copy(h)
    h_w=h.shape[0]
    h_h=h.shape[1]
    d_w=dist2line.shape[0]
    d_h=dist2line.shape[1]
    h=np.zeros((h_w+d_w-1, h_h+d_h-1), dtype=np.float64)
    h[0:h_w, 0:h_h]=old_h
    h[h_w-1: h_w+d_w-1, h_h-1: h_h+d_h-1]=dist2line
    h=h/(np.sum(h)+eps*len*len)
    #h(end + (1:end)-1, end + (1:end)-1) = dist2line;
    if cosphi>0:
        h=np.flipud(h)
    return h
```

//подогнать примеры

3. Усредняющее размытие

```
def averaging_filter(n,m):
```

```
    result=np.ones((n,m), dtype=float)
    result/=(n*m)
    return result
```

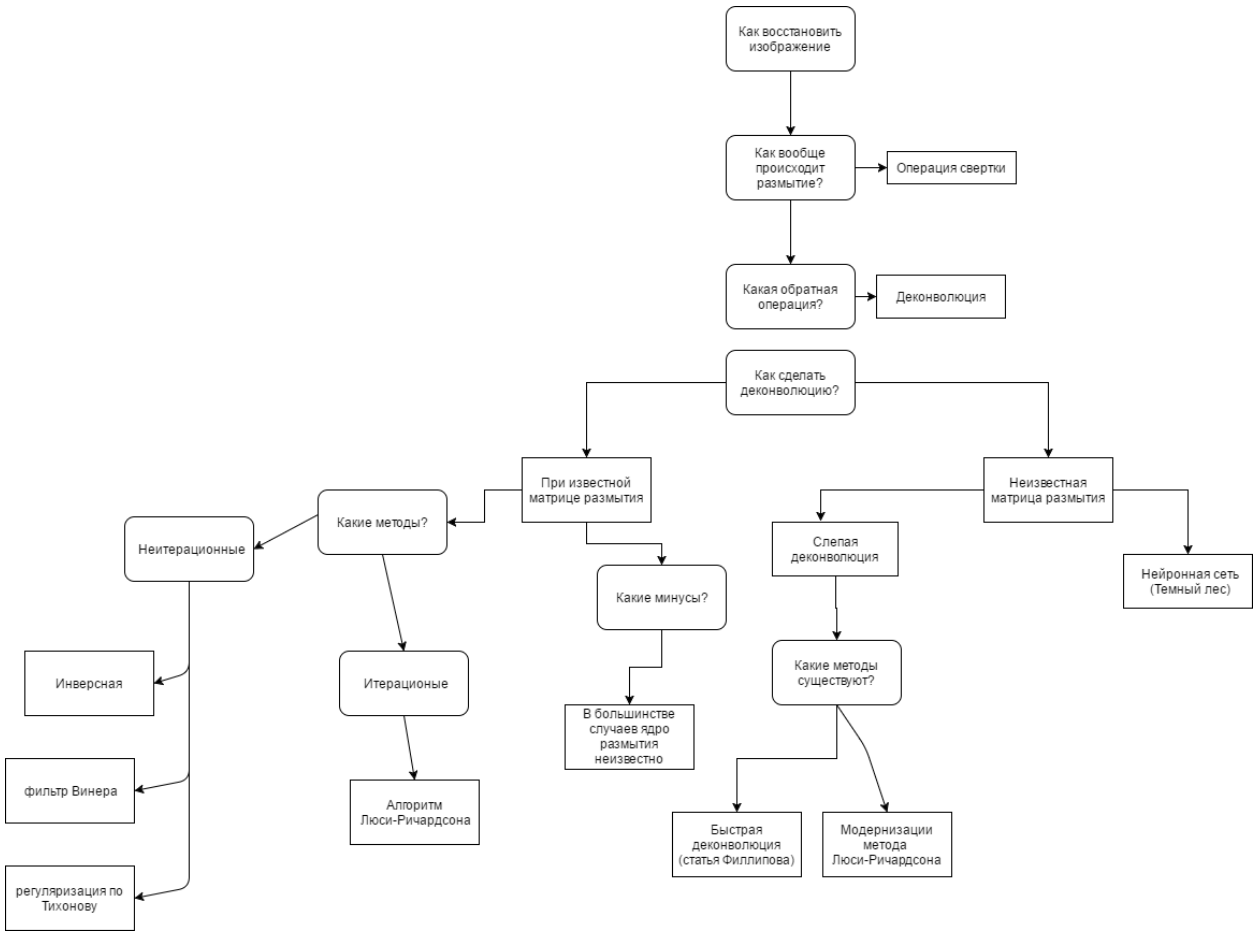


4. Рандомная матрица

```
def random_psf(n,m):

    result=np.random.rand(n,m)
    result/=np.sum(result)
    return result
```

// тоже подогнать примеры

Методы восстановления



схема

1. Самый простой — инверсный фильтр. Предполагаем, что шум незначительный или его нет и пренебрегаем им

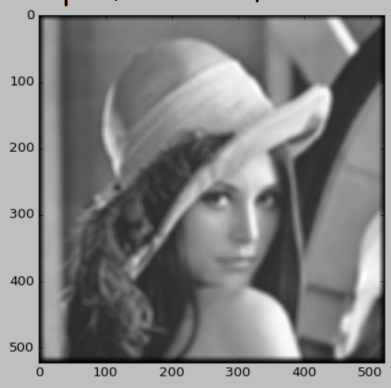
$$\hat{F}(u,v)=\frac{G(u,v)}{H(u,v)}$$

```
def inverse_filter(g,h):

    width_g=g.shape[0]
    height_g=g.shape[1]
    width_h=h.shape[0]
    height_h=h.shape[1]
    g1=np.zeros((2*width_g,2*height_g))
    h1=np.zeros((2*width_g, 2*height_g))
    g1[0:width_g,0:height_g]=g
    h1[0:width_h, 0:height_h] = h
    G=np.fft.fft2(g1)
    H=np.fft.fft2(h1)
    F=G/H
    f=np.fft.ifft2(F)
```

```
f=np.real(f)
f=f[0:width_g,0:height_g]
return f
```

Исходное изображение

Искаженное ядром Гаусса
матрица 11 на 11, сигма = 5



Восстановленное