```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                       Radarsat_1 光盘中数据
%                             CSA 成像
%
%
%                              WD
%                      2014.10.19. 13:53 p.m.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 程序说明：
% 主程序是：    Radarsat_1_CSA.m
%
% （1）原始数据说明：
% 文件夹中的 data_1 和 data_2 是已经经过下列方法得到的原始数据，
% 可以直接进行后续成像
% ----------------------------------------------------------
% 使用现成的程序‘compute.azim.spectra.m’中读出数据的方法；
% 利用函数 'laod_DATA_block.m'，实现
%                     - reads /loads data for a block
%                     - converts to floating point
%                     - compansates for the receiver attenuation
% 变量 b -- 需要设置数据取自哪个分区
%                     - b = 1 , from CDdata1
%                     - b = 2 , from CDdata2
% 得到所需要的数据，也即可以直接进行后续 processing 的数据 data。
% ----------------------------------------------------------
% 因此，文件夹中的 data_1 和 data_2 分别是分区 1 和分区 2 的数据，经过了下变频，
%        转换为了浮点数，进行了 AGC 增益补偿，最后转换为了 double 双精度浮点数。
%        因此，直接载入这两个数据就可以进行后续成像。
```

%

% （2） 本文件夹中还有一个文件：CD_run_params

% ——这里面是仿真中需要用的许多参数，直接载入即可。

%

% （3）成像程序说明：

% 由 CSA 的点目标程序修改而来；

% （4）成像流程：

% ——原始数据

% ——经过方位向 FFT，变换到距离多普勒域，进行“补余 RCMC”

% ——经过距离向 FFT，变换到二维频域，进行“距离压缩”、“SRC”、“一致 RCMC”

% ——经过距离向 IFFT，变换到距离多普勒域，进行“方位压缩”和“附加相位校正”

% ——经过方位向 IFFT，回到图像域，成像结束。

%

% 本程序修改截止到： 2014.10.19. 13:53 p.m.

%

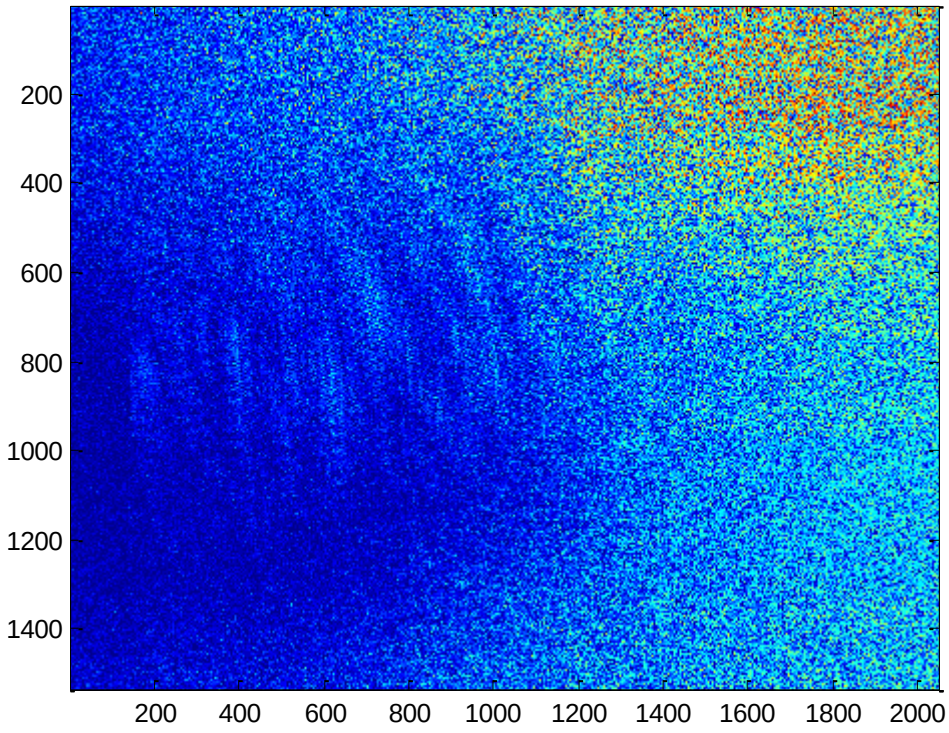
% 注：修改后的程序中，主要是附加了一步：**对原始数据进行补零，再进行后续处理。**

参考目标，参考距离，参考频率的选择：

```
R_ref = R0;                % 参考目标选在场景中心，其最近斜距为 R_ref
fn_ref = fnc;              % 参考目标的多普勒中心频率
```

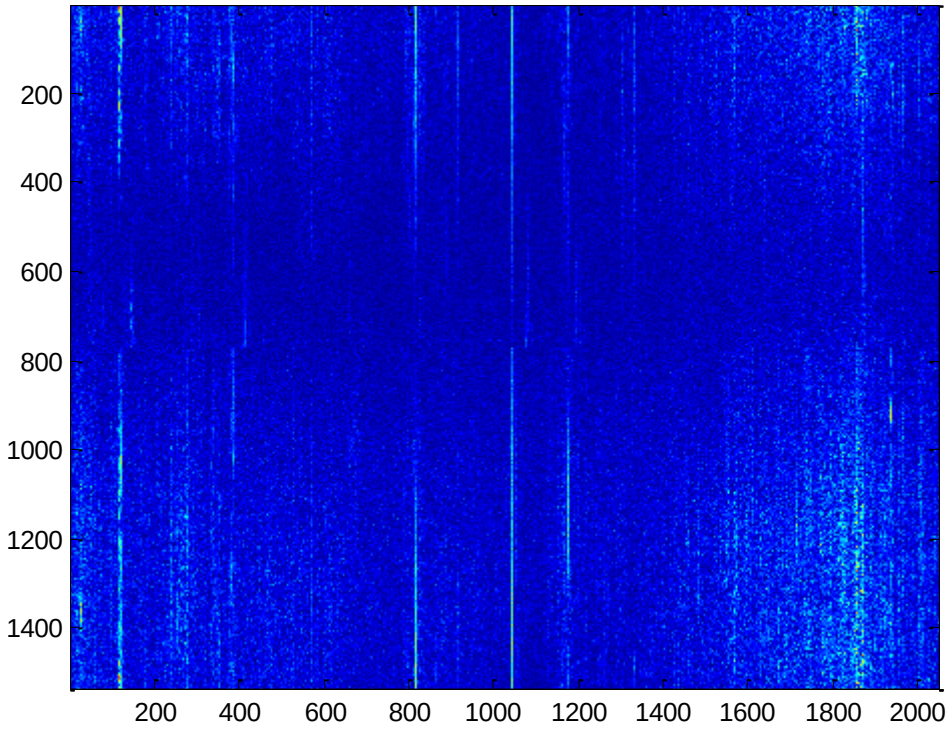
1. 分区 1 成像

原 始 数 据



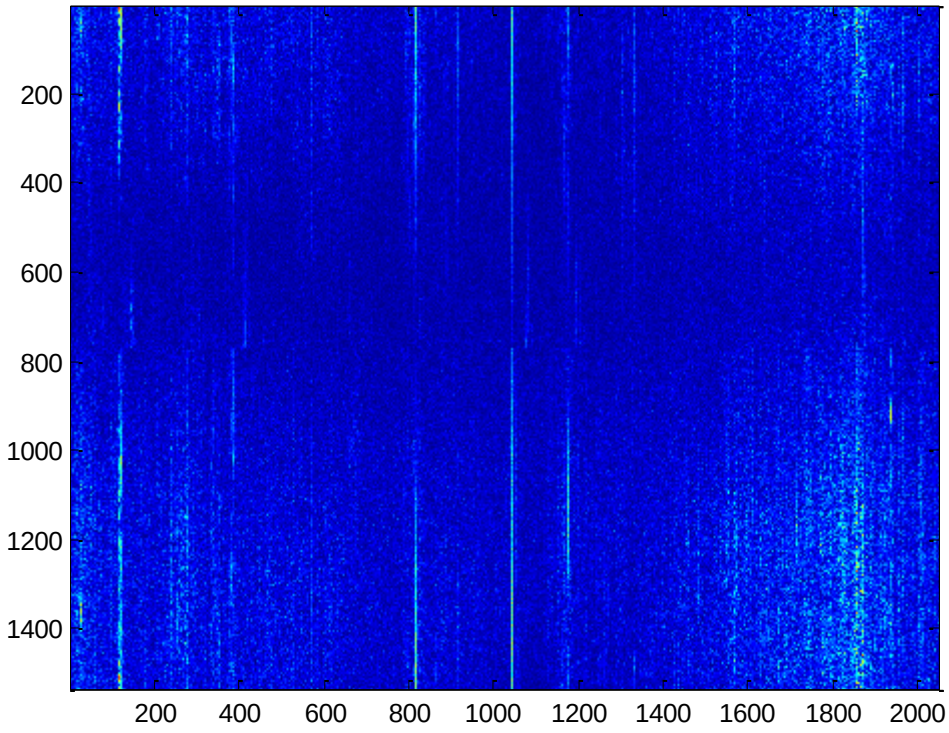
原始数据

完 成 距 离 压 缩 ， SRC， 一 致 RCMC后 ， 距 离 多 普 勒 域

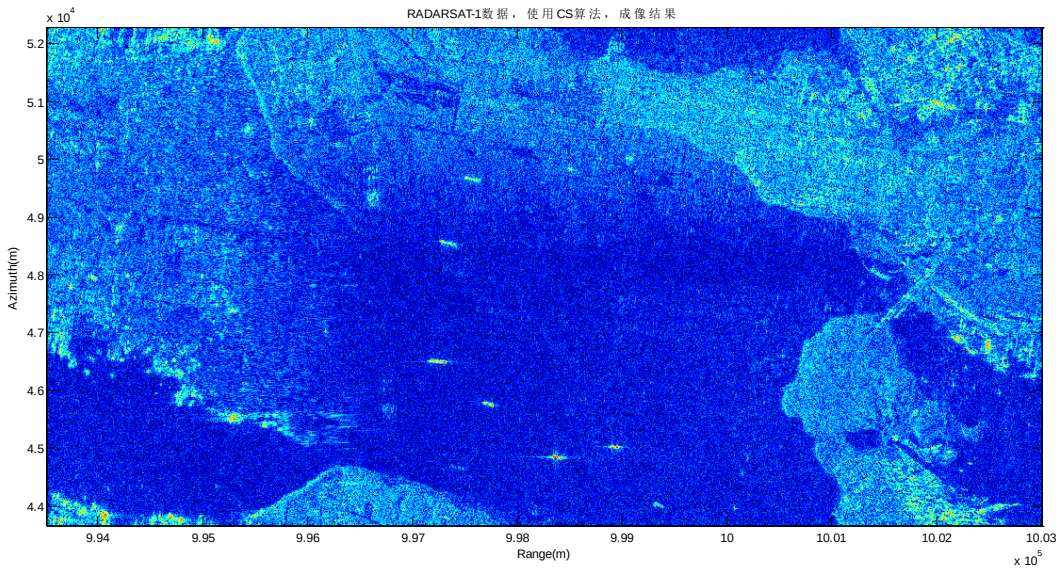


完成所有距离处理（补余 RCMC，距离 MF，SRC，一致 RCMC 后），距离多普勒域

距离多普勒域，进行了相位相乘后（方位MF和附加相位校正）



距离多普勒域，进行了相位相乘后（方位 MF 和附加相位校正）



成像结果（这是未对原始数据补零的成像结果）

我们观察成像结果，发现明显有个问题：就是左右关系不太正确：图像的聚焦情况比较好，说明成像过程基本是没有问题的。但是成像结果的左右关系不太正确，比如左半边的一部分图像应该是在最右边的。

所以，我立马想到了之前点目标仿真时分析过的 CSA 算法是将目标压至参考频率对应的距离单元处，而不是零多普勒处。会不会就是因为这个？我立马进行了以下分析

计算以下计算：

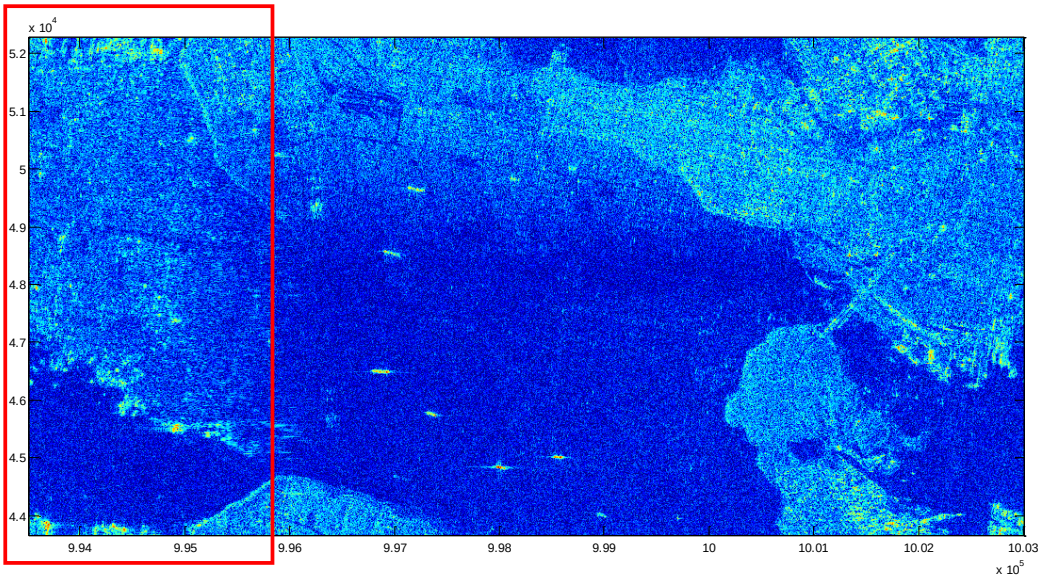
$$\frac{2\left(\frac{R_0}{D\left(f_{\eta_{ref}},V_r\right)}-R_0\right)}{c}\times F_r = 81.4272$$

也就是说，参考目标（最近斜距为R_0）应该被向右偏移了 81.4272 个距离采样单元。

基于此，我人为将整个图像向左搬移 round（81.4272）=81 个采样单元：即将原图像距离向第 81 个采样单元到最后一个采样单元放在新图像的最左边；将原图像第 1 个采样单元到底 80 个采样单元放在新图像的最右边。

```
tmp = round(2*(R0/D_fn_ref_Vr-R0)/c*Fr);
s_tmp(:,1:Nrg-tmp+1) = G(:,tmp:end);
s_tmp(:,Nrg-tmp+1+1:Nrg) = G(:,1:tmp-1);
figure;
imagesc(((0:Nrg-1)+first_rg_cell)/Fr*c/2+R0,((0:Naz-1)+first_rg_line)/Fa*Vr,s_tmp,clim);
axis xy;
```

得到的结果如下：



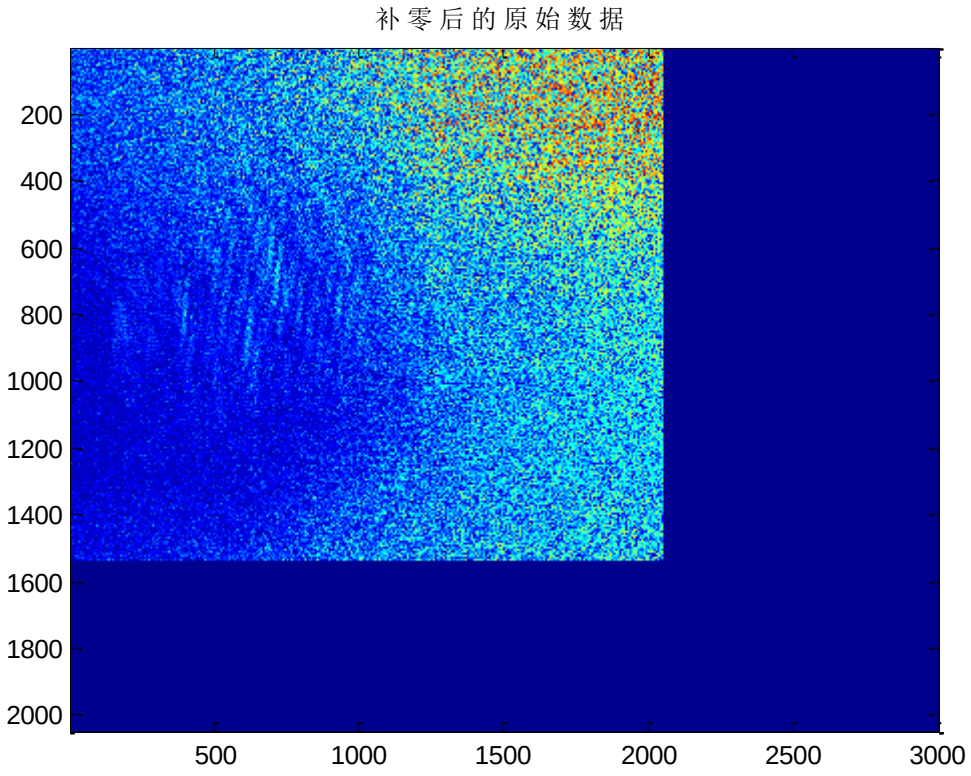
如上所述，移动后的结果

我们看到：因为整个距离向一共有 $N_{rg}=2048$ 个采样单元，因此我们所移动的 81 个采样单元相比于整个距离轴是很小的。移动后的图像相比于原来的变化不大，没有达到我们预期的结果：**红色方框中的结果应该在图像最右侧，而这里的移动没有达到这样的目标。**

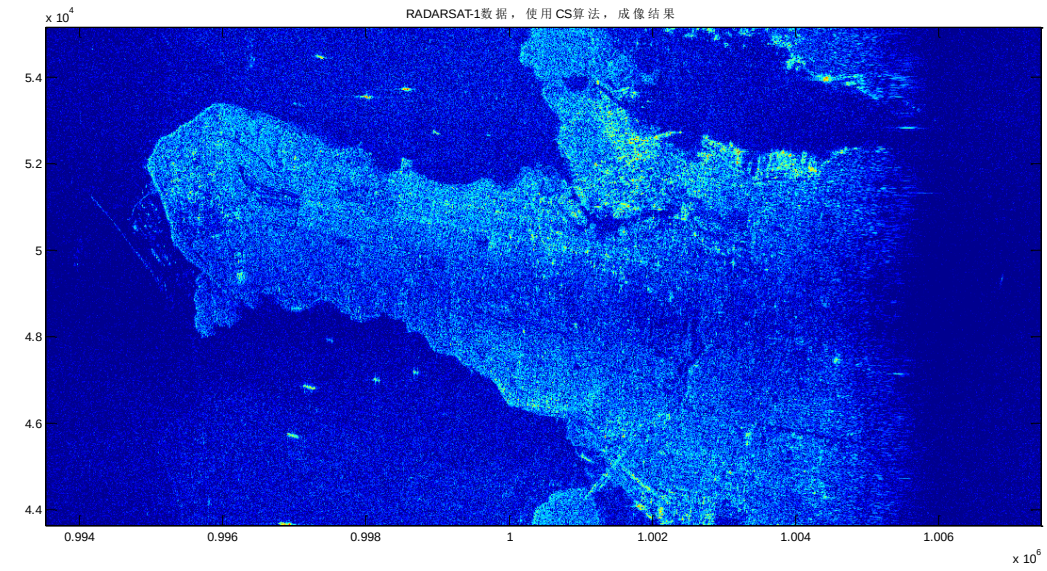
怎么办？

原始数据是 $N_{az}\times N_{rg}$ 的矩阵，我们**对原始数据进行补零**，补零到： 2048×3000

然后再进行后续处理，有：



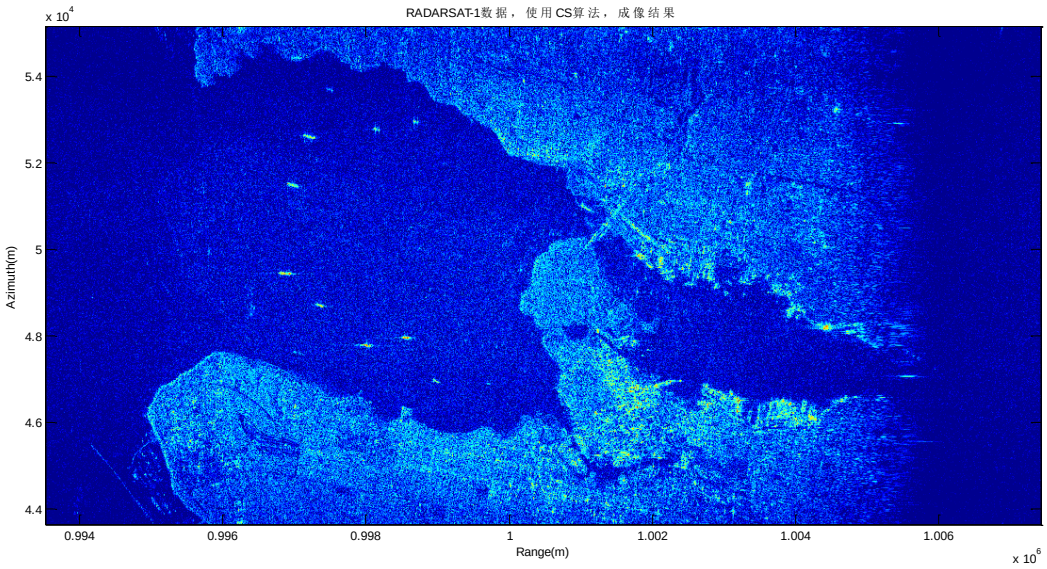
补零后的原始数据



成像结果（经过了 81 个采样单元的移位）

这时候的上下关系反了，但左右是我们想要的结果。

进行 fftshift，将图像的上下半边进行互换，得到以下结果：



最终结果

这个“最终结果”的得来，在原来的基础上，我一共经过了以下几步的处理：

1) 对原始数据补零；

2) 对成像结果进行向左移位：移动的点数和方法如前所述。理论依据是：

$$\frac{2\left(\frac{R_0}{D\left(f_{\eta_{ref}},V_r\right)}-R_0\right)}{c}\times F_r = 81.4272$$

3) 对移位后的图像，再通过 fftshift 进行上下半边的互换，得到了“最终结果”

我认为这个最终结果是很理想的。对比书上的结果，我觉得这块数据的成像结果基本是没问题的。

但是，为何要补零，以及补零后为什么就达到了这样的效果？

——待分析

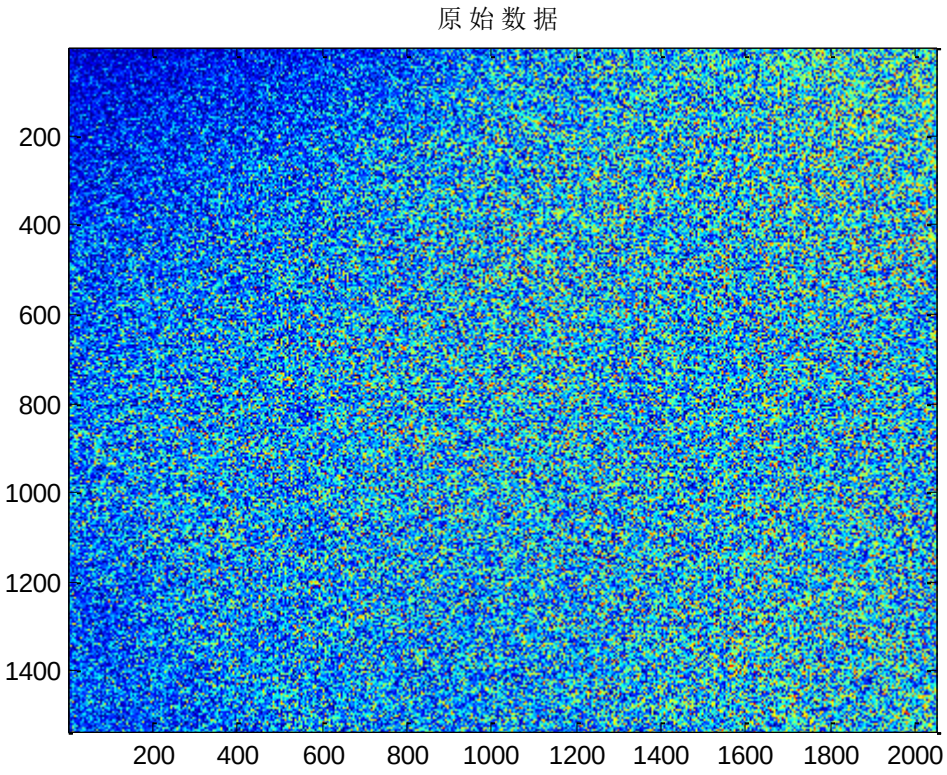
——分析角度：类似于距离向的弃置区问题，以相同的方法来分析方位向的情况（这是师哥给我提供的思路，我还没有进行仔细分析，这是接下去马上就要做的任务。2014.10.20. 晚）

说明：

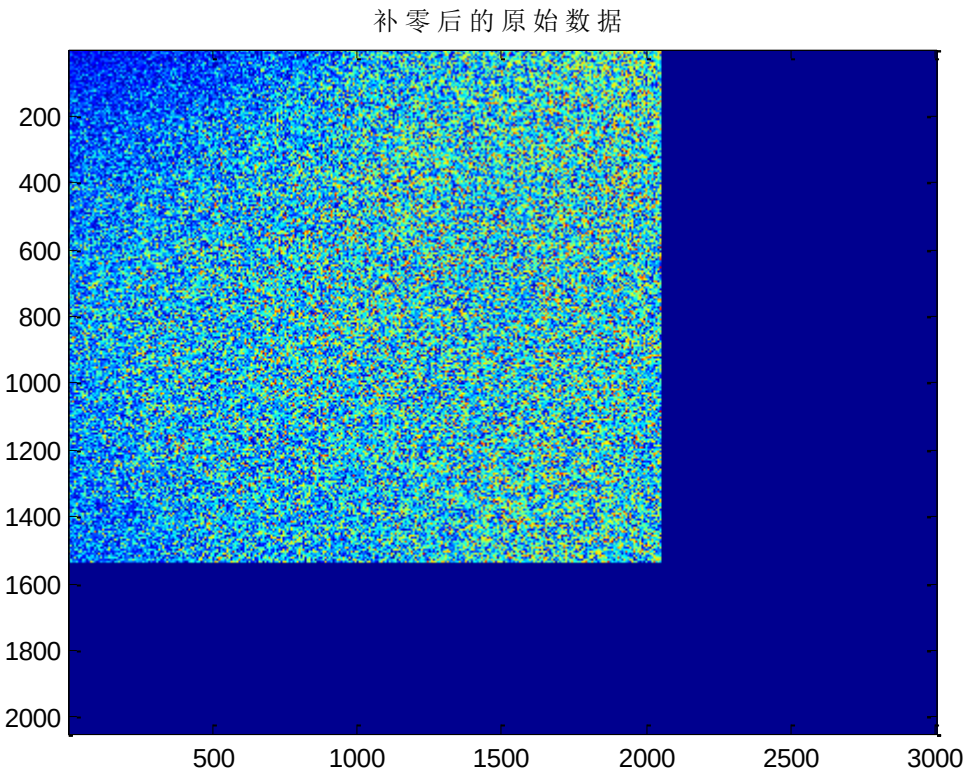
后面再对分区 2，以及对分区 1 和分区 2 的整块数据进行成像时，都进行以下的操作：

1) 对原始数据补零；
2) 对成像结果进行向左移位：移动的点数和方法如前所述；
3) 对移位后的图像，再通过 fftshift 进行上下半边的互换，得到“最终”的成像结果。

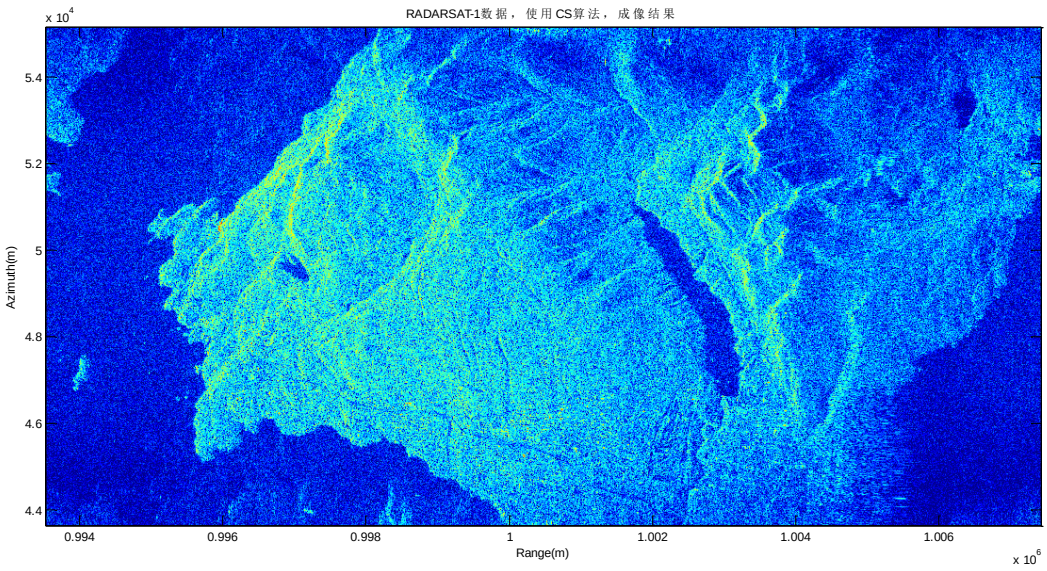
2. 分区 2 成像



分区 2 的原始数据



补零后的原始数据

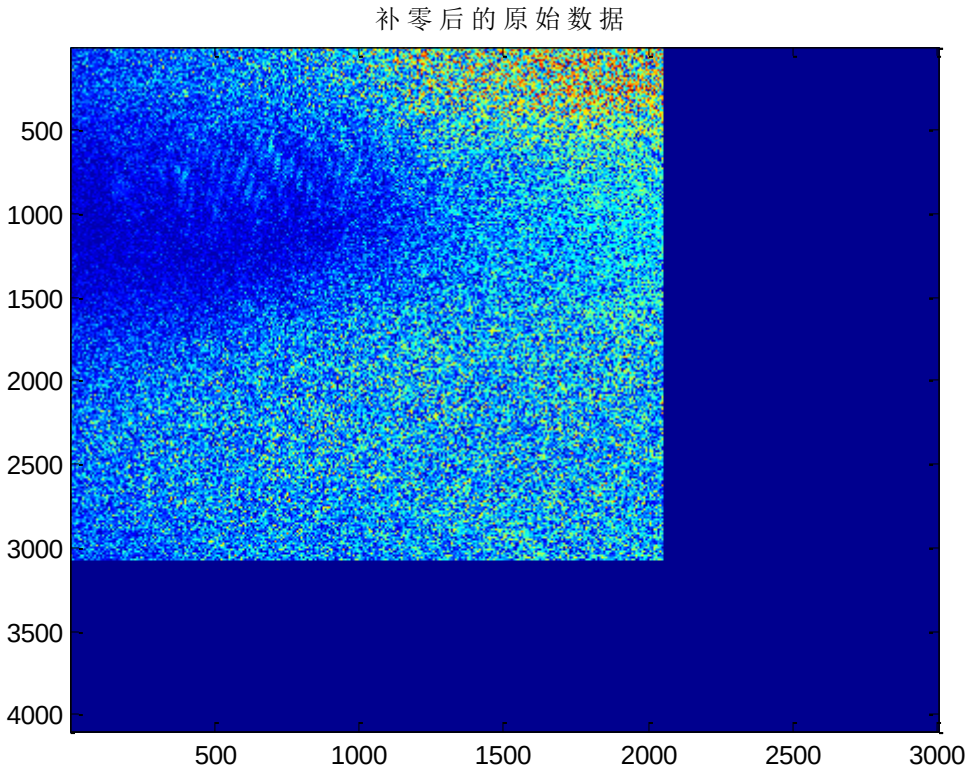

RADARSAT-1数据，使用CS算法，成像结果
x 10^4
5.4
5.2
5
4.8
4.6
4.4
Azimuth(m)
0.994
0.996
0.998
1
1.002
1.004
1.006
Range(m)
x 10^6

成像结果

3. 将分区 1 和分区 2 合并成一个数据块，进行成像

注意此时，由于是将两个分区合并成一个更大的数据块，这时候的原始数据大小 $N_{az}\times N_{rg}=3072\times 204$，因此补零时要将之前分区 1（或分区 2）补零的矩阵再扩大，即补零到：4096×3000。

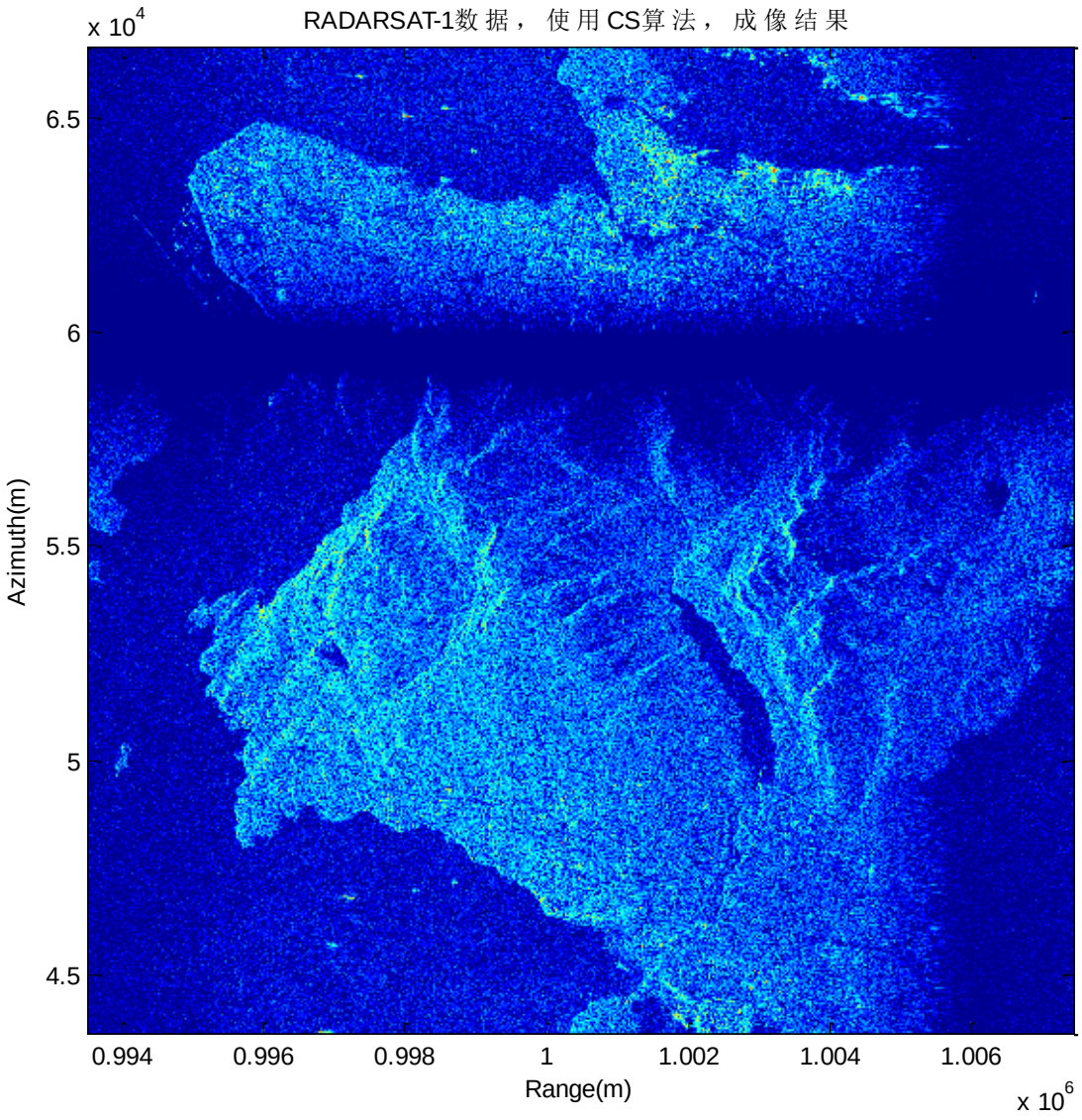
因此，在此种情况下，将原始数据补零至 4096×3000，再进行后续成像。



原始数据（补零后）

以下的成像结果有：

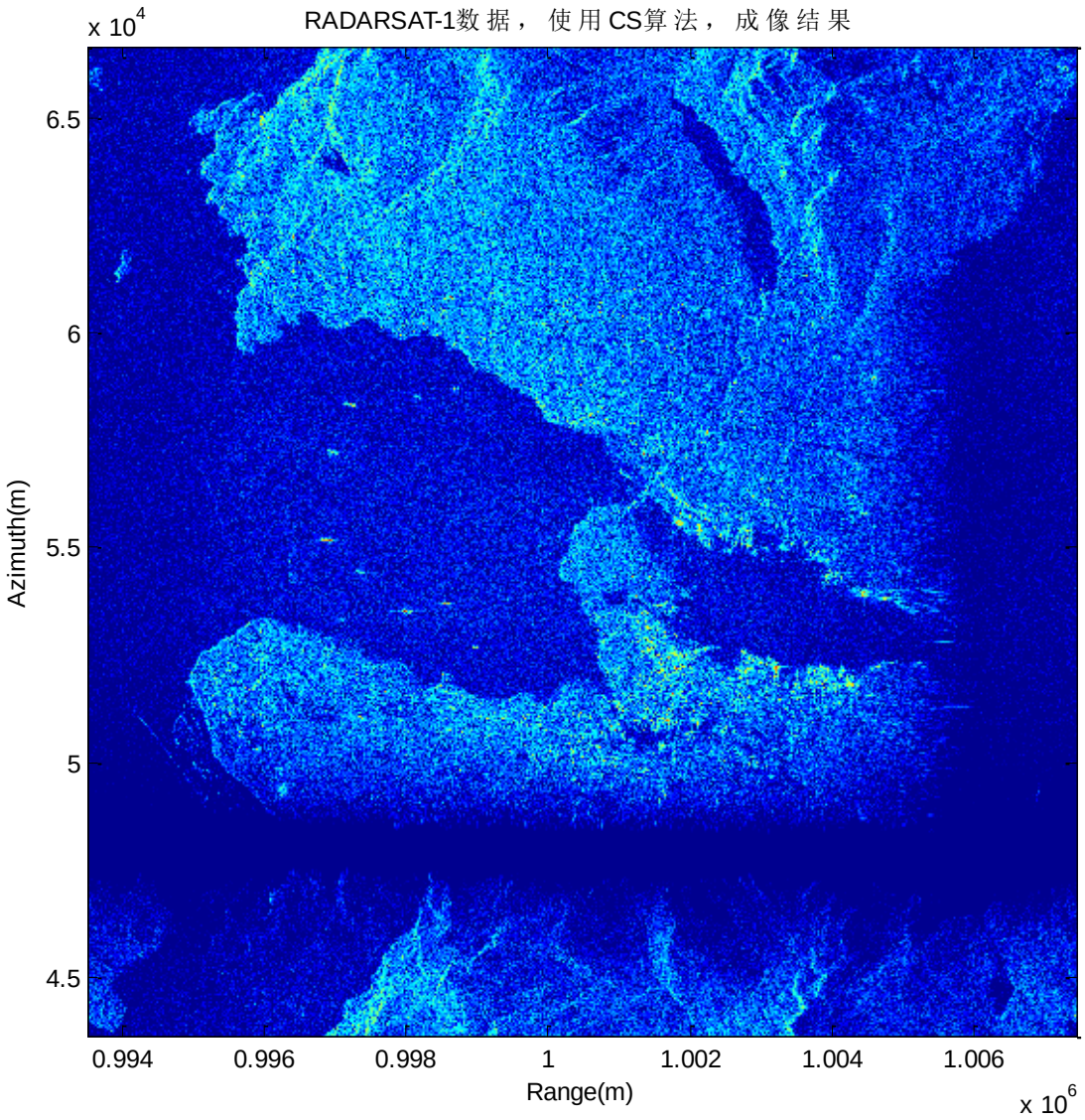
1) 和分区 1 和分区 2 一样，对成像结果进行向左移位：移动的点数和方法如前所述；但是不对结果进行 fftshift（不进行上下半边互换）



对成像结果进行了向左移位

理论依据：CSA 将目标压至参考频率对应的距离单元处，而不是压至零多普勒

2) 在 1）的基础上，再对成像结果进行 fftshift，实现上下半边的互换（分区 1 和分区 2 都进行了这样的操作。我们想看看进行了这样操作后的结果如何）



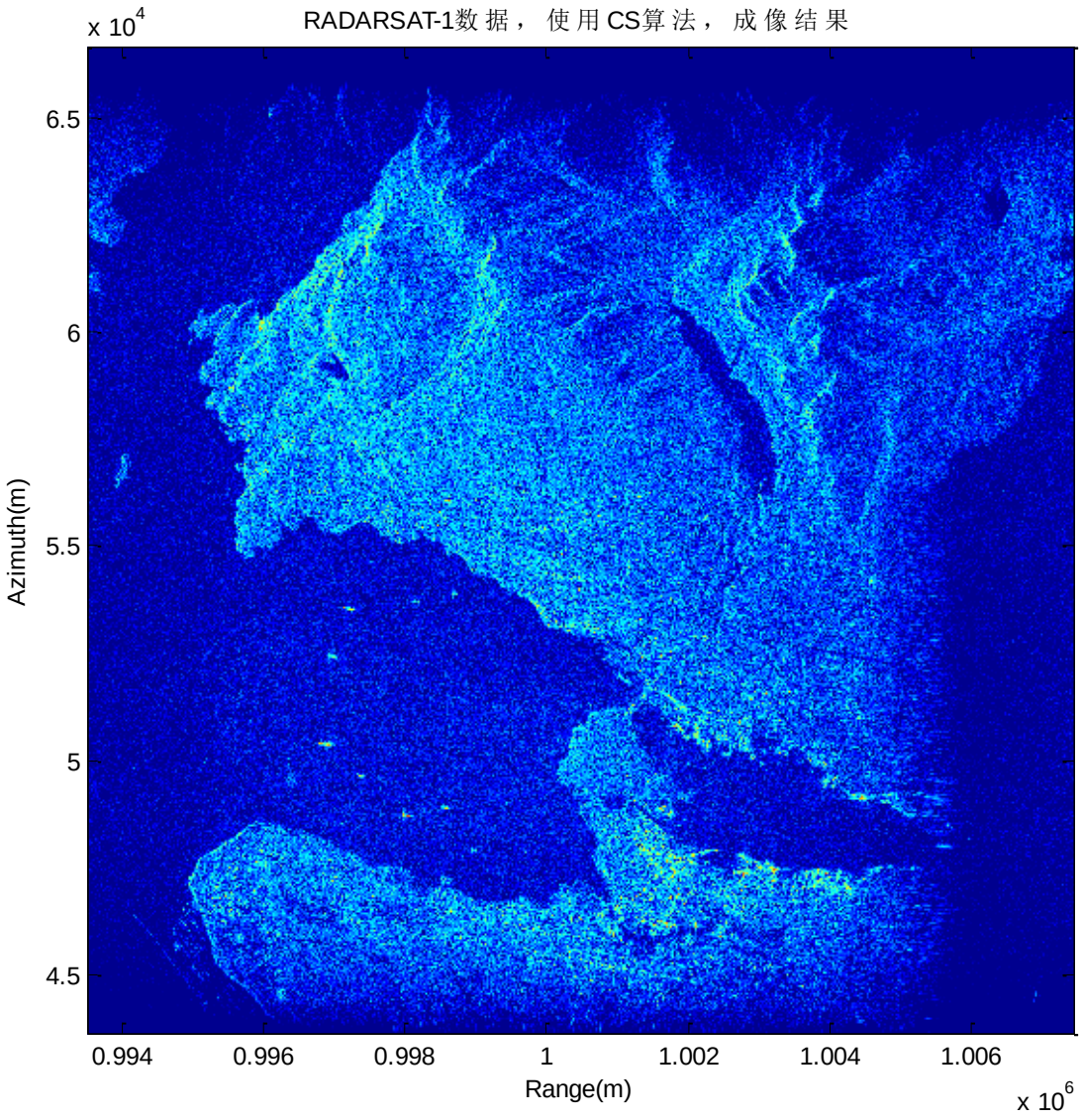
在上面结果的基础上，再进行上下半边的互换（用 fftshift 实现）

这个时候我们可以看到，在方位向上（上下方向）有一个很明显的间隙。这应该是由于补零造成的。也正是因为有了这样的补零，才使得最后的结果没有在方位向混在一起。但是这时候的上下关系依然是有些问题的。因为间隙下方到底部的图像应该在图像的最上方。

因此，我再通过移位的方式，将间隙最下方的图像手动移动到最上方。得到以下结果：

3) 将上下部分进行一定的移位

原来的图像的第 2900 行到最后一行应该在新图像的最开头，有以下结果：



这是我目前成出的最终结果

这是目前经过了以下几步，得到的最终结果：

a) 对原始数据补零；

b) 对成像结果进行向左移位：移动的点数和方法如前所述；

c) 对移位后的图像，再通过 fftshift 进行上下半边的互换；

d) 最后，还要将上下部分进行一定的移位（ 原来的图像的第 2900 行到最后一行应该在新图像的最开头 ）

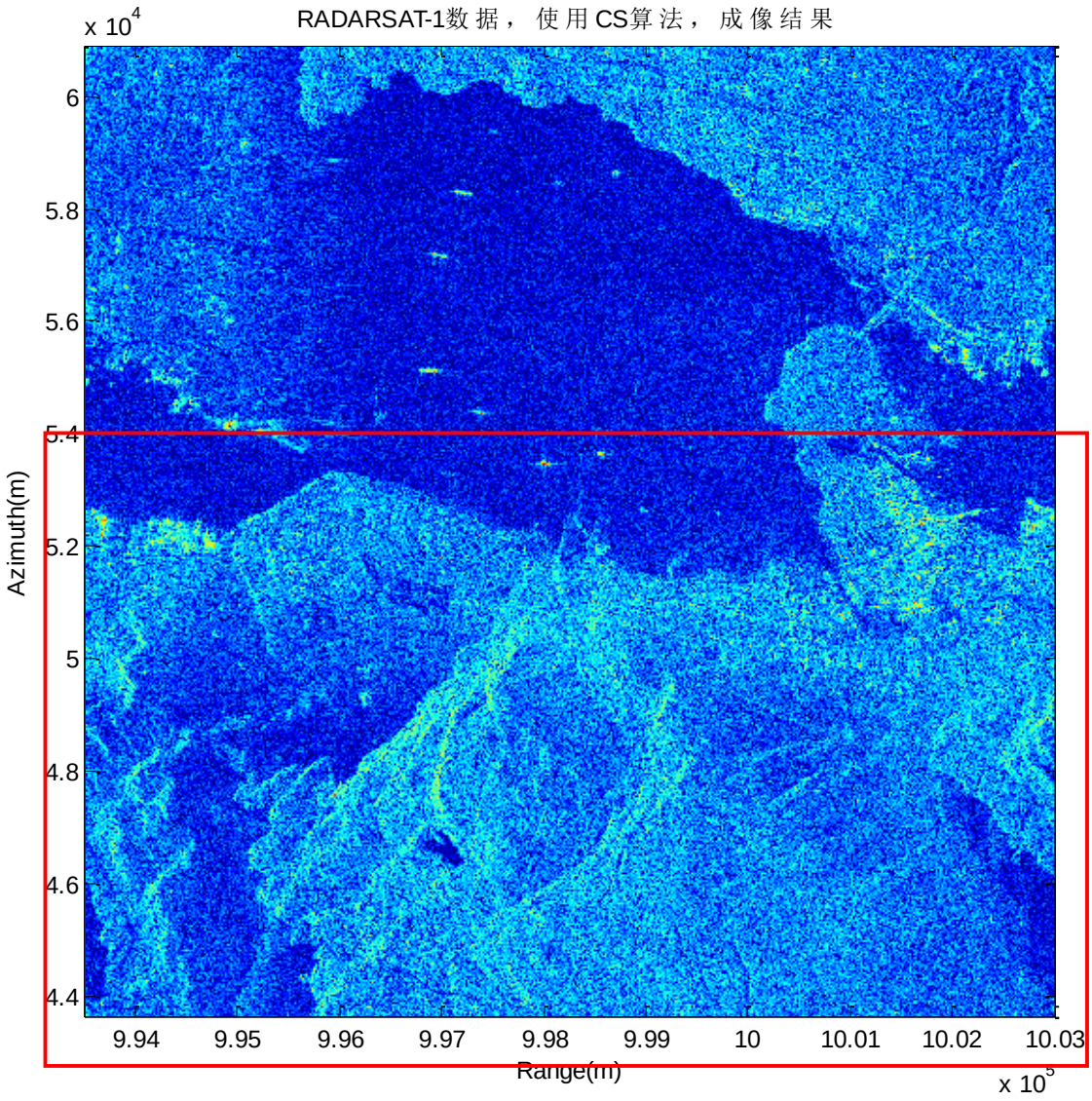
至此，得到了上面的图像。

为了与上面的结果进行对比，我再次进行仿真：

此时不对原始数据进行补零，而直接进行后续成像操作；

此外，除了对成像结果进行向左移位外，不进行 fftshift 等操作。（ 即上面的 c）和 d）都不进行 ）

得到的成像结果如下：



我们忽略上下位置关系或者左右位置关系的一些问题（因为这里也没有进行调整。不过其实这也就是来源于没有补零）。但是，更为显著地问题是，红色方框中的结果，在两方面都有问题：一方面是刚才提到的上下位置不对，有一部分应该在图像最上方，这其实也是由于没有补零导致的；另一方面，更重要的是，这时即使对该图像进行移位，也没有得到理想的结果，

因为这些图像根本就混在一起了（对比红色方框部分和上面补零后的成像结果，会发现，这时候不仅仅是上下位置不对，同时还有上下部分的混合）。

因此，补零是很重要的——成像之前，一定要对原始数据进行补零。

理论上我还没有分析，这时我接下去要做的工作。

注：

关于对原始数据补零的分析，还需要后续继续进行。

WD