

# Abstract

---

- Using "Poisson Image Editing" technique from `sigGraph 2004` to make two images fuse together.
- 已知target的被贴图区域的边缘一圈的像素值，以及贴图的散度，使用拉普拉斯算子，运用泊松方程求出融合图像被贴图区域的像素值。使用共轭梯度法求解，并使用稀疏矩阵运算库 `scipy` 进行加速。
  - 注意：对于图片融合任务，我们使用三个图片进行一次融合，分别是source（源图像）、mask（即源图像ROI部分的掩膜）、target（待插入贴图的图像），我们将源图像ROI部分称为贴图。

# Keyword

---

- Poisson Equation
- Laplacian Operator
- Sparse Matrix
- Conjugate Gradient Iteration

# Principle

---

## 泊松方程

- 泊松方程是数学中的一个常见的偏微分方程。

- 公式： $\Delta f = \Omega$

- $\nabla$  梯度：一阶微分（离散形式）

$$\frac{df(x)}{dx} = \frac{f(x+h) - f(x-h)}{2h}$$

- $\Delta$  散度：二阶微分（离散形式）

$$\frac{d^2 f(x)}{dx^2} = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

- 随着 $h \rightarrow 0$ ，式子的值也越来越逼近微分结果，而对于离散形式的图片来说， $h$ 最小为1，即相邻像素的距离。因此，二阶微分可以转换成如下卷积的形式：

- 一维

$$[1, -2, 1]$$

- 二维，又称拉普拉斯卷积核

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

# 偏微分方程

- [WIKI](#)
- Partial Differential Equation, 简称PDE。
- 指含有未知函数及其偏导数的方程，符合这个关系的函数是这个方程的解。

## 梯度共轭法

- [WIKI](#)
- 适用于系数矩阵为实正定对称的稀疏矩阵的线性方程组的求解。

# Mathematics

1. 设原图为 $4 * 4$ 矩阵，其ROI为中间 $2 * 2$ 的小矩形区域，构建泊松方程。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1	4*4 source	a	b	c	d													2*2 mask	f	g							
2		a	b	c	d														j	k							
3		e	f	g	h																						
4		i	j	k	l																						
5		m	n	o	p																						
6																											
7																											
8	A	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	*	X	R	G	B	=	b	R	G	B
9	a	1																	a	Ra	Ga	Ba		a	Ra	Ga	Ba
10	b		1																b	Rb	Gb	Bb		b	Rb	Gb	Bb
11	c			1															c	Rc	Gc	Bc		c	Rc	Gc	Bc
12	d				1														d	Rd	Gd	Bd		d	Rd	Gd	Bd
13	e					1													e	Re	Ge	Be		e	Re	Ge	Be
14	f						1	-4	1										f	Rf	Gf	Bf		f	div(Rf)	div(Gf)	div(Bf)
15	g							1	-4	1									g	Rg	Gg	Bg		g	div(Rg)	div(Gg)	div(Bg)
16	h								1										h	Rh	Gh	Bh		h	Rh	Gh	Bh
17	i								1										i	Ri	Gi	Bi		i	Ri	Gi	Bi
18	j									1	-4	1							j	Rj	Gj	Bj		j	div(Rj)	div(Gj)	div(Bj)
19	k										1	-4	1						k	Rk	Gk	Bk		k	div(Rk)	div(Gk)	div(Bk)
20	l											1							l	Rl	Gl	Bl		l	Rl	Gl	Bl
21	m												1						m	Rm	Gm	Bm		m	Rm	Gm	Bm
22	n													1					n	Rn	Gn	Bn		n	Rn	Gn	Bn
23	o														1				o	Ro	Go	Bo		o	Ro	Go	Bo
24	p															1			p	Rp	Gp	Bp		p	Rp	Gp	Bp

2. 因为需要使用梯度共轭法求解方程，这要求系数矩阵是实正定对称的稀疏矩阵，故对方程进行一定的等价变换。

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1	4*4 source	a	b	c	d													2*2 mask	f	g							
2		a	b	c	d														j	k							
3		e	f	g	h																						
4		i	j	k	l																						
5		m	n	o	p																						
6																											
7																											
8	A	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	*	X	R	G	B	=	b	R	G	B
9	a																		a	Ra	Ga	Ba		a	Ra	Ga	Ba
10	b																		b	Rb	Gb	Bb		b	Rb	Gb	Bb
11	c																		c	Rc	Gc	Bc		c	Rc	Gc	Bc
12	d																		d	Rd	Gd	Bd		d	Rd	Gd	Bd
13	e																		e	Re	Ge	Be		e	Re	Ge	Be
14	f																		f	Rf	Gf	Bf		f	div(Rf)-Rb-Re	div(Gf)-Gb-Ge	div(Bf)-Bb-Be
15	g																		g	Rg	Gg	Bg		g	div(Rg)-Rc-Rh	div(Gg)-Gc-Gh	div(Bg)-Bc-Bh
16	h																		h	Rh	Gh	Bh		h	Rh	Gh	Bh
17	i																		i	Ri	Gi	Bi		i	Ri	Gi	Bi
18	j																		j	Rj	Gj	Bj		j	div(Rj)-Ri-Rn	div(Gj)-Gi-Gn	div(Bj)-Bi-Bn
19	k																		k	Rk	Gk	Bk		k	div(Rk)-Rl-Ro	div(Gk)-Gl-Go	div(Bk)-Bl-Bo
20	l																		l	Rl	Gl	Bl		l	Rl	Gl	Bl
21	m																		m	Rm	Gm	Bm		m	Rm	Gm	Bm
22	n																		n	Rn	Gn	Bn		n	Rn	Gn	Bn
23	o																		o	Ro	Go	Bo		o	Ro	Go	Bo
24	p																		p	Rp	Gp	Bp		p	Rp	Gp	Bp

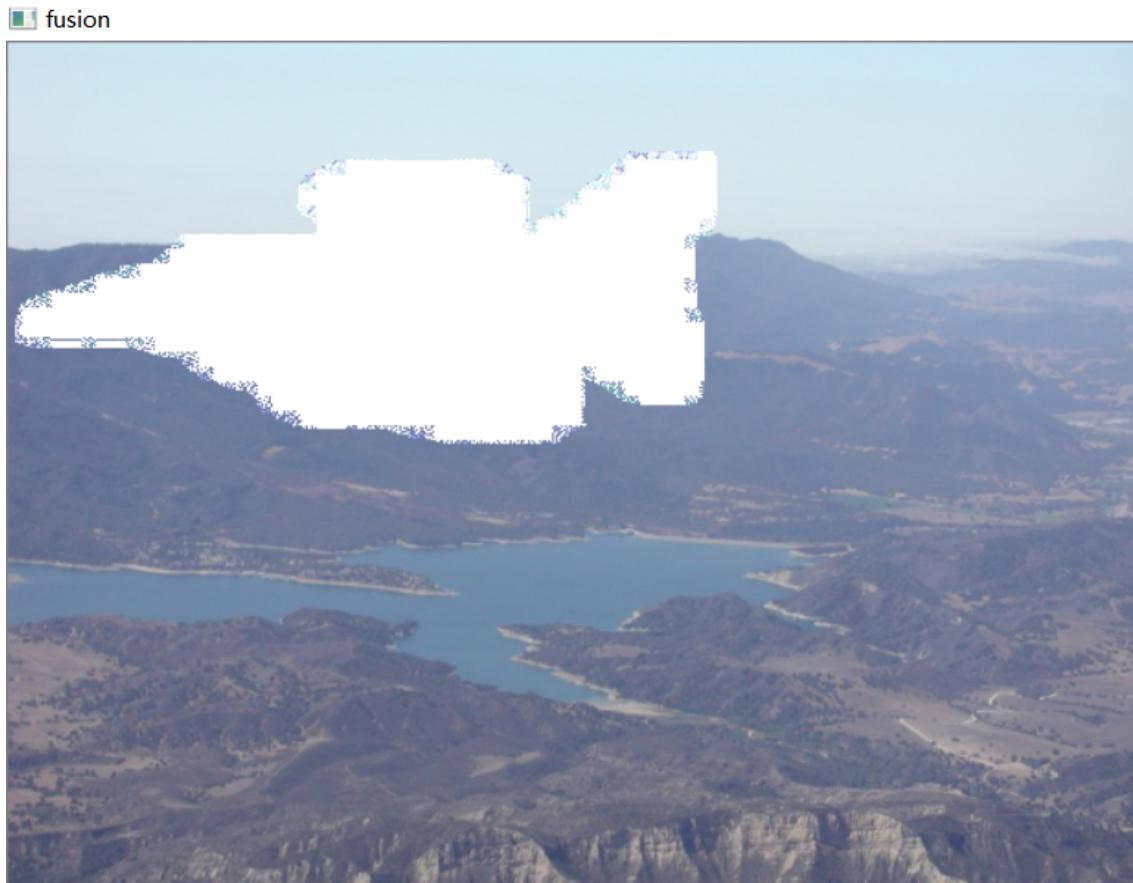
3. 注意，如果令内部区域为不规则图形，如将f-g-i作为内部区域，则更有助于理解方程的等价变换之后代码应该怎么写。这里留给读者自行验证。

## Coding

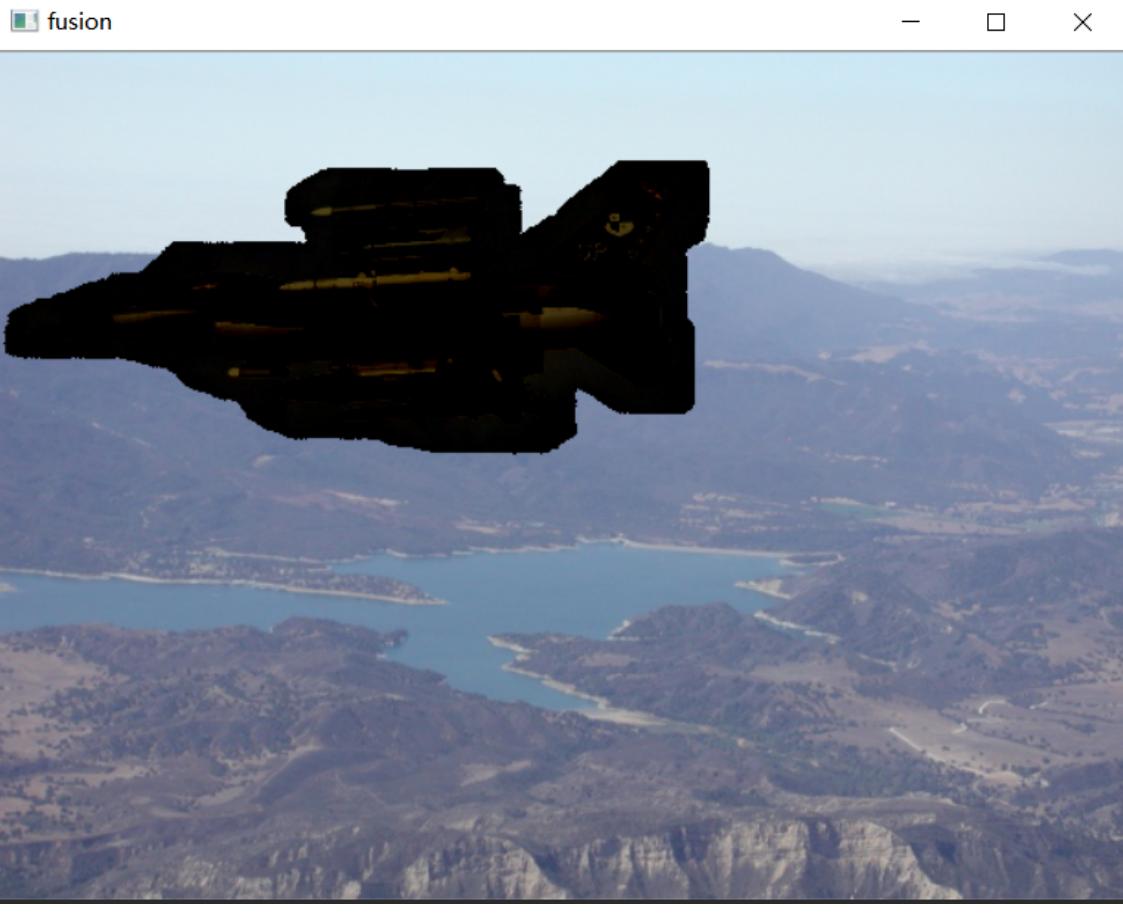
- 文件目录结构

- docs: 存放报告文档
- imgs: 存放使用的图像
- source: 存放工程代码
  - utils: 存放实验参数以及各种脚本。
    - `args.py`: 实验参数, 供main.py使用
    - `createMask.py`: 生成mask的脚本。
  - `main.py`: 主函数, 进行泊松编辑
  - `PIE.py`: 泊松编辑的实现类

1. 第一个问题, 贴图内部像素值太大, 且边缘有毛刺。



2. 毛刺显然可以通过改变mask的判断方式解决, 即将mask图中偏向于黑色的像素全部算作mask, 属于数据层面的不严谨; 而颜色全都过大则是由于读入的图片在进行计算时会出现溢出, 导致结果不准确, 可以通过强制类型转换破除计算过程中像素值为0-255的限制。



3. 最后这个bug比较难找，经观察，泊松方程解出的解非常小。再观察系数矩阵A和值矩阵b，发现b的值非常小，应该是边缘部分的真实值没有成功过渡给b。最终成功找出代码错误，完成复现。

```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Poisson Image Editing [E:\common-folders\research\Computer Vision\intern\Poisson Image Editing] - PIE.py
Project Poisson Image Editing E:\common-folders\research\Computer Vision\intern\Poisson Image Editing source PIE.py
  main.py args.py
  docs
  imgs
  outputs
  source
    utils
      __init__.py
      main.py
      PIE.py
External Libraries
Scratches and Consoles
Run: main
Run: main x
E:\Anaconda\Anaconda\envs\Graphics\python.exe "E:/common-folders/research/Computer Vision/intern/Poisson Image Editing/PIE.py"
--- build A
--- 转换为稀疏矩阵
spending time=50s
100% [██████████] 40800/40800 [00:01<00:00, 21000.64it/s]
--- build b
--- forward
100% [██████████] 40800/40800 [00:02<00:00, 19564.05it/s]

  print('--- 转换为稀疏矩阵')
  pre = time()
  self.A = sparse.lil_matrix(self.b)
  print(f'spending time={int(time()) - pre}s')

  print('--- build b')
  for i in tqdm(range(self.b.shape[0])):
    for j in range(self.b.shape[1]):
      x, y = self.jdOrd[i]
      self.A[i, self.jdOrd2id[(x1, y1)]] = 1
      if self.neighbor_is_mask[i][1]:
        self.A[i, self.jdOrd2id[(x1, y1)]] = 1
      if self.neighbor_is_mask[i][2]:
        self.A[i, self.jdOrd2id[(x, y-1)]] = 1
      if self.neighbor_is_mask[i][3]:
        self.A[i, self.jdOrd2id[(x, y+1)]] = 1

```

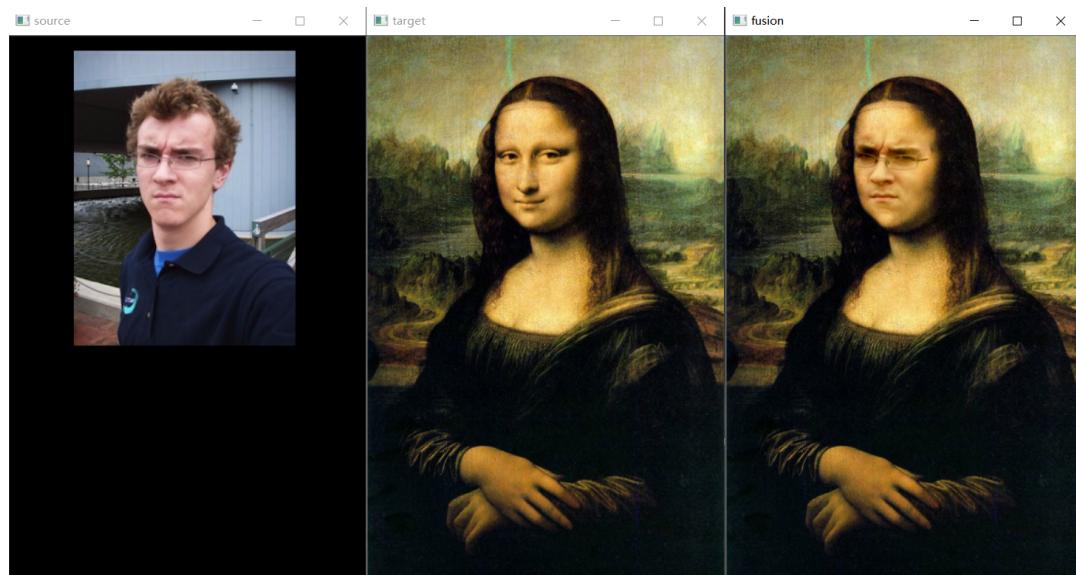
4. 检查调整target图片ROI区域的代码正确性。

 fusion

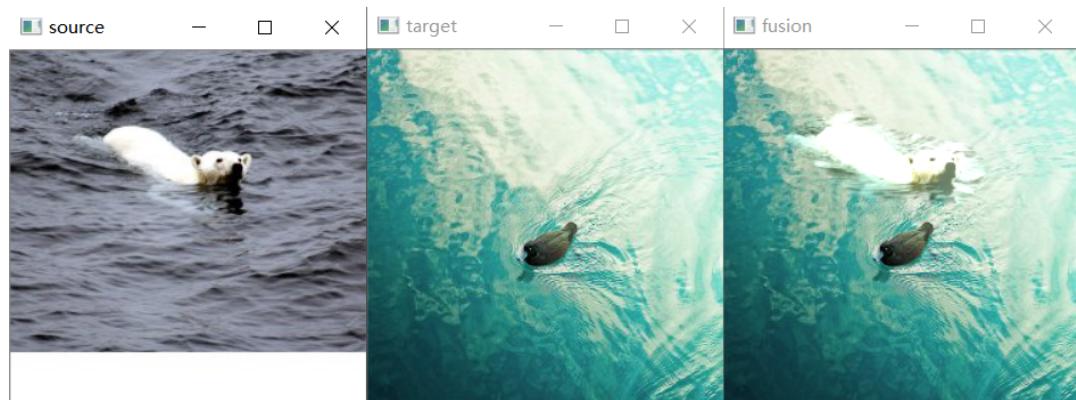


5. 检查其它图片示例的代码正确性。

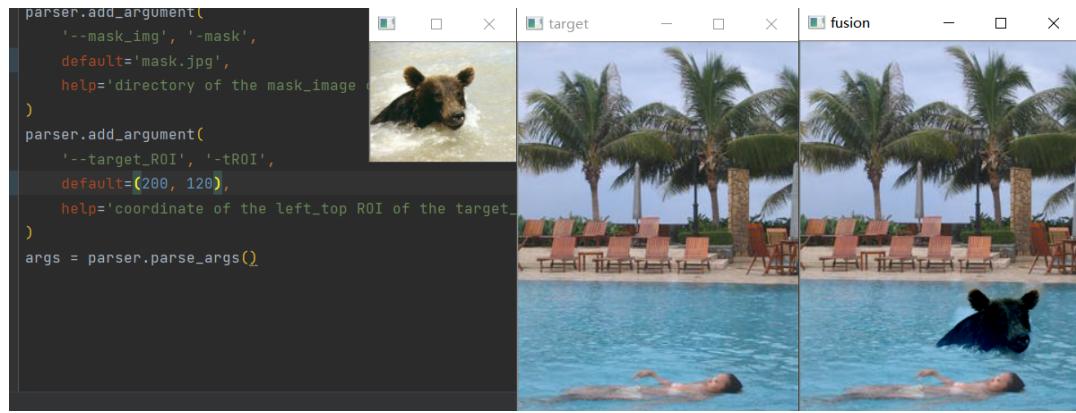
1. mona



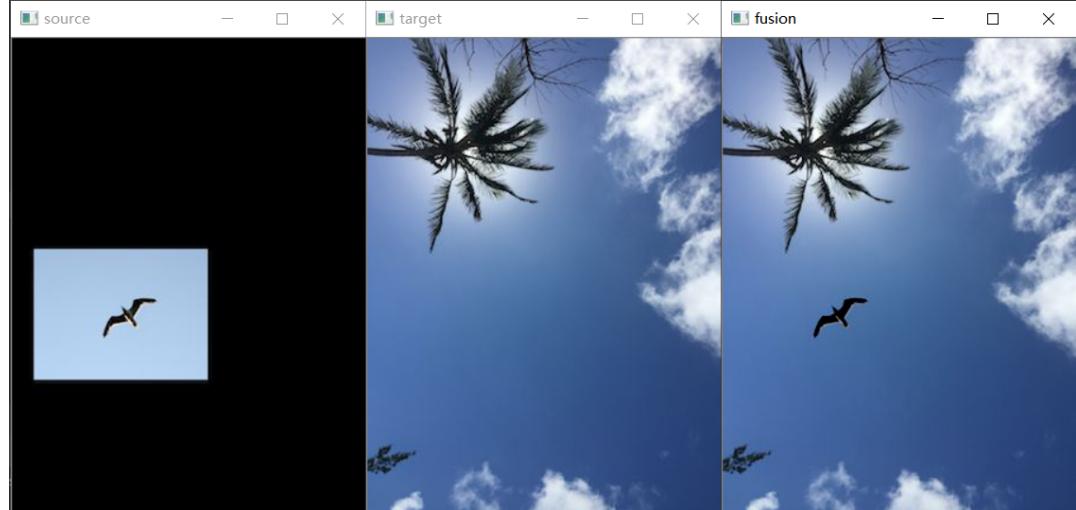
2. sealion



### 3. bear

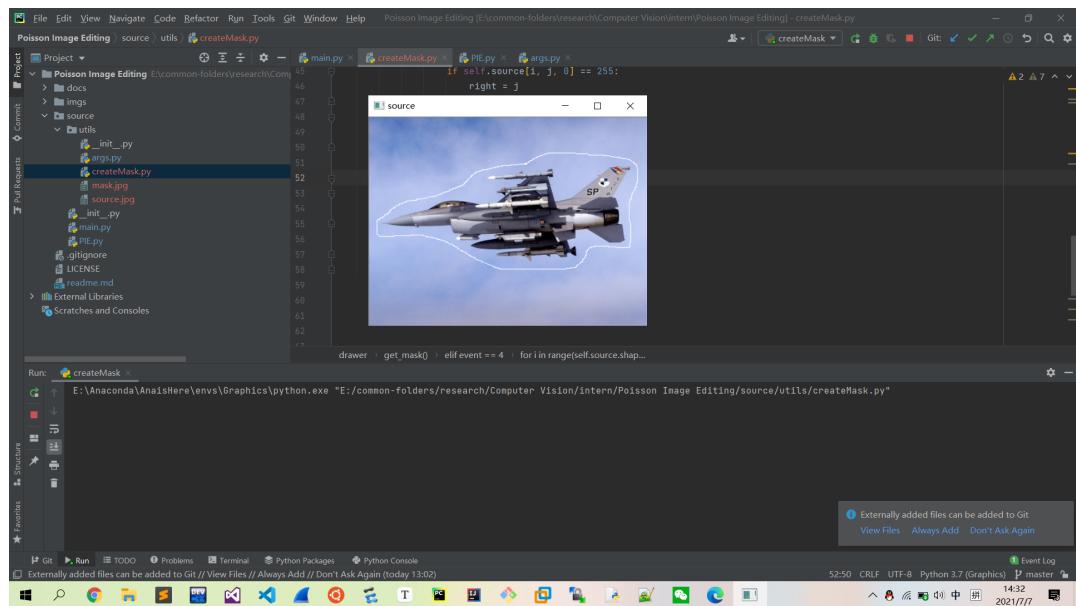


### 4. eagle

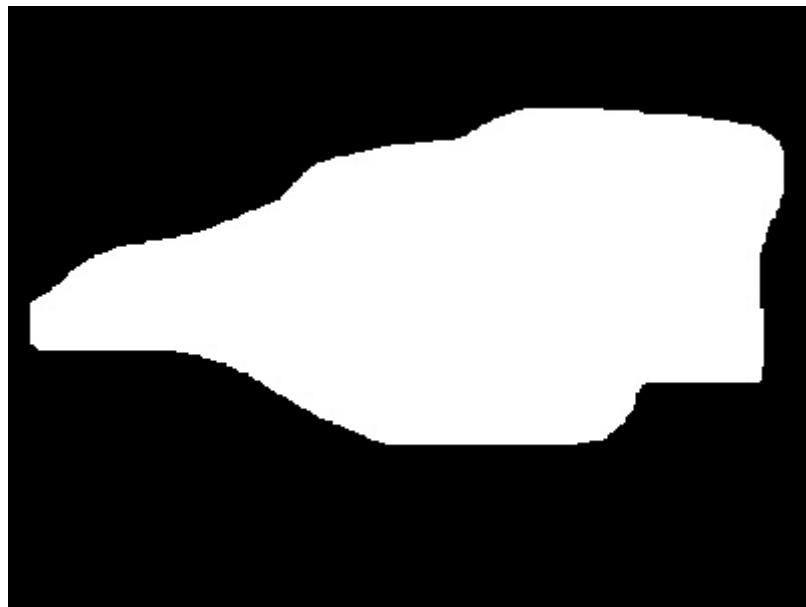


## 6. 制作mask绘制工具

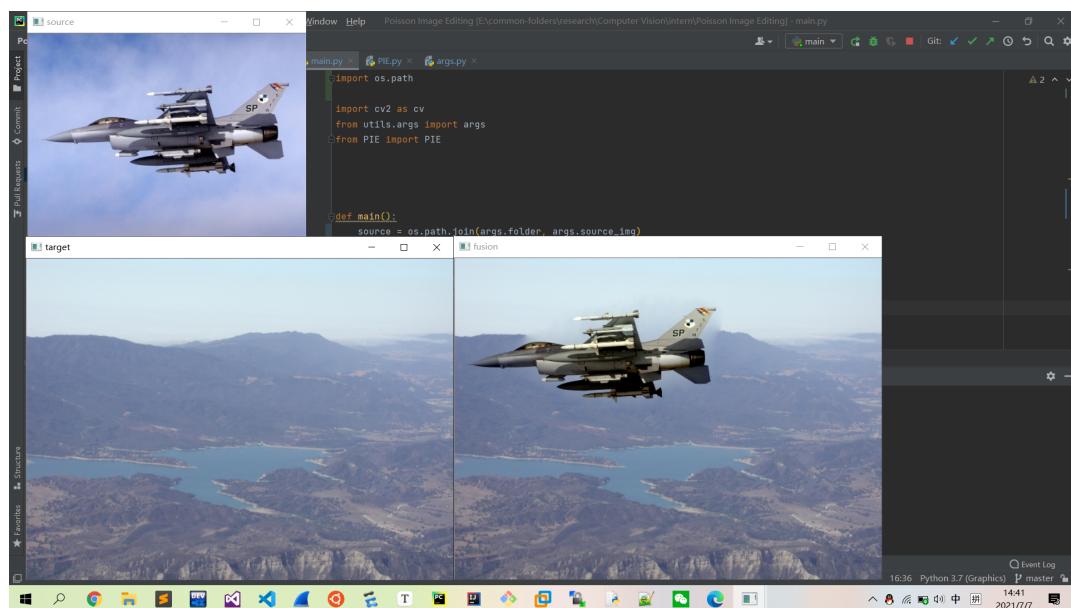
- 首先读入一张source图片并显示。
- 捕获用户鼠标动作，用户长按鼠标左键进行拖拽，绘制出轮廓线。



- 用户释放鼠标左键后，生成mask图片，并存储。



4. 验证生成mask的正确性。



## Conclusion

- 从飞机的融合图片可以看出来贴图周围有明显的泛白，但是鹰的融合图片又没有这个问题，应继续对数值运算部分进行debug。通过使用mask生成工具进行验证，发现飞机的周围的泛白是原图飞机周围的云雾，由于原图的mask不够精细，贴图时引入了目标图中本不存在的云雾。

## References

- [Poisson Image Editing](#)
- [从泊松方程的解法，聊到泊松图像融合](#)
- [Seamless cloning](#)
- [Code of huajh](#)
- [opencv鼠标绘图](#)