# 实验一 输入右清晰闹钟图像 使用KSVD训练 之后恢复实验

## 原图



## 测试一 out\_put=IMout./Weight; 仿造ASR

vecOfMeans = mean(blocks(:,jj:jumpSize));

blocks(:,jj:jumpSize)= Dictionary\*Coefs + ones(size(blocks,1),1) \* vecOfMeans;

IMout(row:row+bb-1,col:col+bb-1)=IMout(row:row+bb-1,col:col+bb-1)+block;

Weight(row:row+bb-1,col:col+bb-1)=Weight(row:row+bb-1,col:col+bb-1)+ones(bb);

**out\_put=IMout./Weight;**



## 测试二 这个实验做得很没有意义 因为没有牵扯训练出来的字典

vecOfMeans = mean(blocks(:,jj:jumpSize));

blocks(:,jj:jumpSize)= Dictionary\*Coefs + ones(size(blocks,1),1) \* vecOfMeans;

out\_put=IMout./Weight;

IOut = (Image+0.034\*0.85)./(1+0.034\*0.85);

errors=Image-IOut;

errors最大值是五点多

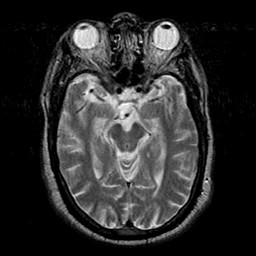


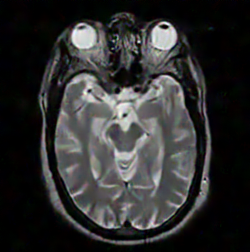
## 测试三 IOut = (Image+0.85\*IMout)./(1+0.85\*Weight);



这个测试说明原图根本就不能够恢复完整

## 测试四 使用JSR+AR直接测试 下图1为原图 使用IOut = (Image+0.85\*IMout)./(1+0.85\*Weight);恢复效果





此测试说明是可以恢复原图的 但是就是不知道是否和多聚焦原因有关 但是现在唯一不解的是为什么要通过IOut = (Image+0.85\*IMout)./(1+0.85\*Weight)获得结果 而不是通过像ASR中使用mean1 = mean(patch\_1(:));mean\_f=(mean1+mean2)/2;Patch\_f = Patch\_f + mean\_f;

## 测试五 直接使用JSR+AR dictionary.m测试闹钟·

### 闹钟

6554680.13832469



IOut=IMout./Weight;结果

error=sum(sum((image\_input1-y)^2));

值为6409012.28857142

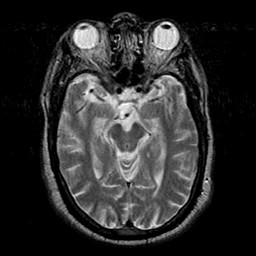
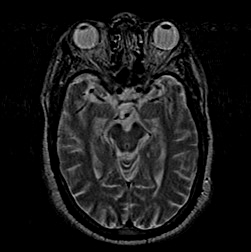


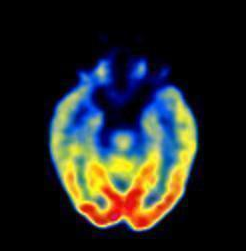
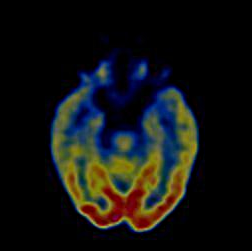
### MRI

IOut = (Image+0.034\*sigma\*IMout)./(1+0.034\*sigma\*Weight);

## 测试六 使用ASR测试恢复效果



直接对进行恢复效果都那么差说明可能原因就是一是因为字典原因 字典训练集是基于多聚焦那些数据集 二是由于字典不能精确表示原图像 下一步工作安排就是如何将图像恢复效果更好 步骤就是先进行IHS变换，进行图像块状分类 再进行分类。

# 实验二 IHS变换 论文中给出的转换矩阵无法复原

function [OUTPUT,FORRI,AA,BB]=RGB2IHS(input)

[M,N,Q]=size(input);

OUTPUT=zeros(M,N,Q);

FORRI=zeros(M,N,Q);

v11=zeros(3,1);

% AA=[1/3,1/3,1/3;

% -sqrt(2)/6,-sqrt(2)/6,-2\*(sqrt(2)/6); %RGB－>IHS正变换矩阵

% 1/sqrt(2),-1/sqrt(2),0];

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BB=[1,-1/sqrt(2),1/sqrt(2);

% 1,-1/sqrt(2),-1/sqrt(2); %IHS->RGB逆变换矩阵

% 1,sqrt(2),0];

AA=[1/sqrt(3),1/sqrt(3),1/sqrt(3);

1/sqrt(6),1/sqrt(6),-2/sqrt(6); %RGB－>IHS正变换矩阵

1/sqrt(2),-1/sqrt(2),0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BB=[1/sqrt(3),1/sqrt(6),1/sqrt(2);

1/sqrt(3),1/sqrt(6),-1/sqrt(2); %IHS->RGB逆变换矩阵

1/sqrt(3),-2/sqrt(6),0];

for i=1:M

for j=1:N

v11(1)=double(input(i,j,1));

v11(2)=double(input(i,j,2));

v11(3)=double(input(i,j,3));

u1=AA\*v11;

OUTPUT(i,j,1)=u1(1);

OUTPUT(i,j,2)=1/(tan(u1(3)/u1(2)));

OUTPUT(i,j,3)=sqrt(u1(2)^2+u1(3)^2);

u2=BB\*u1;

FORRI(i,j,1)=u2(1);

FORRI(i,j,2)=u2(2);

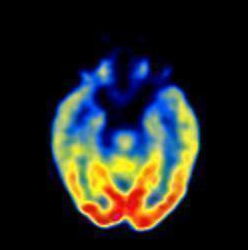
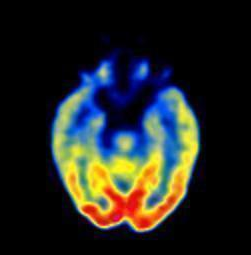
FORRI(i,j,3)=u2(3);

end

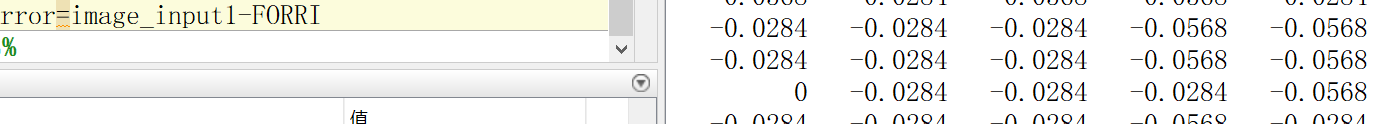
end

end

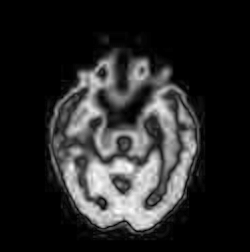
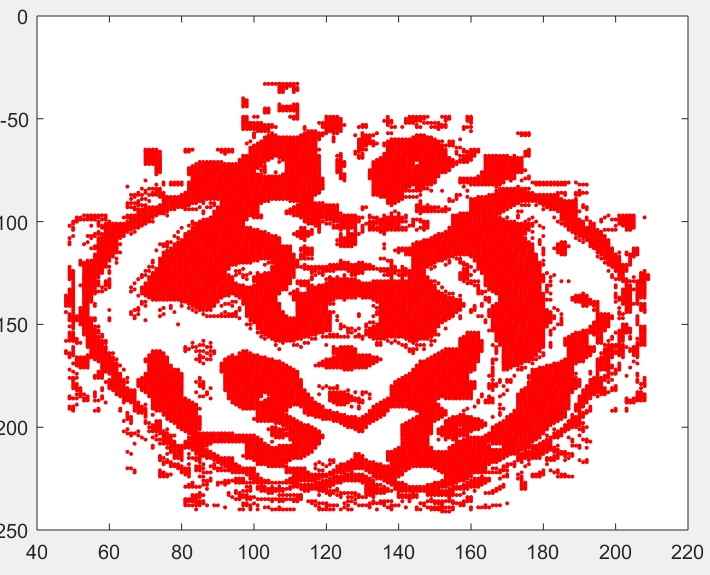
下图是复原前和复原后

还是有些略微差别 可能是由于根号原因

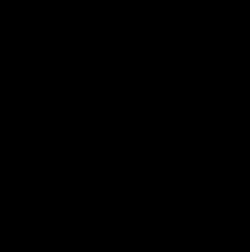


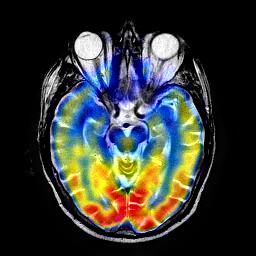
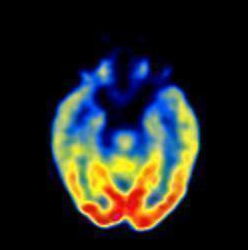
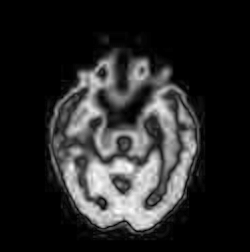
分别是IHS分量





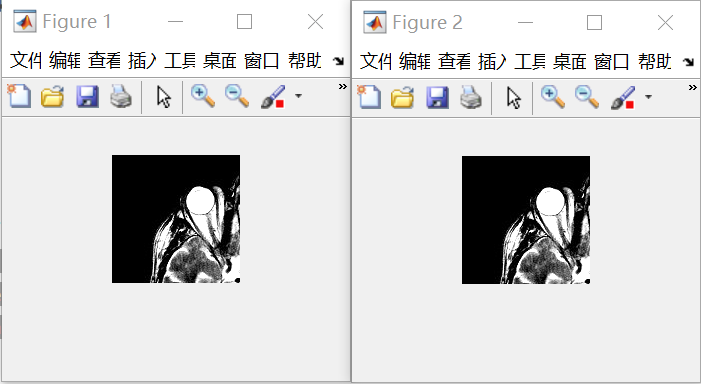


分量3全部为零 由于灰度图像三个分量对应相等



现在主要就是设计一个实验能够验证对I分量图像进行复原

# 实验三 梯度信息得到字典 然后进行恢复 kk=size(X,1) 1是列数 2是行数

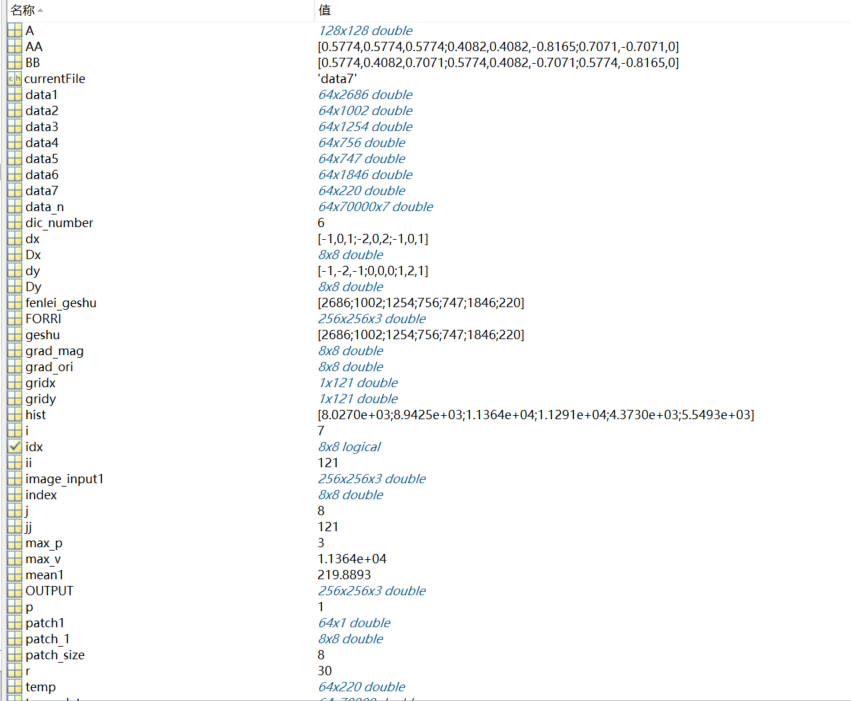


通过对MRI亮度分量进行分成8X8小块然后使用KSVD算法 发现其恢复能力较强，但是此实验唯一不足就是没有将梯度角属于第七类的加入，理论上来说，加上加上效果不会比这个差，下一步计划就是通过对多样本学习，或者将已经学习好的字典中选择效果最好的（这个步骤存疑，就是不知道是单单训练一张图像，然后训练好一张图像之后将字典替代DCT字典加入，还是将所有样本进行分类统一训练效果好），下一步实验应该是验证联合字典对PET和MRI分量的表征能力。本实验代码见TEST3TO1。

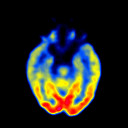
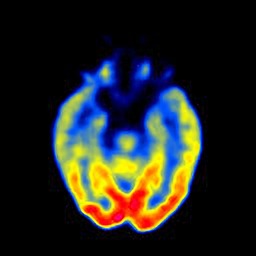
# 实验四 联合字典对PET和MRI分量表征能力测试

## 测试一 训练数据集选取 选取每种疾病SF越大的 TEST4 PET取得SF前八张 MRI取得14类SF排第一的

## 测试二 将MRI和PET分别进行HIS变换 将I分量进行字典训练 看效果



发现个问题 就是PET图像是128的 通过放大之后效果 可以说是个好方法 可以试试用在图像处理中



## 测试三 将两张PET和两张MRI的I分量合并一次性就行字典训练 看看合成效果 现在主要就是还有一点改进的地方就是将PET进行无损放大 而不是简单的imresize

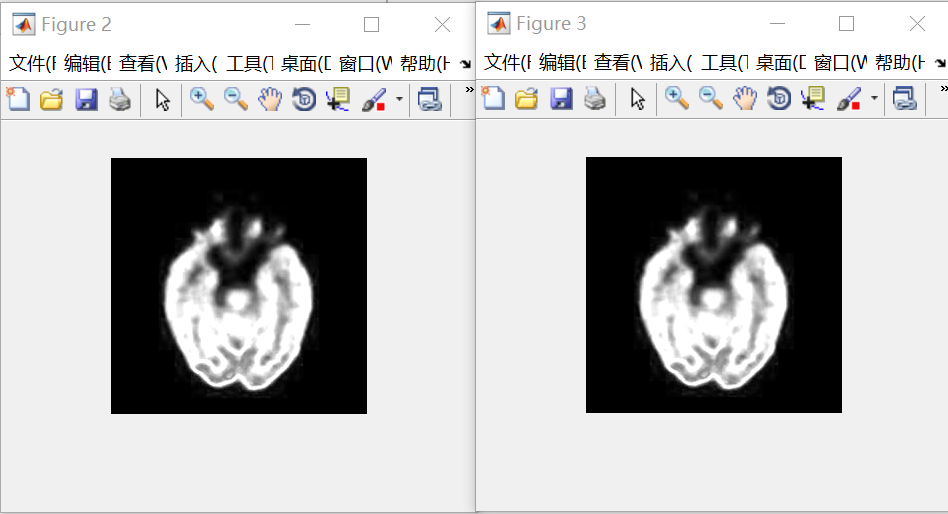


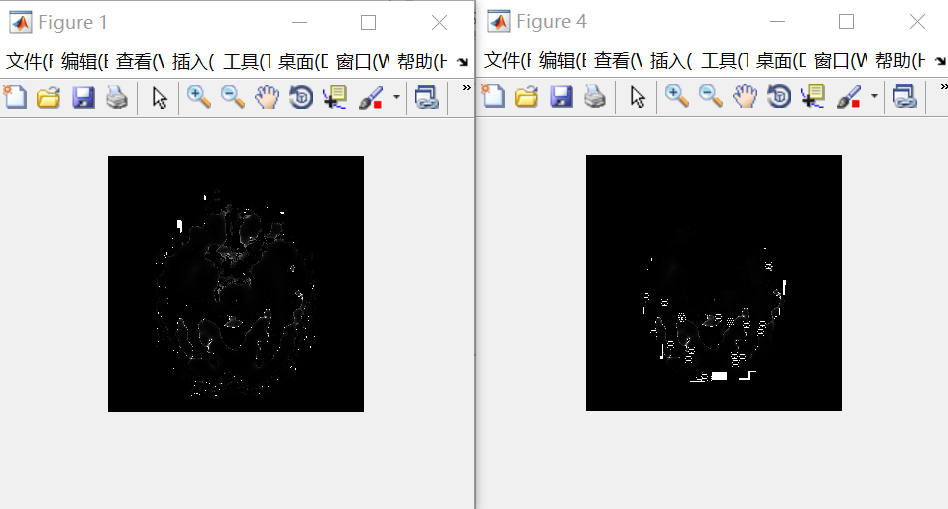


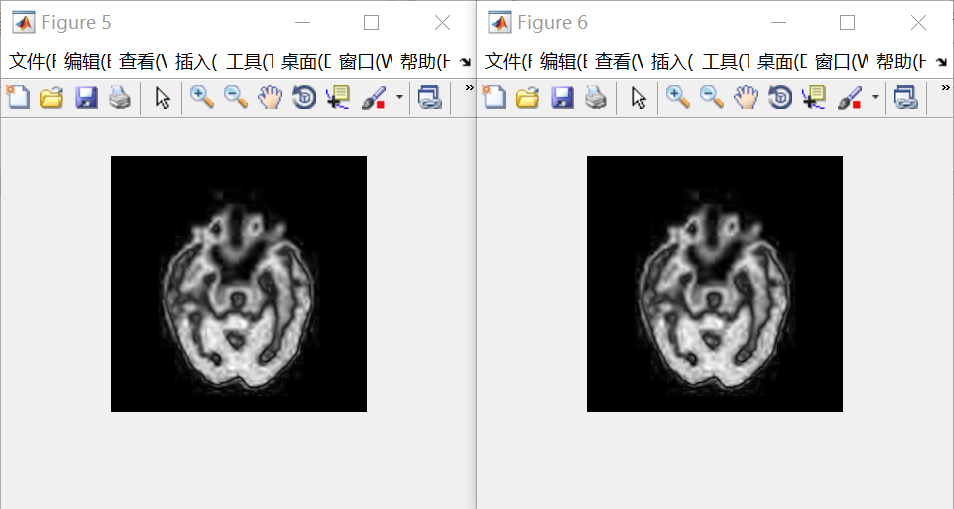
可以看到恢复效果还是不错的 这还只是在单单使用128X128训练情况下

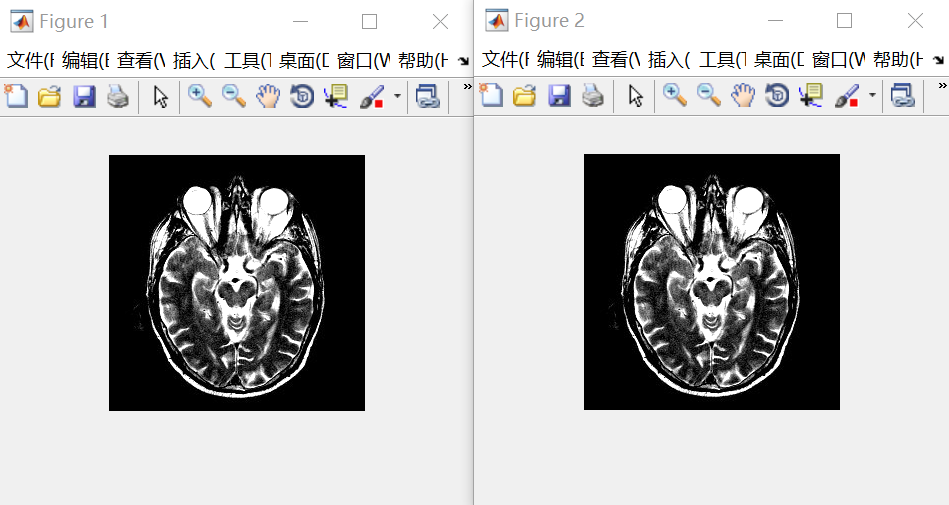
## 测试四 写一个综合程序 然后根据输入的四张图像分别得到I H S分量 分别进行存储得到MAT文件

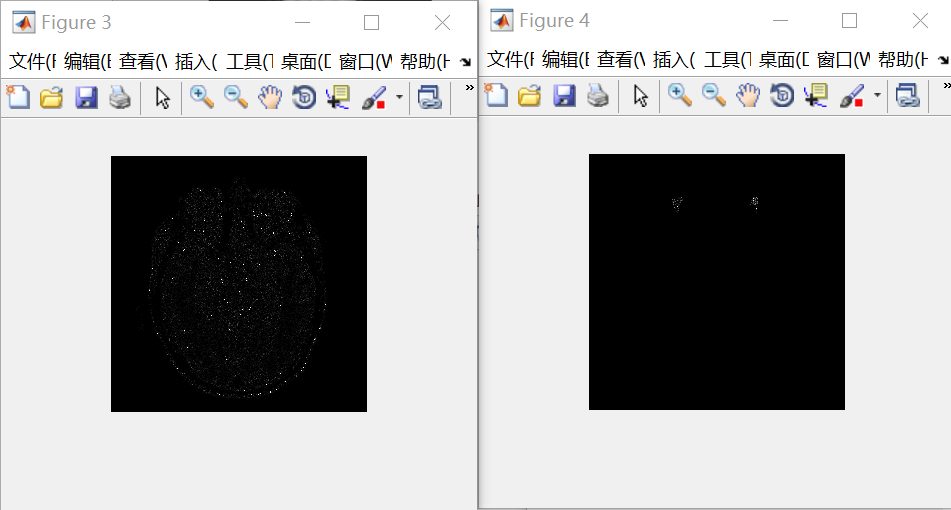
# 实验五 通过对PET图像I H S分量分别进行梯度分类 字典训练 发现其表现能力较差对于H分量 由于其稀疏性

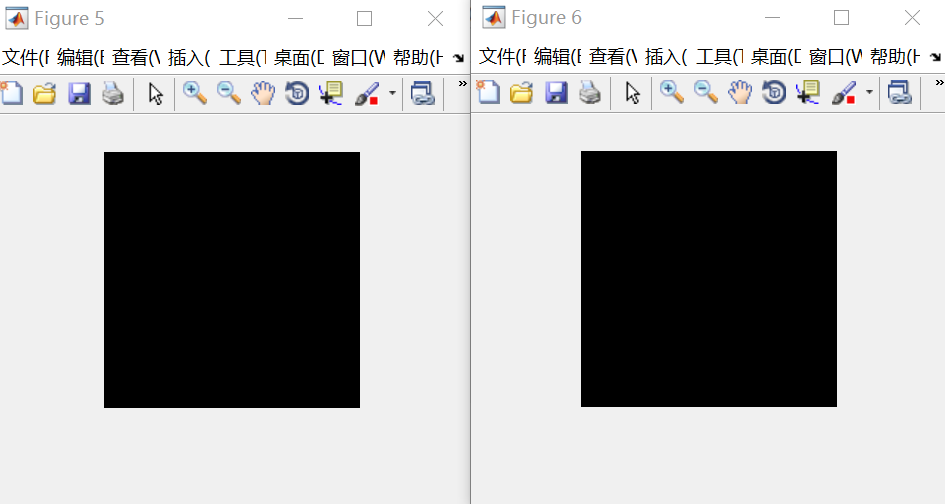


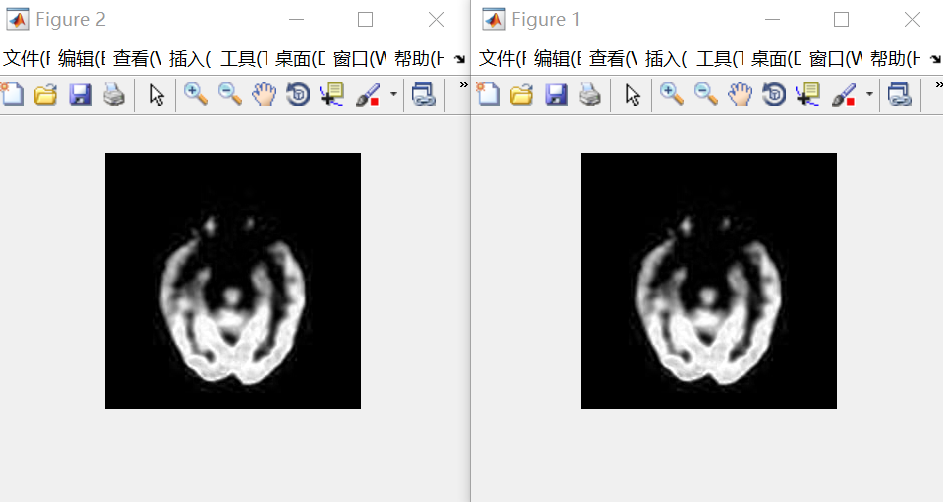


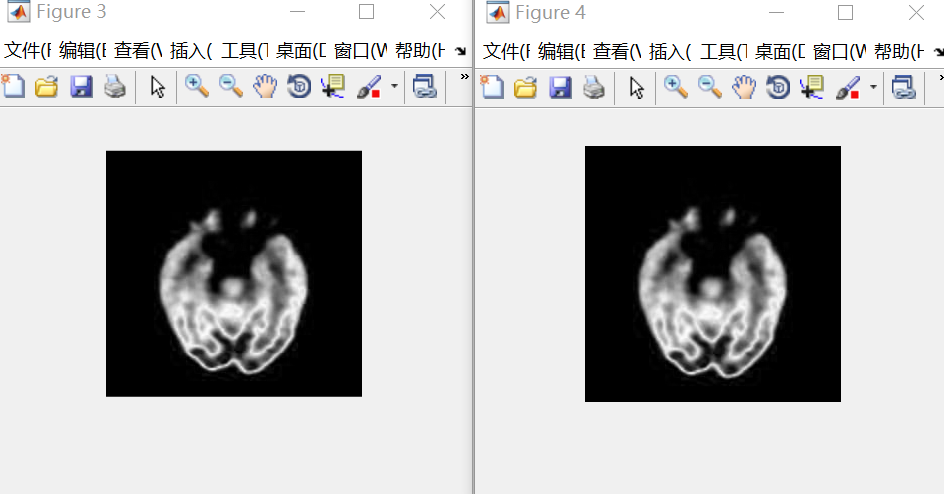


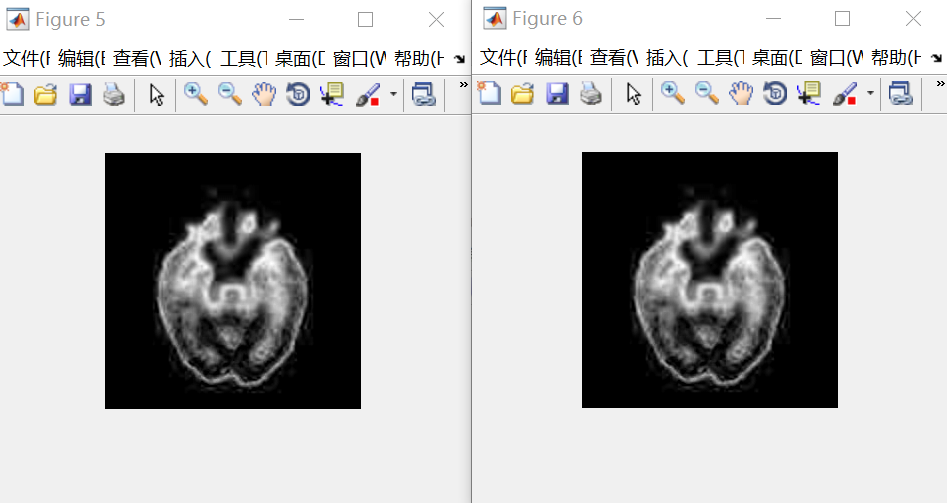






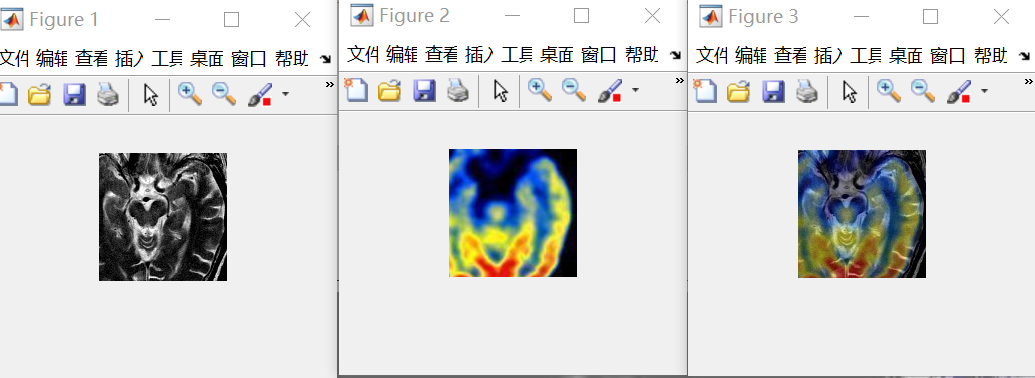






# 实验六 通过对小块进行I H S变换之后 获取亮度权重 然后进行用RGB分别进行表示合成

## 测试一 采用加权方式一的话 效果就这样 不太好



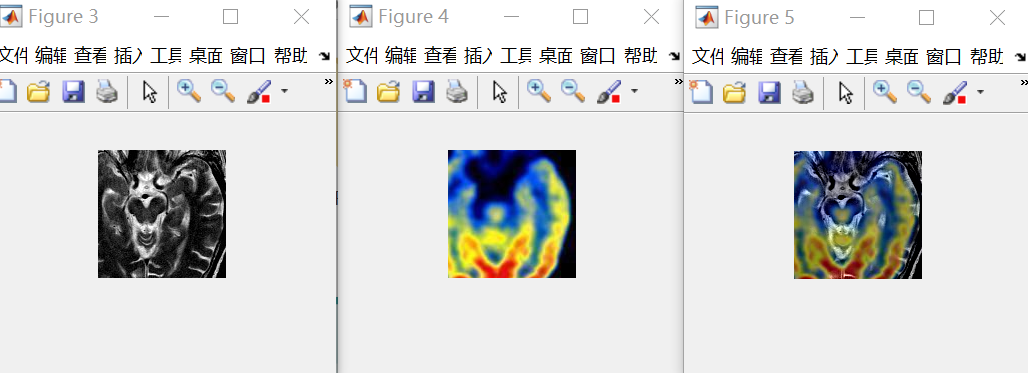
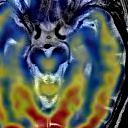
% [F1,w1,max\_p1]=USE\_DICTIONARY(patch\_1,0);

% [F2,w2,max\_p2]=USE\_DICTIONARY(patch\_1,0);

这个测试要做过一次 因为程序写错了 此处使用的都是patch\_1，但是神奇的是为什么程序写错了还是会出现融合效果

关于亮度分析就是由于是取各自的部分还进行了加权 导致就算是很大的加权一下其实也很小了

## 测试二 加权方式采用了最大值原则



这里之所以出现这种情况是因为I分量中的凸出部分和彩色I分量相比 其实没有多大的优势 因为彩色图像的I分量可能也很大 导致二者不相上下 就无法比较出来了 所以一种更好的方法需要先通过实验验证 能够将二者在一起比较时 能够准确把I分量识别出来

## 测试三 做一次测试 在测试一的基础上分别得出完整复原数据 再进行加权 因为通过（mean1+mean2）/2综合加权方式容易导致亮度不够 在测试一可以看出总体效果还是不错的 实在不行就用区域生长的办法

# 改进点

## 1 字典训练可以不用每次都是DCT重新生成 可以用训练好的字典代入

## 2 PET图像放大效果不适用imresize

## 3 A=A(32:size(A,1)/2+32,32:size(A,2)/2+32);为了减少运算量梯度分类选取了部分

## 4 currentFile = sprintf('fenlei\_data%d.mat',i); load(currentFile);不宜放在函数中被调用

## 5 对源数据求最大无关组 分别获取稀疏表示系数 之后再进行解坐标向量方式 加权稀疏系数 这样就可以获取系数 减少时间