

图像重建 fdk 大作业

周浩

2015011685

FDK 算法

算法分为三步：


1. 投影数据修正。
2. 卷积滤波。
3. 反投影。

Weight: $\frac{R}{\sqrt{R^2 + a^2 + b^2}} = \cos \gamma \cos k$






Filtration: $\tilde{p}(\beta, a, b) = \left(\frac{R}{\sqrt{R^2 + a^2 + b^2}} g(\beta, a, b) \right) \otimes h^p(a)$

Weighted backprojection:

$$f(x, y, z) = \frac{1}{2} \int_0^{2\pi} \frac{R^2}{(R + x \cos \beta + y \sin \beta)^2} \tilde{p}(\beta, a(x, y, \beta), b(x, y, z, \beta)) d\beta$$

 $a = \frac{R(-x \sin \beta + y \cos \beta)}{R + x \cos \beta + y \sin \beta}, \quad b = \frac{zR}{R + x \cos \beta + y \sin \beta}$

以上三步封装为三个函数 funcWeightProjectData、funcFilter、funcBackprojectionFdk。

 funcBackprojectionFdk.m
 funcFilter.m
 funcWeightProjectData.m
 hw1.m
 readrawdata1.m

```
%%
tic
for i=1:beta_num
    beta=(i-1)*pi/180;
    pro_beta=squeeze(p(:, :, i));
    %    pro_beta=reshape(pro_beta, [N_d, N_d]);
    %=====加权=====
    weight_project_beta=funcWeightProjectData(pro_beta, N_d, SDD, detector_channel_size);
    %=====卷积滤波=====
    filtered_projection=funcFilter(weight_project_beta, fh_RL, N_d);
    %=====反投影=====
    rec=rec+funcBackprojectionFdk(filtered_projection, pixelsize, SOD, beta, beta_num, N, detector_channel_size, SDD);
    t=toc;
    fprintf('theta: %d, timecost: %d s\n', i-1, t);
end
rec=rec*SDD/SOD; %从虚拟探测器映射到实际探测器
```

重建结果,

非中心平面上头模型的重建结果如图 1 所示，中心平面上的重建结果，图像窗都是 [1,1.05]。

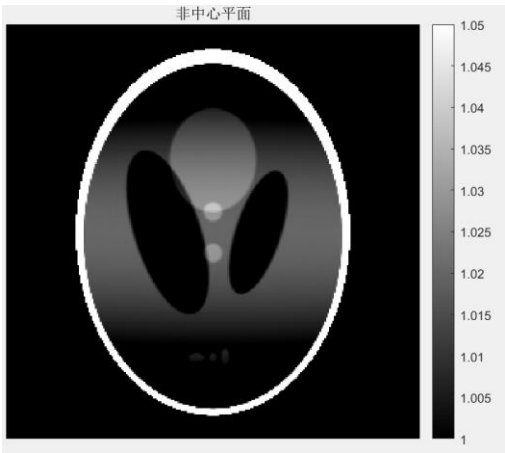


图 1.非中心平面重建结果

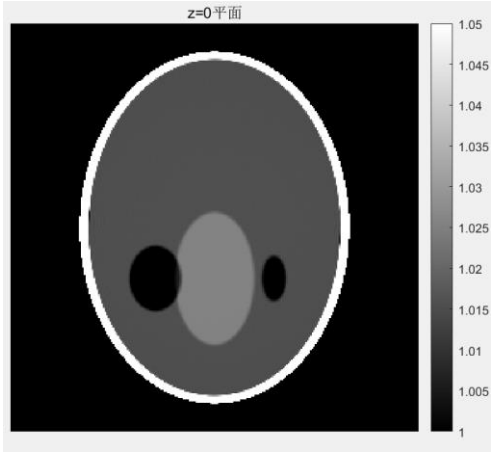


图 2.中心平面的重建结果

为了观察重建的准确性，选择了头模型和重建模型的中心平面的切片，通过观察剖面图 profile，可以看出重建结果是正确的，如图 3 所示。

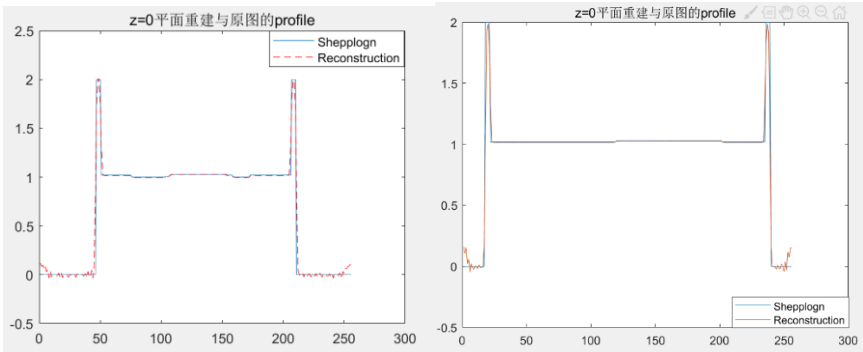


图 3.中心切片重建图与原图的 profile 对比