# LAMPIRAN A

**Lampiran A : Listing Program Pembuatan *Codebook* FCM tersegmentasi (clustering)**

```
close all;
clear all;
clc;
tic
for iterRow=1:4
   for iterCol=1:4
      fprintf('iRow,iCol=%d,%d\n',iterRow,iterCol);
      clear MI;
      clear I;
      clear bRow;
      for imgIter=1:3
         % open image file
         if (imgIter==1)
           strFile = 'XrayA.bmp';
           I=imread(strFile);
           fprintf('Citra: %s\n',strFile);
%             display('picture: XrayA.bmp')
         elseif (imgIter==2)
           strFile = 'XrayB.bmp';
           I=imread(strFile);
           fprintf('Citra: %s\n',strFile);
%             display('picture: XrayD.bmp')
         elseif (imgIter==3)
           strFile = 'XrayH1.bmp';
           I=imread(strFile);
           fprintf('Citra: %s\n',strFile);
%             display('picture: XrayH1.bmp')
         end
```

```
vHDim = 4;              % horz dimension
vVDim = 4;              % vert dimension
% Original image size
[HeightOri,WidthOri]=size(I);

% Adjust image size according to vector dimesion
VMin=mod(WidthOri,vHDim);
if (VMin>0)             % fill
   for i=1:(vHDim-VMin)
      I(:,WidthOri+i)=I(:,WidthOri);
   end
end
HMin=mod(HeightOri,vVDim);
if (HMin>0)
   for i=1:(vVDim-HMin)
      I(HeightOri+i,:)=I(HeightOri,:);
   end
end

[Height,Width]=size(I);

nHBlock = Width/vHDim;    % num horz block/vector
nVBlock = Height/vVDim;   % num vert block/vector

if (imgIter==1) iData = 0; end;
switch iterRow
   case 1
      iRowBegin=1;
      iRowEnd=Height/4;
   case 2
      iRowBegin=Height/4;
      iRowEnd=Height/2;
```

```
        case 3
            iRowBegin=Height/2;
            iRowEnd=3*Height/4;
        case 4
            iRowBegin=3*Height/4;
            iRowEnd=Height-vVDim;
    end
    switch iterCol
        case 1
            iColBegin=1;
            iColEnd=Width/4;
        case 2
            iColBegin=Width/4;
            iColEnd=Width/2;
        case 3
            iColBegin=Width/2;
            iColEnd=3*Width/4;
        case 4
            iColBegin=3*Width/4;
            iColEnd=Width-vHDim;
    end

    for iRow=iRowBegin:vVDim:iRowEnd        % 1:vVDim:(Height/4) ;
(Height/4)+1:vVDim:(Height/2) ; (Height/2)+1:vVDim:(3*Height/4) ;
(3*Height/4):vVDim:Height-vVDim
        for iCol=iColBegin:vHDim:iColEnd    % 1:vHDim:(Width/4) ;
(Width/4)+1:vHDim:(Width/2) ; (Width/2)+1:vHDim:(3*Width/4) ;
(3*Width/4):vHDim:Width-vHDim
            % get pixel from image accordingly
            BRow=iRow;
            BCol=iCol;
            iData = iData + 1;
```

```
                for pixHorz=1:vVDim
                    startBRow = 1 + (pixHorz-1)*vHDim;   % correction vHDim
                    for pixVert=1:vHDim
                        posH = BRow + pixHorz - 1;
                        posV = BCol + pixVert - 1;
                        counter = startBRow + pixVert - 1;
                        bRow(1,counter)=I(posH,posV);
                    end
                end
                MI(iData,:)=bRow;
            end
        end
    end

    fprintf('Clustering using FCM ...\n');
%       display('Clustering ...');
    MI=double(MI);
    [center,U,obj_fcn] = fcm(MI, 256);

    display('end');
    time=toc

    filename=['trainFCM',num2str(vHDim),num2str(vVDim),'-
',num2str(iterRow),num2str(iterCol)];
    save(filename);
  end
end
% combine centers all together
close all;
clear all;
load trainFCM44-11;
save('center11','center');
```

```
trainVec=center;
load trainFCM44-12;
save('center12','center');
trainVec=[trainVec;center];
load trainFCM44-13;
save('center13','center');
trainVec=[trainVec;center];
load trainFCM44-14;
save('center14','center');
trainVec=[trainVec;center];
load trainFCM44-21;
save('center21','center');
trainVec=[trainVec;center];
load trainFCM44-22;
save('center22','center');
trainVec=[trainVec;center];
load trainFCM44-23;
save('center23','center');
trainVec=[trainVec;center];
load trainFCM44-24;
save('center24','center');
trainVec=[trainVec;center];
load trainFCM44-31;
save('center31','center');
trainVec=[trainVec;center];
load trainFCM44-32;
save('center32','center');
trainVec=[trainVec;center];
load trainFCM44-33;
save('center33','center');
trainVec=[trainVec;center];
load trainFCM44-34;
```

```
save('center34','center');
trainVec=[trainVec;center];
load trainFCM44-41;
save('center41','center');
trainVec=[trainVec;center];
load trainFCM44-42;
save('center42','center');
trainVec=[trainVec;center];
load trainFCM44-43;
save('center43','center');
trainVec=[trainVec;center];
load trainFCM44-44;
save('center44','center');
trainVec=[trainVec;center];

save('trainVec44','trainVec');

% last FCM to get last codebook
close all;
clear all;
load trainVec44;
tic;
display('Last FCM Clustering ...');
[center,U,obj_fcn] = fcm(double(trainVec), 256);
time=toc
display('End last training ..... ');
save lastTrain44;
center=round(center);
save('codebook44','center');
```

# LAMPIRAN B

**Lampiran B : Listing Program Image Encoder**

```
clear all;
close all;
clc;

tic;

load codebook44;        % load codebook - sesuaikan dengan [nama codebook].mat

strFile = 'XrayH1.bmp'; % isikan dengan nama file citra medik
I=imread(strFile);

% Original image size
[VOri,HOri]=size(I);
vHDim = 4;              % horz dimension
vVDim = 4;              % vert dimension

% Adjust image size according to vector dimesion
VMin=mod(HOri,vHDim);
if (VMin>0)
   for i=1:(vHDim-VMin)
      I(:,HOri+i)=I(:,HOri);
   end
end
HMin=mod(VOri,vVDim);
if (HMin>0)
   for i=1:(vVDim-HMin)
      I(VOri+i,:)=I(VOri,:);
   end
end

% adjusted image size
```

```
[V,H]=size(I);
nHBlock = H/vHDim;   % num horz block/vector
nVBlock = V/vVDim;   % num vert block/vector

dataNum = nHBlock*nVBlock;
dimension = vHDim*vVDim;

iData = 0;
% for iHorz=1:vVDim:V
idxRow = 1;
for iRow=1:vVDim:V
   idxCol = 1;
   for iCol=1:vHDim:H
      % get pixel from image accordingly
      BRow=iRow;
      BCol=iCol;
      iData = iData + 1;
      for pixHorz=1:vVDim
         startBRow = 1 + (pixHorz-1)*vVDim;
         for pixVert=1:vHDim
            posH = BRow + pixHorz - 1;
            posV = BCol + pixVert - 1;
            counter = startBRow + pixVert - 1;
            bRow(1,counter)=I(posH,posV);
         end
      end
%        display(['calculating index row= ',num2str(idxRow),' col=
',num2str(idxCol)]);
      EucDist=distfcm(center,double(bRow(1,:)));
      [minVal,idx]=min(EucDist);
      EncodedImg(idxRow,idxCol)=idx;
      idxCol = idxCol + 1;
```

```
    end
    idxRow = idxRow + 1;
end
timeEnc=toc

save('XrayH1-idx', 'EncodedImg', 'HOri', 'VOri', 'timeEnc')
```

# LAMPIRAN C

**Lampiran C : Listing Program VQ Decoder**

```
close all;
clear all;
clc;
tic;

load codebook44;

load XrayH1-idx;
vHDim = 4;              % horz dimension
vVDim = 4;              % vert dimension

% ---- begin decoding

IDX=EncodedImg;
[height,width]=size(EncodedImg);
clear EncodedImg;
I=zeros(height*4,width*4);
nWB = width;
nHB = height;

for i=1:nHB
   for j=1:nWB
      curIdx=IDX(i,j);
      decStream=center(curIdx,:);
      streamIdx=1;
      for p=1:vVDim
         for q=1:vHDim
            decBlock(p,q)=decStream(streamIdx);
            I(((i-1)*vHDim)+p,((j-1)*vVDim)+q)=decStream(streamIdx);
            streamIdx=streamIdx+1;
         end
```

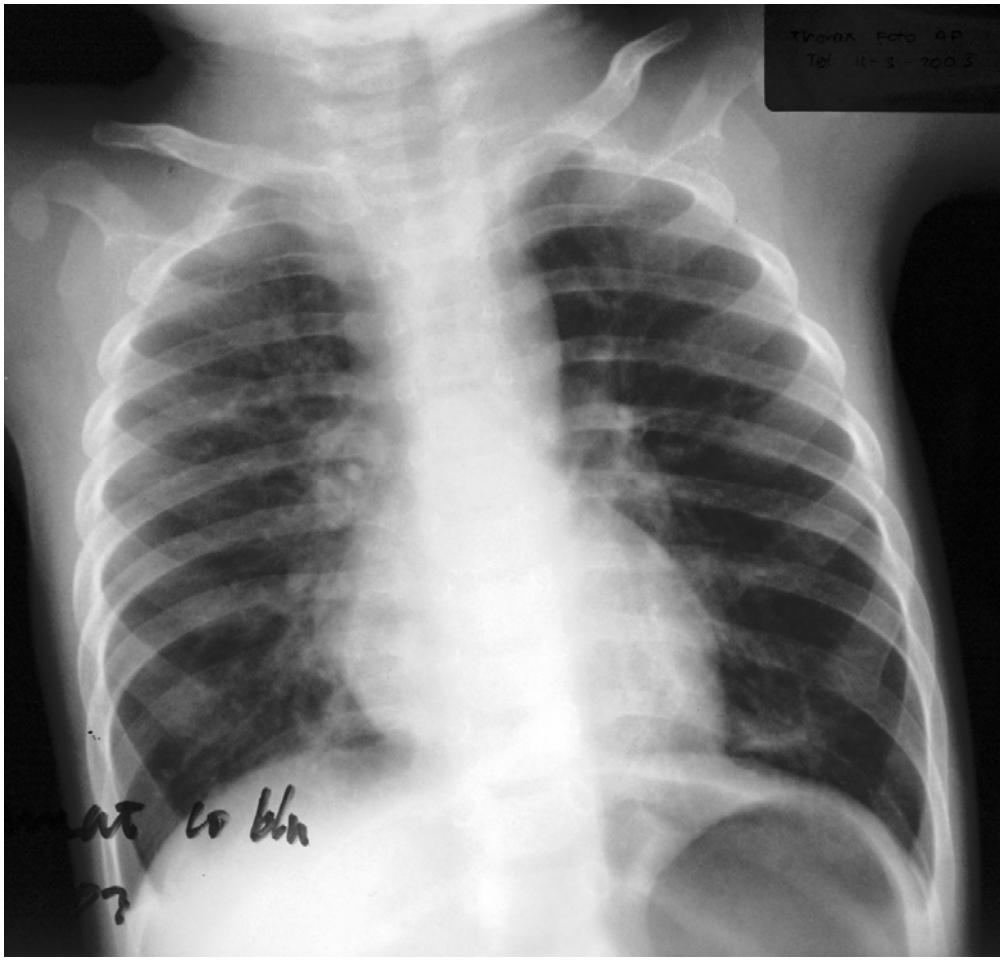
```
      end
   end
end

time=toc

% ---- finish decoding and calculating residue

IOri=imread('XRayH1.bmp');      % original image in integer format
IOriD=double(IOri);             % original image in double format
IRec=I(1:VOri,1:HOri);          % recontructed image
residue=IOriD-IRec;             % image residue
residueIn=uint8(residue+128);   % image residu + 128 in integer format

IOriNorm = IOriD./255;          % normalized original image
IRecNorm = IRec./255;           % normalized reconstructed image

imwrite(uint8(IRec), 'XRayH1Rec.bmp', 'bmp');         % save recontructed image
                    in BMP format

fid=fopen('resXRayH1.bin','wb');                      % save image residue in binary
                    format
fwrite(fid, residueIn, 'integer*1');
fclose(fid);

figure(1), imagesc(IOri),  axis image, colormap(gray); colorbar; title('Original
                    Image');
figure(2), imagesc(uint8(IRec)),      axis image, colormap(gray); colorbar;
                    title('Reconstructed Image');
figure(3), imagesc(residue), axis image, colormap(gray); colorbar; title('Residu
                    Image');
```

```
% ---- PSNR calculation
A = IOriNorm;
B = IRecNorm;
if A == B
  error('Images are identical: PSNR has infinite value')
end

max2_A = max(max(A));
max2_B = max(max(B));
min2_A = min(min(A));
min2_B = min(min(B));

if max2_A > 1 | max2_B > 1 | min2_A < 0 | min2_B < 0
  error('input matrices must have values in the interval [0,1]')
end

error = A - B;
decibels = 20*log10(1/(sqrt(mean(mean(error.^2)))));
disp(sprintf('PSNR before selection = +%5.2f dB',decibels))

figure(2);
[x,y,pos,rect]=imcrop;
rect1=double(round(rect));
xmin=rect1(1,1);
ymin=rect1(1,2);
width=rect1(1,3);
height=rect1(1,4);
xmax = xmin + width;
ymax = ymin + height;
% %
figure(4), imagesc(IRec(ymin:ymax,xmin:xmax)), colormap(gray), axis image;
```

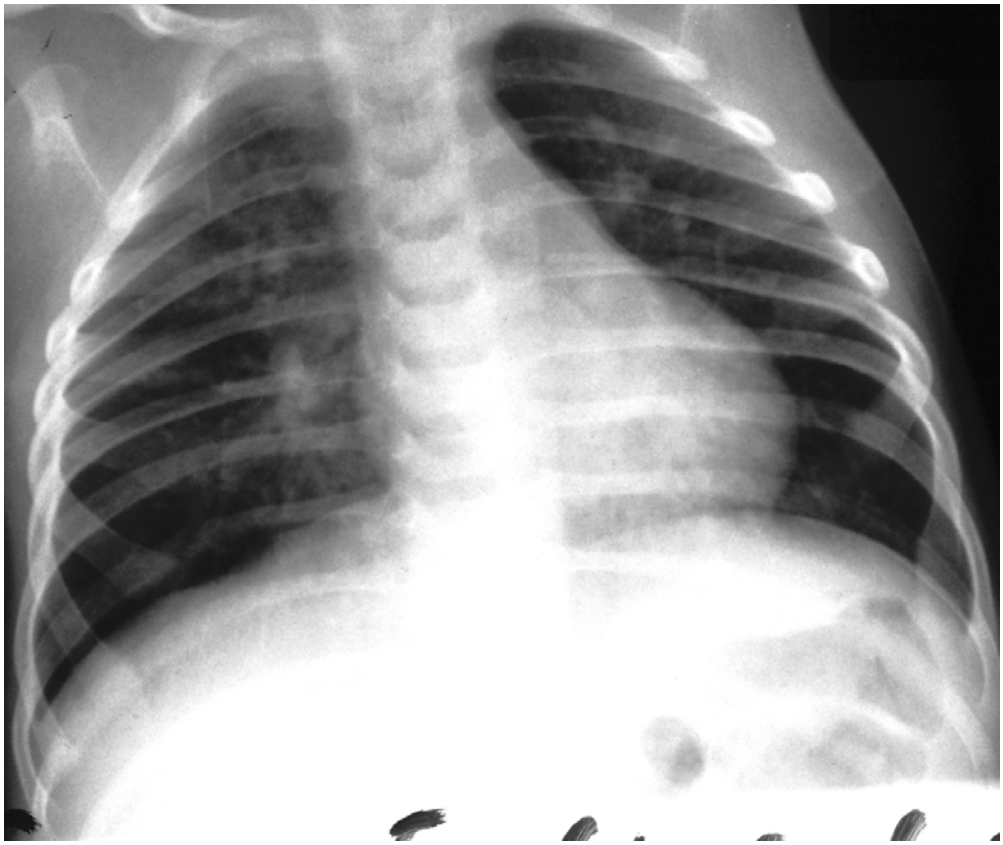
```
IRec(ymin:ymax,xmin:xmax) = IRec(ymin:ymax,xmin:xmax) +
                residue(ymin:ymax,xmin:xmax);
figure(5), imagesc(IRec), colormap(gray), axis image;

A = IOriNorm;      % normalize
B = IRec./255;     % normalize
error = A - B;
decibels = 20*log10(1/(sqrt(mean(mean(error.^2)))));
disp(sprintf('PSNR after selection = +%5.2f dB',decibels))
```

# LAMPIRAN D

**LAMPIRAN D : Citra medis latih**



ThoraxA.bmp

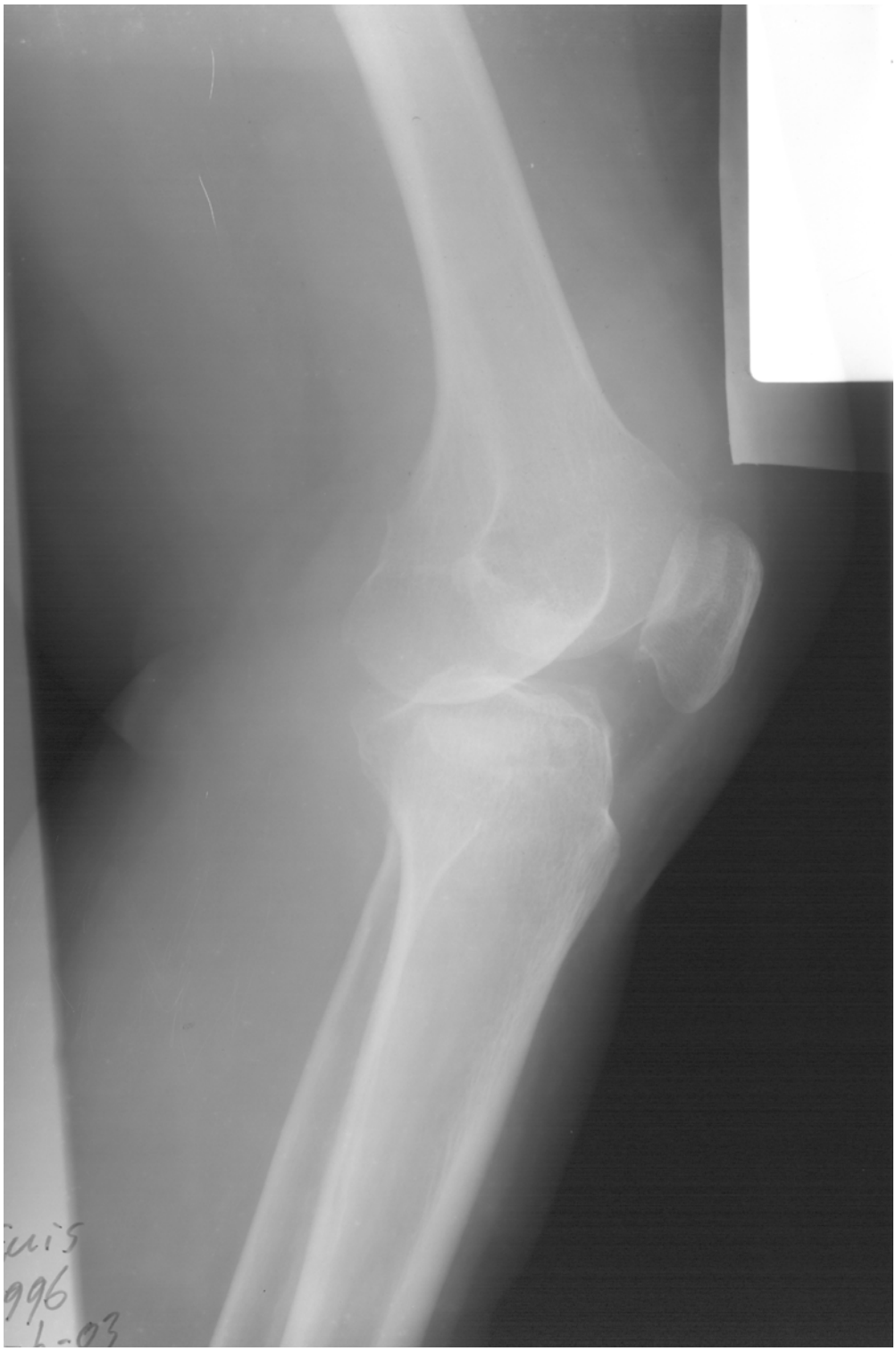


ThoraxB.bmp


Nama : Ferdians
Tanggal : 17 Febru
Med Rek : 0303031
DK/ : TB paru
Rö : thorax AP

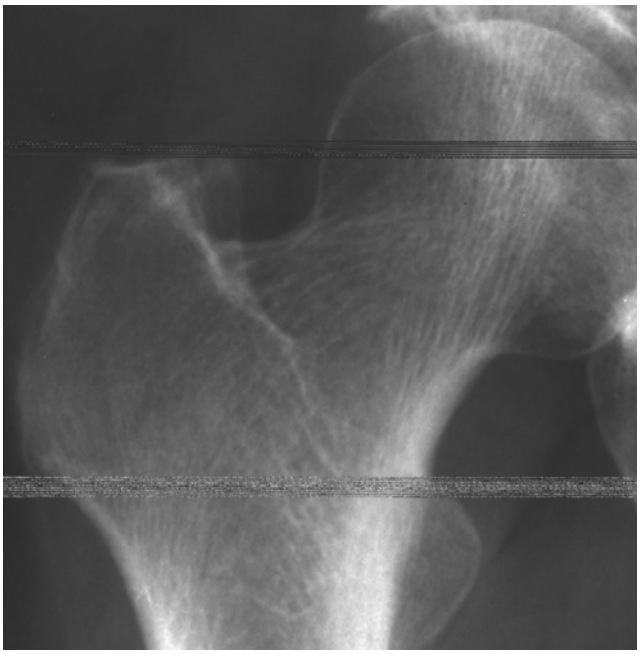
ThoraxD.bmp

XrayF1.bmp

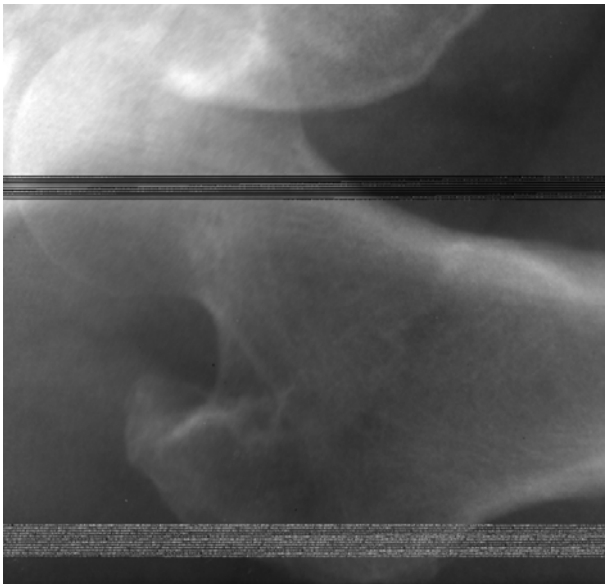


XrayF2.bmp

XrayFE1.bmp



XrayFE2.bmp



XrayFE3.bmp

XrayH1.bmp


XrayH2.bmp