# 附录

## Distance\_of\_CT.m

|  |
| --- |
| load data.mat data\_2  data = data\_2;  data(:, 14 : 108) = 0;  for i = 1 : 13      data(1 : 370, i) = 0;  end  for i = 109 : 180      data(112 : 512, i) = 0;  end  b=[];  for i = 1 : 180  b = [b length(find(data(:,i)))];  end  b = [b(1:13) b(109:180)];  d = 8 / mean(b);  disp(d); |

## cenfunc.m

|  |
| --- |
| function res = cenfunc(n1\_1, n1\_2, n2\_1, n2\_2)  % origin data  d = 0.2774;  a = 40;  b = 15;  c = sqrt(a^2 - b^2);  e = c / a;  n3 = 279;    % 1st line  y0\_2 = (a^2 - (2 \* a \* b / (n1\_1 \* d))^2) / e^2;  x0\_2 = (1 - y0\_2 / a^2) \* b^2;  y0 = sqrt(y0\_2);  x0 = sqrt(x0\_2);  y1\_k = - x0 \* a^2 / (b^2 \* y0);  y1\_bias = a^2 / y0;  y1\_bias = y1\_bias + n1\_2 \* d / sin(pi / 2 + atan(y1\_k));    % 2nd line  y0\_2 = (a^2 - (2 \* a \* b / (n2\_1 \* d))^2) / e^2;  x0\_2 = (1 - y0\_2 / a^2) \* b^2;  y0 = -sqrt(y0\_2);  x0 = sqrt(x0\_2);  y2\_k = - x0 \* a^2 / (b^2 \* y0);  y2\_bias = a^2 / y0;  y2\_bias = y2\_bias + n2\_2 \* d / cos(atan(y2\_k));    % 3rd line  y3\_k = 0;  y3\_bias = n3 \* d;    % move  res = [y2\_bias/sqrt(y2\_k^2+1)-y1\_bias/sqrt(y1\_k^2+1) y2\_bias/sqrt(y2\_k^2+1)-y3\_bias/sqrt(y3\_k^2+1)]/...      [y1\_k/sqrt(y1\_k^2+1)-y2\_k/sqrt(y2\_k^2+1) y3\_k/sqrt(y3\_k^2+1)-y2\_k/sqrt(y2\_k^2+1);       1/sqrt(y1\_k^2+1)-1/sqrt(y2\_k^2+1) 1/sqrt(y3\_k^2+1)-1/sqrt(y2\_k^2+1)];  end |

## Center\_specific.m

|  |
| --- |
| %  % n1\_1 = 209.5;  % n1\_2 = 140.5;  % n2\_1 = 210.5;  % n2\_2 = 401.5;  %  % cenfunc(n1\_1, n1\_2, n2\_1, n2\_2)  %    % origin data  d = 0.2774;  a = 40;  b = 15;  c = sqrt(a^2 - b^2);  e = c / a;    % 1st line  % parameter  n1\_1 = 209.5; % data\_2(:, 13) num of tuoyuan  n1\_2 = 140.5;    y0\_2 = (a^2 - (2 \* a \* b / (n1\_1 \* d))^2) / e^2;  x0\_2 = (1 - y0\_2 / a^2) \* b^2;  y0 = sqrt(y0\_2);  x0 = sqrt(x0\_2);  y1\_k = - x0 \* a^2 / (b^2 \* y0);  y1\_bias = a^2 / y0;  fprintf('y1 = %f\*x + %f\n', y1\_k, y1\_bias);  % y1 = -1.1046\*x + 43.2957;    y1\_bias = y1\_bias + n1\_2 \* d / sin(pi / 2 + atan(y1\_k));  fprintf('y1\_edited = %f\*x + %f\n', y1\_k, y1\_bias);  % y1\_edited = -1.104572\*x + 101.367747;    % memory clear  clear n1\_1 n1\_2 x0\_2 y0\_2 y0 x0    % 2nd line  % parameter  n2\_1 = 210.5; % data\_2(:, 109) num of tuoyuan  n2\_2 = 401.5;    y0\_2 = (a^2 - (2 \* a \* b / (n2\_1 \* d))^2) / e^2;  x0\_2 = (1 - y0\_2 / a^2) \* b^2;  y0 = -sqrt(y0\_2);  x0 = sqrt(x0\_2);  y2\_k = - x0 \* a^2 / (b^2 \* y0);  y2\_bias = a^2 / y0;  fprintf('y2 = %f\*x + %f\n', y2\_k, y2\_bias);  % y2 = 1.0916\*x - 43.2214;    y2\_bias = y2\_bias + n2\_2 \* d / cos(atan(y2\_k));  fprintf('y2\_edited = %f\*x + %f\n', y2\_k, y2\_bias);  % y2\_edited = 1.091554\*x + 208.099009;    % memory clear  clear n2\_1 n2\_2 x0\_2 y0\_2 y0 x0    % 3rd line  % parameter  n3 = 279;    y3\_k = 0;  y3\_bias = n3 \* d;  fprintf('y3 = %f\*x + %f\n', y3\_k, y3\_bias);  % y3 = 0.000000\*x + 77.394600;    % test  %y3\_bias = 0;    % plot  x = -100 : 0.1 : 100;  y1 = y1\_k \* x + y1\_bias;  y2 = y2\_k \* x + y2\_bias;  y3 = y3\_k \* x + y3\_bias;  figure  %axis([-50 50 -50 50])  plot(x, y1);  hold on  plot(x, y2);  %hold on  plot(x ,y3);    % move  % res = [y3\_bias-y2\_bias y3\_bias-y1\_bias] / [y2\_k-y3\_k y1\_k-y3\_k; -2 -2];  res = [y2\_bias/sqrt(y2\_k^2+1)-y1\_bias/sqrt(y1\_k^2+1) y2\_bias/sqrt(y2\_k^2+1)-y3\_bias/sqrt(y3\_k^2+1)]/...      [y1\_k/sqrt(y1\_k^2+1)-y2\_k/sqrt(y2\_k^2+1) y3\_k/sqrt(y3\_k^2+1)-y2\_k/sqrt(y2\_k^2+1);       1/sqrt(y1\_k^2+1)-1/sqrt(y2\_k^2+1) 1/sqrt(y3\_k^2+1)-1/sqrt(y2\_k^2+1)];    x\_cen = res(1);  y\_cen = res(2);  fprintf('x\_cen = %f, y\_cen = %f\n', x\_cen, y\_cen);    % plot  x = -15 : 0.1 : 15;  plot(x, sqrt((1 - x .^ 2 / 15^2) \* 40^2));  plot(x, -sqrt((1 - x .^ 2 / 15^2) \* 40^2));  scatter(x\_cen, y\_cen); |

## Center\_data.m

|  |
| --- |
| %load data.mat data\_2    tmp1 = [];  for i = 1 : 13      tmp1 = [tmp1 data\_2(:, i)];  end    for i = 1 : 13      for j = 371 : 512          tmp1(j, i) = 0;      end  end    for i = 1 : 512      for j = 1 : 13          if tmp1(i, j) ~= 0              tmp1(i, j) = 1;          end      end  end    n1\_1 = sum(tmp1);    n1\_2 = [];  for i = 1 : 13      for j = 512 : -1 : 1          if tmp1(j,i) ~= 0              n1\_2 = [n1\_2 j];              break;          end      end  end  n1\_2 = 512 - n1\_2 + 1;    %%  tmp1=[];  for i = 140 : 152      tmp1 = [tmp1 data\_2(:, i)];  end    for i = 1 : 13      for j = 1 : 80          tmp1(j, i) = 0;      end  end    for i = 1 : 512      for j = 1 : 13          if tmp1(i, j) ~= 0              tmp1(i, j) = 1;          end      end  end    n2\_1 = sum(tmp1);    n2\_2 = [];  for i = 1 : 13      for j = 1 : 512          if tmp1(j, i) ~= 0              n2\_2 = [n2\_2 j];              break;          end      end  end  n2\_2 = 512 - n2\_2 + 1;    %disp(n1\_1);  %disp(n1\_2);  %disp(n2\_1);  %disp(n2\_2);    r = [];  for i = 1 : 13      for j = 1 : 13          tmp3 = cenfunc(n1\_1(i), n1\_2(i), n2\_1(j), n2\_2(j));          if imag(tmp3) == 0              r = [r; tmp3];          end      end  end      figure  scatter(r(:, 1), r(:, 2));  hold on    % scatter(-9.5471, -6.9475);    x\_cen = mean(r(:, 1));  y\_cen = mean(r(:, 2));  scatter(x\_cen, y\_cen);  disp([x\_cen y\_cen]);  axis([-50 50 -50 50]) |

## Mark\_Point.m

|  |
| --- |
| function mark\_point\_matrix = Mark\_Point(projection\_matrix)  mark\_point = zeros(size(projection\_matrix));  for i = 1 : size(projection\_matrix, 2)      if projection\_matrix(1,i) == 0          flag = 0;      else          flag = 1;      end      for j = 2:size(projection\_matrix, 1)          if projection\_matrix(j,i) == 0              next\_flag = 0;          else              next\_flag = 1;          end          if next\_flag ~= flag              mark\_point(j,i) = j;          end          flag = next\_flag;      end  end  mark\_point\_matrix = mark\_point;  end |

## GetScale.m

|  |
| --- |
| function scale = GetScale(length\_projection)  scale\_list = [];  for i = 1 : size(length\_projection, 2)      if length\_projection(1, i) == 0          scale\_pre = 0;      elseif length\_projection(2, i) == 0          scale\_pre = length\_projection(1, i) ./ 28.83;      elseif length\_projection(1, i) <= length\_projection(2, i)          scale\_pre = length\_projection(2, i) ./ 28.83;      else          scale\_pre = length\_projection(1, i) ./ 28.83;      end      scale\_list = [scale\_list, scale\_pre];  end  scale = scale\_list;  end |

## GetLengthOfProjection.m

|  |
| --- |
| function length\_projection = GetLengthOfProjection(mark\_point\_matrix)  length\_vector = zeros(2,size(mark\_point\_matrix, 2));  for i = 1:size(mark\_point\_matrix, 2)      not\_zero = [];      for j = 1:size(mark\_point\_matrix, 1)          if mark\_point\_matrix(j,i) ~= 0              not\_zero = [not\_zero, mark\_point\_matrix(j, i)];          end      end      if isempty(not\_zero)          length\_vector(:, i) = [0; 0];      elseif length(not\_zero) == 2          length\_vector(:, i) = [not\_zero(2) - not\_zero(1); 0];      else          length\_vector(:, i) = [not\_zero(2) - not\_zero(1); not\_zero(4) - not\_zero(3)];      end  end  length\_projection = length\_vector;  end |

## AngleSolve.m

|  |
| --- |
| function angle = AngleSolve(s1)  syms pro\_angle  syms equ  equ = 10 \* sqrt(((64 + 9 \* (tan(pro\_angle))^2)) / (1 + (tan(pro\_angle))^2)) - s1;  angle\_result = roundn(eval(solve(equ, 'pro\_angle')), -4);  %angle\_result = rad2deg(angle\_result)  angle = abs(angle\_result(1));  end |

## AngleMain.m

|  |
| --- |
| function angle = AngleSolve(s1)  syms pro\_angle  syms equ  equ = 10 \* sqrt(((64 + 9 \* (tan(pro\_angle))^2)) / (1 + (tan(pro\_angle))^2)) - s1;  angle\_result = roundn(eval(solve(equ, 'pro\_angle')), -4);  %angle\_result = rad2deg(angle\_result)  angle = abs(angle\_result(1));  end |

## rebuild.m

|  |
| --- |
| % REBUILD Concolution Backprojection (CEP).  %    REBUILD(OriginData, Angles) returns the matrix of absorptance.    function ret = rebuild(OriData, Theta)    % calculate fundamental image data  Num\_Cam = size(OriData, 1);  Num\_Ang = length(Theta);  MidIdx = (Num\_Cam + 1) / 2;  Rad = (pi / 180) \* Theta;  ret = zeros(Num\_Cam, Num\_Cam);    % coordinate settings  [X, Y] = meshgrid(1 : Num\_Cam, 1 : Num\_Cam);  X = X - MidIdx;  Y = Y - MidIdx;    % 1-D filter  FiltData = myfilter(OriData);    % backprojection  for i = 1 : Num\_Ang      % calculate spots address to be added filtered data      Idx = round(MidIdx + X \* sin(Rad(i)) - Y \* cos(Rad(i)));        % accumulate      tmp = zeros(Num\_Cam);      tmpIdx = find((Idx > 0) & (Idx <= Num\_Cam));      tmp(tmpIdx) = FiltData(Idx(tmpIdx), i);      ret = ret + tmp;  end  ret = ret / Num\_Ang;  end |

## myfilter.m

|  |
| --- |
| function f = myfilter(PR)    a = 1;  [Length, Count] = size(PR);  w = [-pi : (2 \* pi) / Length : pi - (2 \* pi) / Length];    rn1 = abs(2 / a \* sin(a .\* w ./ 2));  rn2 = sin(a .\* w ./ 2);  rd = (a \* w) ./ 2;  r = rn1 \* (rn2 / rd)^2;    f = fftshift(r);  for i = 1:Count      IMG = fft(PR(:, i));      fimg = IMG .\* f';      f(:, i) = ifft(fimg);  end  f = real(f);  end |

## CTplot.m

|  |
| --- |
| function ImgData = CTplot(OriData, Theta, x0, y0, d, option)    if option == 1      oc = sqrt(x0^2 + y0^2);      koc = y0 / x0;        for i = 1 : size(OriData, 2)          tmp = round(oc \* sin(Theta(i) - atan(abs(koc))) / d);          if tmp >= 0              for j = size(OriData, 1) : -1 : 1 + tmp                  OriData(j, i) = OriData(j - tmp, i);              end          else              for j = 1 : size(OriData, 1) + tmp                  OriData(j, i) = OriData(j - tmp, i);              end          end      end  end    ImgData = rebuild(OriData, rad2deg(Theta));  figure;  imagesc(ImgData);  end |

## AbCount.m

|  |
| --- |
| function Ab\_data = AbCount(ImgData, Ab\_std)  %ABCOUNT counts the absorption rate(256\*256) of ImgData(512\*512).    Ab\_data = zeros(256);  for i = 1 : 2 : 512      for j = 1 : 2 : 512          Ab\_data((i + 1) / 2, (j + 1) / 2) = ImgData(i, j) + ...              ImgData(i + 1, j) + ImgData(i, j + 1) + ImgData(i + 1, j + 1);      end  end  Ab\_data = (Ab\_data - Ab\_std) / k;    end |

## Count\_Ab\_std.m

|  |
| --- |
| %ImgData = CTplot(data\_2,a,x\_cen,y\_cen,0.2774,1);  Ab\_std= zeros(256);  for i=1:2:512      for j=1:2:512          Ab\_std((i+1)/2,(j+1)/2) = ImgData(i,j)+ImgData(i+1,j)+ImgData(i,j+1)+ImgData(i+1,j+1);      end  end  temp = Ab\_std(Ab\_std>2);  k = sum(temp)/size(temp,1);  Ab\_std = Ab\_std- data\_1\*k; |

## Switch\_projection.m

|  |
| --- |
| function [pro\_after\_switch, pro\_ref] = Switch\_projection(pro, pro\_ref)  if size(pro) ~= size(pro\_ref)      fprintf('different sizes!\n');  else      project = reshape(pro, 1, size(pro,1) .\* size(pro, 2));      project\_ref = reshape(pro\_ref, 1, size(pro\_ref,1) .\* size(pro\_ref, 2));      for i = 1:size(project\_ref, 2)          if project\_ref(i) ~= 0              head\_ref = i;              break          end      end      for j = 1:size(project\_ref, 2)          k = size(project\_ref,2) - j + 1;          if project\_ref(k) ~= 0              tail\_ref = k;              break          end      end      for a = 1:size(project,2)          if project(i) ~= 0              head = i;              break          end      end      for b = 1:size(project, 2)          c = size(project, 2) - b + 1;          if project(c) ~= 0              tail = c;              break          end      end |