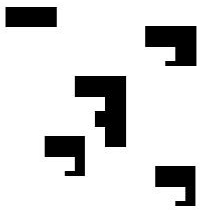
TODO *6

主要步骤

- 把原图切割，获取图形轮廓坐标并切分好存储，
- 对图中每一个连续图形依次进行OPC，
    - 对于一个图形，对切分好的线段，分别计算移动 - MAX ~ MAX的EPE值（MAX为设定的最大的可移动的量），选取最小EPE值对应的处理方法，下次移动线段在此方法处理图像上进行应用。
- 至所有线段都处理过后得到最终处理完成的结果。

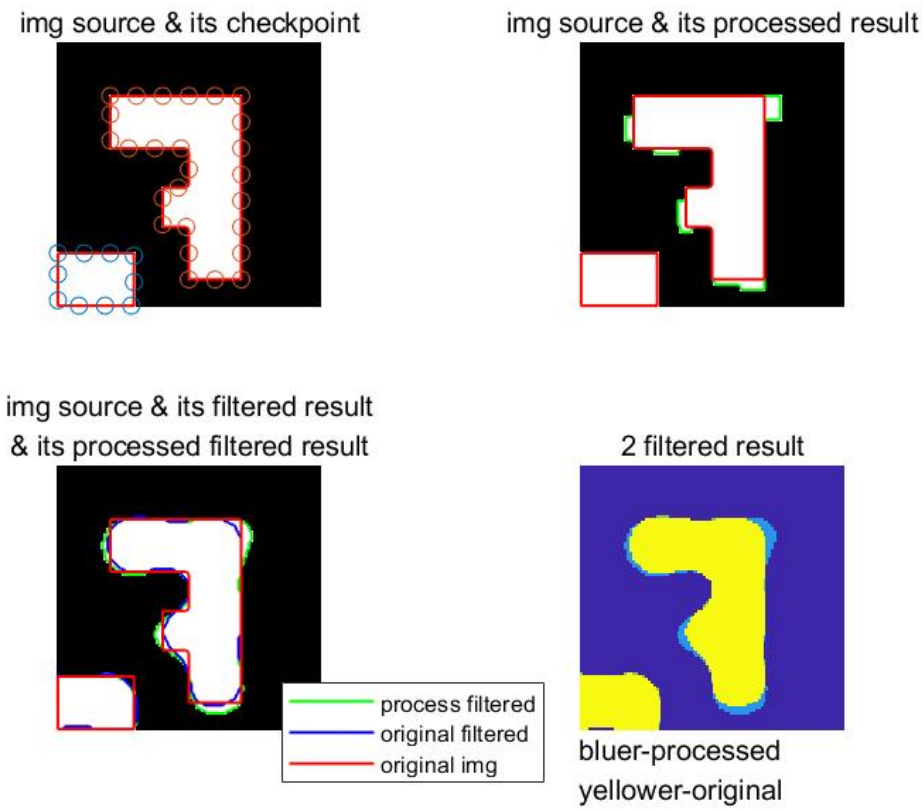
目前结果

原始未切割图形：



处理结果：

（切割图像及分割点）（处理结果）（滤波前后和目标）（目标及处理后的滤波对比）



主要函数

```
function OPC(img_source) % main
function
[img_process,bs,flag,EPE]=opc_process(bs,img_source,img_process_base,EPE_min,knu
m) % start opc, foreach part pof image do opc
function
[img_process,bs,flag,EPE_min]=opc_process_k(bs,img_source,img_process_base,EPE_m
in,k) % for one part opc, do

function [img,flag]=cal_opc_w(img,img_source_i,bs,type,k,w) % for specified
width do opc
function [img,flag]=cal_opc_w_process(img,img_source_i,x1,x2,y1,y2,type,width)
% for specified line & width do opc
function EPE=cal_EPE(img_source,img_process) % calculate EPE

function img_process=filtering(img_target)
function p=show_edge(img,color,is_scatter)
% following 2 func are used to do filtering in full img
function img=cut_center_img(img_source)
function img_source=restore_center_img(img_source,img_filled)
```

OPC

input: img_source % 原始未裁剪过的正方形图

主要做开始的前期准备。

先把原图切割，获取图形轮廓坐标（**？这里有个问题是不是很一致**TODO）并存储在元组bs中，调用opc_process开始实行opc，得到[img_process,bs,flag,EPE]。opc_process里对分开的图形依次调用opc_process_k。

opc_process_k

input: bs,img_source,img_process_base,EPE_min,k

 % 轮廓相关, origin_not_cut_img, best processed img_cut & EPE, k^th part graphic

output: img_process,bs,flag,EPE_min

 % best processed img_cut & EPE，处理后的轮廓相关，是否直接达到预期EPE_min

对于第k个图形。首先看是否idx已到0，即所有的轮廓处理完毕，完毕则直接退出。

对于type=0，1，-1分别计算EPE并比较哪个使得EPE小，就选择哪个。

- 如果是0，直接判断是否小于指定最小值（则记录w, EPE, flag然后退出该循环），或者小于当前最小值，否则继续循环。
- 如果是1/-1，则对于不同的width判断哪个使得EPE最小并记录
    - 以img_process_base为基础，type,k,w,bs等为输入调用cal_opc_w进行处理。处理完还原到原图中再进行滤波后调用cal_EPE进行计算，注意只是计算，不进行记录。

如果此时最小值已经使得超过指定EPE，则记录bs{k,3}中并退出，否则用达到最小值的type和width再进行计算（这里可替换成每次最小值记录，?会不会麻烦了呢TODO）得到图片处理结果并以此作为下一次opc_process_k的base，记录bs，然后进行下一轮廓线段的移动（轮廓线是由最开始的来确定的，而没有每次进行迭代，？后序可考虑TODO）。

cal_opc_w_process

input: img,img_source_i,x1,x2,y1,y2,type,width

 % 当前图片, origin_cut_img, 坐标, type, width

output: img,flag

 % 处理后的图，是否正常操作(是否不是边缘)

只根据current_idx和type, width对现有image进行修改。

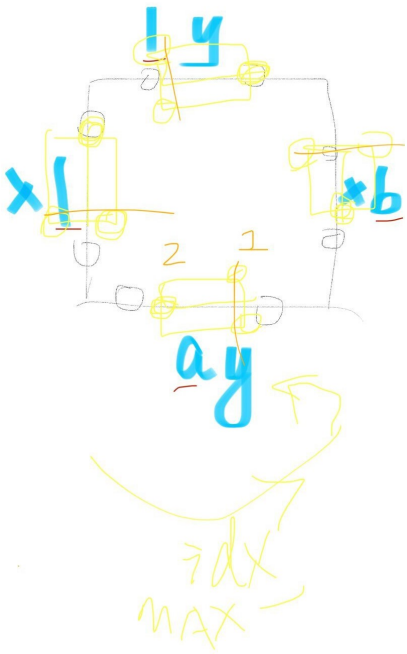
❓事实上current_idx还是基于最初图像的划分，而不是用迭代、逐步修改。

在type!=0下

- 如果是一条水平线，在上/下则不画退出
    - 检查xy是否在图片范围内，不是则变成内。根据中点的上下情况和type共同决定4情况下的draw_rec坐标输入。
- 如果是一条竖直线，在左/右则不画退出
    - 检查xy是否在图片范围内，不是则变成内。根据中点的左右情况和type共同决定4情况下的draw_rec坐标输入。
- 如果是一条对角线（既不水平也不竖直）

- 根据(x1,y2)/(x2,y1)情况和是否是角落确定draw_rec的中心点。
- 无法判断的情况就直接返回退出。

实际画前两种情况的时候为避免重叠的问题，对于一个线段，只包括一边的点，❓不确定现在的做法是否会导致EPE时候算错。



cal_EPE

输入原始目标版图和处理后结果版图，返回EPE值。

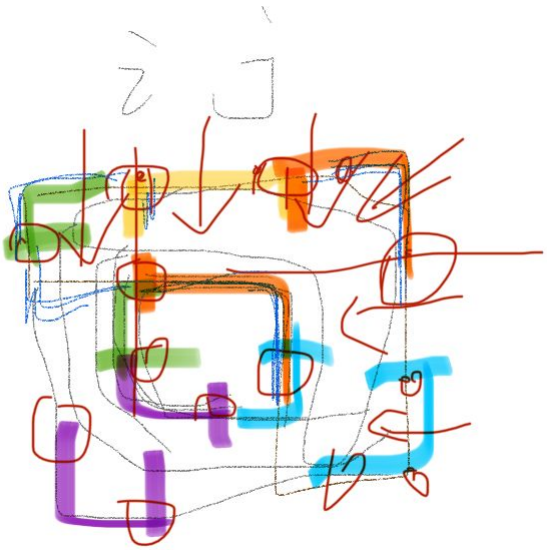
计算方法是对于原始版图每一条线段的后端点（前后由求轮廓函数决定，后=idx小），在处理版图找到对应点，然后计算距离，统计整个版图的总和作为EPE。

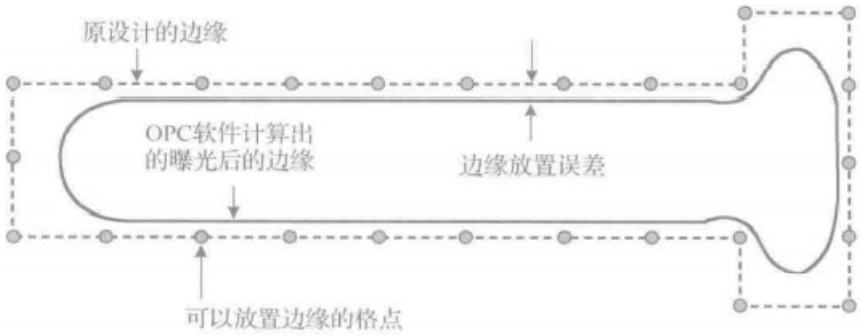
-- 这里的每次计算都是对所有图形算一遍，因为不确定计算轮廓的函数得到的k是否是对应的。

- 对于水平线段，计算后端点的竖直切线下，在处理后版图中离它最近的点的距离。
- 对于竖直线段，计算后端点的水平切线下，在处理后版图中离它最近的点的距离。
- 对于角落，L or -| 情况同水平情况，|- or _| 情况同竖直情况。

如果找不到这样的值（找到的的最近点距离超过最大移动距离则？TODO）

❓感觉有比较大的问题。现在是切割线段的一个端点对应的在滤波后的图形的点到目标的距离。比如我取的总是左端点的到原图形的边缘，这就导致比如总是试图填满上面角落但是忽视下面角落的问题。所以这里仍需改进TODO。考虑取线段中点&角落点？





注意事项

- OPC只对完整图中的一部分做，但每次filter都是把处理好的的填回去然后进行滤波处理。
- bs是一个轮廓相关数据的cell，k表示第k个轮廓、图形，
  - bs(k,1)=xs_list,　　bs(k,2)=ys_list,
  - bs(k,3)=choose_type_list （每条边向内/外移动的量，边缘轮廓不做处理--0），
  - bs(k,4)=current_idx，初始值为length。
- 做cal_opc时候，事实上current_idx还是基于最初图像的划分，而不是根据新图像重新进行线段划分，这点要注意。
- 所有的xy都是第x行，第y列的意思，因为矩阵存储时img(x,y)表示试试这样的。

存在问题

❗明明是角落但是没有检测到是角落的错是因为cal_opc传入进来的图像不是初始图像，是处理后的，所以judge corner使用对象错误。

❗ 错位，差一格没操作？,实际画的时候应该画一个点以此避免重叠的问题，现在的做法可能会导致EPE时候算错吗?

❗ 一直都是最后一个w的值的问题？代码错误，已更正。

⭕仍然有5：-1：2与2：5结果不一样的问题？是因为出现两个EPE相等情况时，是选取先计算得到最小值的，所以两个顺序造成结果不同。

❓左下角没受边缘控制不画图，因为不是直线type，TODO？情况不见了？

❓分割点目前是等距离分割的，TODO