Comparison of Principal Component Analysis (PCA), Residual Vector Quantization (RVQ), Exhaustive Search Vector Quantization (ESVQ), and Tree Structured Vector Quantization (TSVQ)

Salman Aslam
Georgia Tech
Technical Report GT-CoveringSets-1

July 14, 2011

Contents

Abstract

a

1 Introduction

2 Experiments

3 Results

4 Conclusions

A Notation

- N. Number of data points.
- D. Dimensionality of data points.
- **Design matrix**. A matrix containing input data. The terminology is used by Andrew Ng from Stanford in his Machine Learning course. In the design matrix (DM), each row contains one data observation, i.e., there is one data vector per row. This is a NxD matrix. For one data vector per column, I use DM2 which is a DxN matrix and is simply the transpose of DM.
- **cfn**. This stands for "complete filename."
- **posneg image**. A planar image with the following planes: R, G, B, 255-R, 255-G, 255-B. A plane consists of all pixels belonging to a particular channel. In other words, a posneg image contains all red pixels, followed by all green pixels, followed by all blue pixels, followed by the inverted channels. To view such an image, open it in the Irfanview free software. Make sure the image has a .raw extension.

B Source code

B.1 Stand-alone applications for sanity checking.

```
%% This file compares PCA, RVQ and TSVQ by generating its own simple data.
%
% It can be used without any outside data, except if useDataSet=2 is used.
%
% usage example
% -------------
% A good way to check RVQ is to use the following settings:
% useDataSet          =   4   %this is 256 numbers going from 1:1:256 as mentioned in the IDDM paper.
% rvq_struct.maxP     =   8
% rvq_struct.M        =   2
% Then rvq_struct_PHI_r (the red channel of the codebooks) is
% 192.4961   64.4961  -32.0000   32.0000  -16.0000 16.0000   -8.0000 8.0000   -4.0000    4.0000   -2.0000    2.0000   -1.0000 ↘
→   1.0000   -0.4961    0.5039
% The green and the blue channels are also the same.
% This behavior is correct, as confirmed by Dr Barnes.
% The first M=2 numbers are the scalar codevectors for stage 1,
% the second M=2 numbers are the scalar codevectors for stage 2, and so on.

% Copyright (C) Salman Aslam.  All rights reserved.
% Data created       : April 20, 2011
% Date last modified : July 7, 2011.
%%

%-------------------------------
%INITIALIZATIONS
%-------------------------------

%matlab
    clear;
    clc;
    close all;
    format compact;

%data input (5 different datasets, pick one of them)
    useDataSet                  =   4;                                  %change this to 1, 2, 3, 4 or 5
    if      (useDataSet==1)    ...
                [DM2, sw, sh]   =   DATAMATRIX_create_simple_DM2;
    elseif  (useDataSet==2)             load testS_DM2_small;           %don't use if you don't have this file
    elseif  (useDataSet==3)   DM2   =   [4 6 8 10 20 22 24 26];sw=1;sh=1;   %sw=1, sh=1 means scalar example
    elseif  (useDataSet==4)   DM2   =   1:256;  sw=1;   sh=1;           %scalar example mentioned in IDDM
    elseif  (useDataSet==5)   a     =   rand(1089,2); DM2 = [a a a a];sw=33;sh=33;  %this is bizarre,
                                                                        %i.e. has repeated data points
    end

    [D, N]                      =   size(DM2);                          %dimensions of DM2

%algorithm parameters
    %bpca
    bpca_struct.tstprm_Neig_1x1     =   16;
    bpca_struct.trgout_descriptors  =   [];
    bpca_struct.tstout_descriptor   =   [];

    %tsvq
    tsvq_struct.P                   =   5;                              %number of stages
    tsvq_struct.M                   =   2;                              %2 is for binary TSVQ
    tsvq_struct.trgout_descriptors  =   [];
    tsvq_struct.tstout_descriptor   =   [];

    %rvq
    rvq__struct.maxP                =   8;                              %number of stages
    rvq__struct.M                   =   2;                              %number of codevectors/stage
    rvq__struct.targetSNR           =   1000;
    rvq__struct.sw                  =   sw;                             %snippet width
    rvq__struct.sh                  =   sh;                             %snippet height
    rvq__struct.dir_out             =   '';
    rvq__struct.trgout_descriptors  =   [];
    rvq__struct.tstout_descriptor   =   [];



%-------------------------------
% PROCESSING
%-------------------------------
%training
    bpca_struct                 =   bPCA_1_train        (DM2, bpca_struct);
    rvq__struct                 =   RVQ__training       (DM2, rvq__struct);
    tsvq_struct                 =   TSVQ_1_train        (DM2, tsvq_struct);

%testing
    tst_idx                     =   8;                                  %any number between 1 and N, index
                                                                        %of training data that you want to test
    tst_Dx1                     =   DM2(:,tst_idx);                     %test vector
    bpca_struct                 =   bPCA_3_test          (tst_Dx1, bpca_struct);
    rvq__struct                 =   RVQ__testing         (tst_Dx1, rvq__struct);
    tsvq_struct                 =   TSVQ_3_test          (tst_Dx1, tsvq_struct);

%-------------------------------
%RESULTS
%-------------------------------
%view
    numDisplayRows              =   5;
    numDisplayCols              =   10;
                                    DATAMATRIX_display_DM2_as_image(DM2,                        sh, sw, numDisplayRows, ↘
                                    →numDisplayCols);
                                    DATAMATRIX_display_DM2_as_image(bpca_struct.trgout_U_DxD, sh, sw, numDisplayRows, ↘
                                    →numDisplayCols);

%print
        %[DM2(:,tst_idx)       bpca_struct.tstout_recon_Dx1     rvq__struct.tstout_recon_Dx1      tsvq_struct.tstout_recon_Dx1]

    [bpca_struct.tstout_snr_1x1    rvq__struct.tstout_snr_1x1       tsvq_struct.tstout_snr_1x1       ]
    [bpca_struct.tstout_rmse_1x1   rvq__struct.tstout_rmse_1x1      tsvq_struct.tstout_rmse_1x1      ]
```

```
rvq__struct.PHI_r;
```

Listing 1: main_compare_bPCA_ESVQ_RVQ_TSVQ.m.

B.2 Training data manipulation

```
%% This file generates a DxN matrix of test data.

% I call this a design matrix DM2 using Andrew Ng's words.
% A DM is NxD, DM2 is DxN.
% In DM, every D-dimensional vector is in one column.
% In DM2, every D-dimensional vector is in one row.
%
% Copyright (C) Salman Aslam.  All rights reserved.
% Data created       : April 18, 2011
% Date last modified : July 7, 2011
%%

function [DM2, sw, sh] = DATAMATRIX_create_simple_DM2()

    D                           =   11;
    X1                          =   [1 5 3 8 15 16 17 28 29 40 45]; %D dimensional data
    X2                          =   X1+0.5*randn(1,D);
    X3                          =   X1+0.7*randn(1,D);
    X4                          =   X1+0.3*randn(1,D);
    DM                          =   [X1; X2; X3; X4];       %create a total of 4 D-dimensional
                                                            %data points clustered around X1


    for i=1:5
        X1                      =   X1+50*i;
        X2                      =   X1+0.1*randn(1,D);       %row vector
        X3                      =   X1+0.2*randn(1,D);
        X4                      =   X1+0.3*randn(1,D);
        DM                      =   [DM; X1; X2; X3; X4];   %4 data points centered around X1
                                                            %4 data points centered around X1+50,
                                                            %4 data points centered around X1+150,
                                                            %4 data points centered around X1+300,
                                                            %4 data points centered around X1+500,
                                                            %4 data points centered around X1+750
                                                            %24 total points
    end

    DM2                         =   DM';
    sw                      =   1;                          %we're saying the snippet width is 1
    sh                          =   D;                          %snippet height is D, so this is a 1D snippet
                                                            %rather than say an image
```

Listing 2: DATAMATRIX_create_simple_DM2.m.

```
%% This function writes input data (training vectors) to a file to be used for training an RVQ codebook.
%
% For all my algorithms, PCA, ESVQ, RVQ, TSVQ, the input data is formatted as a matrix called DM2.
% Refer to readme.pdf for a description of DM2.
%
% The training routines for PCA, ESVQ and TSVQ take the DM2 as input.
% However, RVQ requires its input to be saved as a file before it can be used for training by RVQ_1a_train_gen8.exe or ↘
→RVQ_1a_train_gen16.exe.
% This function takes a DM2 and creates a file in the following manner: (a) 512 byte header, followed by (b) each posneg ↘
→training vector.
%
% The file created by this function is what used to be called F1.sml in the original setup.
%
% Copyright (C) Salman Aslam.  All rights reserved.
% Date created       : March 23, 2011
% Date last modified : July 9, 2011.
%%

function DATAMATRIX_saveInFormat_rvq(DM2, cfn_trainingFile, sw, sh)  %cfn_trainingFile is the complete filename for the ouput ↘
→file
                                                                %this file will contain a header and data in the DM2 ↘
                                                                →matrix
                                                                %sw is snippet width, sh is snippet height

%--------------------
%INITIALIZATIONS
%--------------------
    Nc                      =   6;                          %number of channels in a training snippet, i.e., training vector
    [D, N]                  =   size(DM2);                  %N is the number of training vectors,
    shc                     =   sh*Nc;                      %number of rows in a single training snippet (vector)
    headerSize              =   512;                        %number of bytes in header
    numBytes                =   headerSize + sw*shc*N;      %total bytes in entire file

%--------------------
%PRE-PROCESSING
%--------------------
% open the file
    fid                     =   fopen(cfn_trainingFile, 'w');   %if this file exists, it is overwritten, fid is for file ID
                                UTIL_FILE_checkFileOpen(fid, cfn_trainingFile);

%initialize header with all zeros
    for i=1:headerSize
        fwrite(fid, 0, 'char');
    end

%--------------------
%PROCESSING
%--------------------
%write header
    TSH                     =   1;
    UCH                     =   1;
    REAL                    =   0;

    fseek(fid, 0,  'bof');      fwrite(fid, TSH,        'int32');
    fseek(fid, 4,  'bof');      fwrite(fid, 'abcdefg',  'char' );
    fseek(fid, 36, 'bof');      fwrite(fid, num2str(N), 'char' );
    fseek(fid, 44, 'bof');      fwrite(fid, sw,         'int32');
```

```
    fseek(fid, 48, 'bof');       fwrite(fid, shc,         'int32');
    fseek(fid, 52, 'bof');       fwrite(fid, UCH,         'int32');
    fseek(fid, 56, 'bof');       fwrite(fid, REAL,        'int32');
    fseek(fid, 60, 'bof');

    for i=1:240+212
                                 fwrite(fid, 204,         'uint8');
    end

%write data
    fseek(fid, 512, 'bof');
    for i=1:N
        I                =   DATAMATRIX_extract_ith_image_from_DM2(DM2, i, sw, sh);%this returns an image
        Iposneg          =   RVQ__create_posnegImage(I, '', false, false);     %this converts the image to posneg format
                             fwrite(fid, Iposneg);                               %write each posneg image one after the ↘
                             →other
    end

%--------------------
%POST-PROCESSING
%--------------------
%close file
    fclose(fid);
```

Listing 3: "DATAMATRIX_saveInFormat_rvq.m."

```
%% This function assumes that each column of the input matrix (DM2) is an
%% image.  It extracts a given column and returns it as an image.
%
% Copyright (C) Salman Aslam.  All rights reserved.
% Date created          : April 17, 2011
% Date last modified    : July 9, 2011.
%%

function I_h_x_w = DATAMATRIX_extract_ith_image_from_DM2(DM2, col, w, h)

    I_Dx1                =   DM2(:,col);              %extract D-dimensional column
    I_h_x_w              =   reshape(I_Dx1, h, w);    %reshape as an image and return
```

Listing 4: DATAMATRIX_extract_ith_image_from_DM2.m.

```
%% Takes a DM2 matrix and displays it as an image.
%
% Copyright (C) Salman Aslam.  All rights reserved.
% Date created          : April 17, 2011
% Date last modified    : July 9, 2011.
%%

function DATAMATRIX_display_DM2_as_image(DM2, h, w, numRows, numCols)

    [D, N]               =   size(DM2); %N: number of training observations, D: dimensionality of data

    for n = 1:N
        col_vec          =   DM2(:,n);
        img              =   reshape(col_vec, h, w);
                                                  UTIL_PLOT_tightsubplot(numRows, numCols, n, img);
    end
```

Listing 5: DATAMATRIX_display_DM2_as_image.m.

```
%% Takes a DM2 matrix and displays it as an image.
%
% Assumption is that the data is grayscale.
%
% Here, the design matrix has observations in each row, that's standard
% however, each observation is an image which was vectorized by stacking
% columns onto each other, not row onto each other
%
% Copyright (C) Salman Aslam.  All rights reserved.
% Date created          : April 5, 2011
% Date last modified    : July 9, 2011.
%%

function DATAMATRIX_display_DM_as_image_col(DM, h, w, numCols) %h and w are snippet height and width

    [N, D]               =   size(DM); %N: number of training observations, D: dimensionality of data

    for n = 1:N
        row_vec          =   DM(n,:);
        img              =   reshape(row_vec, h, w);
                             UTIL_PLOT_tightsubplot(numCols, n, img)
    end
```

Listing 6: DATAMATRIX_display_DM_as_image_col.m.

B.3 Training

```
%% This function creates RVQ codevectors.
%
% This is the only function out of all my work that is closed source and does not belong to me.
% This is not to say that I have not incorporated other open source stuff in my work (always with appropriate permissions).
%
% Codebooks are denoted by PHI.
%
% Copyright (C) Salman Aslam.  All rights reserved.
% Data created       : April 17, 2011
% Date last modified : July 7, 2011
%%

function rvq_struct = RVQ_training(DM2, rvq_struct)

%----------------
%INITIALIZATIONS
%----------------
    DM2_u8                  =   uint8(DM2);             %design matrix, one observation per column
    M                       =   rvq_struct.M;           %number of templates per stage
    maxP                    =   rvq_struct.maxP;        %max number of stages
    sw                      =   rvq_struct.sw;          %snippet width
    sh                      =   rvq_struct.sh;          %snippet height
    targetSNR               =   rvq_struct.targetSNR;   %desired SNR
    dir_out                 =   rvq_struct.dir_out;     %directory to store results in

%!! attention: these should be parameters but I'm fixing them !!
    iFlag                   =   0.0005;
    jFlag                   =   0.0005;

%filenames (original names in brackets in comments)
    cfn_trainingFile        =   [dir_out 'positiveExamples.raw']; %file 1: vectorized positive examples, (F1.sml)
    cfn_ecbk                =   [dir_out 'codebook.ecbk'];        %file 2: encoder codebooks,            (F1.ecbk)
    cfn_dcbk                =   [dir_out 'codebook.dcbk'];        %file 3, decoder codebooks,            (F1.dcbk)
    cfn_nodes               =   [dir_out 'codebook.nodes'];       %file 4, linked list of training paths,(F1.nodes)
    cfn_gentxt              =   [dir_out 'rvq_trg_verbose.txt']; %file 5, verbose output of gen.exe,    (F1.stat_gen.txt)
    cfn_bndintxt            =   [dir_out 'bnd_in.txt'];          %file 6, verbose output of bnd_in.exe  (F1.stat_bnd_in.txt)
    cfn_trgsoc              =   [dir_out 'positiveExamples.idx']; %file 7, XDRs for training examples,   (F1.idx)

%delete existing training files
                            UTIL_FILE_deleteFile(cfn_trainingFile);%file 1
                            UTIL_FILE_deleteFile(cfn_ecbk);        %file 2
                            UTIL_FILE_deleteFile(cfn_dcbk);        %file 3
                            UTIL_FILE_deleteFile(cfn_nodes);       %file 4
                            UTIL_FILE_deleteFile(cfn_gentxt);      %file 5
                            UTIL_FILE_deleteFile(cfn_bndintxt);    %file 6
                            UTIL_FILE_deleteFile(cfn_trgsoc);      %file 7
%--------------------
%PRE-PROCESSING
%--------------------
    DATAMATRIX_saveInFormat_rvq     (DM2_u8, cfn_trainingFile, sw, sh); %takes DM2 as input and writes it to a file


%--------------------
%PROCESSING
%--------------------

    if (ispc)

        if (maxP==8)
            system(['RVQ_training_gen8.exe____' cfn_trainingFile  '_' cfn_ecbk '_'  cfn_dcbk '_' num2str(M+1) '_-S' num2str(↘
            →targetSNR) '_-i' num2str(iFlag) '_-j' num2str(jFlag) '_>_' cfn_gentxt]);
        elseif (maxP==16)
            system(['RVQ_training_gen16.exe___' cfn_trainingFile  '_' cfn_ecbk '_'  cfn_dcbk '_' num2str(M+1) '_-S' num2str(↘
            →targetSNR) '_-i' num2str(iFlag) '_-j' num2str(jFlag) '_>_' cfn_gentxt]);
        end

    elseif (isunix)

        if (maxP==8)
            system(['./RVQ_training_gen8.linux_' cfn_trainingFile  '_' cfn_ecbk '_'  cfn_dcbk '_' num2str(M+1) '_-S' num2str(↘
            →targetSNR) '_-i' num2str(iFlag) '_-j' num2str(jFlag) '_>_' cfn_gentxt]);
        elseif (maxP == 16)
            system(['./RVQ_training_gen16.linux' cfn_trainingFile  '_' cfn_ecbk '_'  cfn_dcbk '_' num2str(M+1) '_-S' num2str(↘
            →targetSNR) '_-i' num2str(iFlag) '_-j' num2str(jFlag) '_>_' cfn_gentxt]);
        end
    end



%--------------------
%POST-PROCESSING
%--------------------
%read decoder codebook
    [actualP, M_check, sw_check, sh_check, PHI_r, PHI_g, PHI_b, PHIn_r, PHIn_g, PHIn_b]  =  RVQ_read_codebook        (↘
    →cfn_dcbk);

%error checking
    if (M ~= M_check || sw ~= sw_check || sh ~= sh_check)
        disp('ERROR:_M,_sw,_or_sh_not_correct')
    end

%save to structure that will be passed back from function
    rvq_struct.P            =   actualP;
    rvq_struct.PHI_r        =   PHI_r;
    rvq_struct.PHI_g        =   PHI_g;
    rvq_struct.PHI_b        =   PHI_b;
    rvq_struct.PHIn_r       =   PHIn_r;
    rvq_struct.PHIn_g       =   PHIn_g;
    rvq_struct.PHIn_b       =   PHIn_b;

%compute training SNR
    %compute training SNR and training XDR descriptors using myExplorer
    rvq_struct              =   UTIL_METRICS_compute_training_error(DM2, rvq_struct, 3);
```

```
100     %compute training XDR descriptors using gen -l
101     %rvq__struct.trgout_descriptors_genl=   RVQ_3_check_soc_file(cfn_trgsoc, actualP, M, true);

103     %compare the 2 above
104     %diff                           =   RVQ_compare_myExplorer_w_genl((rvq__struct.trgout_descriptors_genl)', rvq__struct.↘
        →trgout_descriptors);

106 %old code no longer used but here just in case
107     %system(['RVQ_0_create_posraw.exe           '     cfn_poscsv   ' ' num2str(sw)  ' '  num2str(sh)  ' ' cfn_trainingFile]);
108     %system(['./RVQ_0_create_posraw.linux       '     cfn_poscsv   ' ' num2str(sw)  ' '  num2str(sh)  ' ' cfn_trainingFile]);
109     %system(['RVQ_1b_train_bndin8.exe           '     cfn_trainingFile   ' ' cfn_ecbk     ' '  cfn_dcbk     ' ' num2str(M+1)  ↘
        →' '   '-S' num2str(targetSNR)    ' -i' num2str(iFlag)    ' -j' num2str(jFlag)    ' > ' cfn_bndintxt]);
110     %system(['RVQ_1a_train_gen8.exe             '     cfn_trainingFile   ' ' cfn_ecbk     ' '  cfn_dcbk     ' ' num2str(M+1)  ↘
        →' '   '-S' num2str(targetSNR)    ' -l'                                   ' > ' 'temp.txt']);
111     %system(['./RVQ_1b_train_bndin8.linux       '     cfn_trainingFile   ' ' cfn_ecbk     ' '  cfn_dcbk     ' ' num2str(M+1)  ↘
        →' '   '-S' num2str(targetSNR)    ' -i' num2str(iFlag)    ' -j' num2str(jFlag)    ' > ' cfn_bndintxt]);
112     %system(['./RVQ_1a_train_gen8.linux         '     cfn_trainingFile   ' ' cfn_ecbk     ' '  cfn_dcbk     ' ' num2str(M+1)  ↘
        →' '   '-S' num2str(targetSNR)    ' -l'                                   ' > ' 'temp.txt']);
```

Listing 7: "RVQ_training.m"

B.4 Testing

```
% Tests a test vector using an RVQ codebook.
%
% Copyright (C) Salman Aslam.  All rights reserved.
% Date created       : April 17, 2011.
% Date last modified : July 7, 2011.
%%
%Description:

%%
function rvq_struct = RVQ_testing(tst_Dx1, rvq_struct)

    PHI_r                               =   rvq_struct.PHI_r;  %codebook, just the red channel
    P                                   =   rvq_struct.P;      %actual number of stages
    M                                   =   rvq_struct.M;      %number of codevectors/stage
    sw                                  =   rvq_struct.sw;     %snippet width
    sh                                  =   rvq_struct.sh;     %snippet height
    D                                   =   sw*sh;             %dimension of data

    XDR                                 =   P + ones(P,1);%i initialize with P+1, the code for early termination
    recon_Dx1                           =   zeros(D,1);

    psnr_dB                             =   0;
    PSNR_dB                             =   [];
    successiveRecon_DxP                 =   zeros(1*D,P); %1st col contains 1st recon., 2nd column contains refined recon.,

    err_Dx1                             =   tst_Dx1;
    numStagesUsed                       =   0;

        %go over all stages
        for p=1:P
            dmin                        =   1E15;

            %for this stage: go over all codevectors
            for m=1:M
                idx                     =   UTIL_xy_to_idx(m, p, M);
                                PHI_Dx1                             =       PHI_r(:,idx); %!!!caution!!! notice that only R ↘
                                →channel used
                e                       =   err_Dx1-PHI_Dx1;
                d                       =   norm(  e, 2  ); %if e is a matrix, this is largest eigenvalue, if it'm a vector, ↘
                →it'm L2 norm
                if (d<dmin)
                    dmin = d;
                    best_m = m;
                end
            end

            %for this stage: temporarily store metrics
            best_idx                                    =   UTIL_xy_to_idx(best_m, p, M);
                        best_PHI                                        =       PHI_r(:,best_idx);
            temp_recon_Dx1                              =       recon_Dx1 + best_PHI;
            temp_err_Dx1                                =   tst_Dx1 - temp_recon_Dx1;
            temp_psnr_dB                                =   UTIL_METRICS_compute_PSNRdB (255,           temp_err_Dx1);
            %temp_snr_dB                                =   UTIL_METRICS_compute_SNRdB  (tst_Dx1,       temp_err_Dx1);

            %for this stage: decide if it will be "retained"
            if (temp_psnr_dB < psnr_dB)
                %discard
                break
            else
                %keep
                                recon_Dx1                                       =       temp_recon_Dx1;
                successiveRecon_DxP(:,p)=  temp_recon_Dx1;
                err_Dx1                         =       temp_err_Dx1;
                                psnr_dB                                         =   temp_psnr_dB;
                PSNR_dB(p)                  =   temp_psnr_dB;
                XDR(p)                      =   best_m;
                numStagesUsed               =   p;
            end
        end
        %end: going over stages




    %pass out
        rvq_struct.tstout_recon_Dx1     =   recon_Dx1;
        rvq_struct.tstout_err_Dx1       =   err_Dx1;
        rvq_struct.tstout_descriptor    =   XDR;

        rvq_struct.tstout_Nstages_used  =   numStagesUsed;

        rvq_struct.tstout_snr_1x1       =   UTIL_METRICS_compute_SNRdB      (tst_Dx1,  err_Dx1);  %for PSNR, you only give ↘
        →error signal
        rvq_struct.tstout_rmse_1x1      =   UTIL_METRICS_compute_rms_value  (           err_Dx1);
        rvq_struct.tstout_psnr_1x1      =   max(PSNR_dB);
```

Listing 8: "RVQ_testing.m"

B.5 Tracking algorithms

```
%% This function implements the particle filter (condensation algorithm).
%
% 0to1 means that the min value is 0, max value is 1.
%
% I_0to1 is the input image.
% f is the frame number.
%
% Copyright (C) Jongwoo Lim and David Ross (modified by Salman Aslam with permission)
% Date created      : April 25, 2011
% Date last modified: July 14, 2011
%%

function cond_struct = TRK_condensation(I_0to1, f, algo_struct, cond_struct, options_struct, RandomData_sample, RandomData_cdf, algo_code)
%------------------------------
%INITIALIZATIONS
%------------------------------
    sw                      =   algo_struct.sw;          %snippet width
    sh                      =   algo_struct.sh;          %snippet height
    sz                      =   [sh sw];
    D                       =   sw*sh;                   %dimensionality of input data

    Np                      =   options_struct.Np;        %particle filter: # of particles (samples) from density)
    rn1                     =   RandomData_cdf(f,:);     %pre-stored random numbers to ensure repeatability
    rn2(:,:)                =   RandomData_sample(f,:,:);%same as above

%------------------------------
%PRE-PROCESSING
%------------------------------
%1. resample (in some cases it's done at the end, here it's done at the beginning, same thing really)

    if ~isfield(cond_struct,'affineCandidates_6xNp')
        %one time: initialize 6 affine parameters, one for each of the Np candidate snippets
        cond_struct.affineCandidates_6xNp       ...
                                =   repmat(  affparam2geom(cond_struct.tstout_bestAffineParams(:)), [1,Np]  );        %initialize candidates (one time)
    else
        %recurring: resample distribution in 3 lines (read details of this in my article on resampling)
        cumconf             =   cumsum(cond_struct.weights);
        idx                 =   floor(sum(  repmat(rn1,[Np,1]) > repmat(cumconf,[1,Np])  ))+1;
        cond_struct.affineCandidates_6xNp       ...
                            =   cond_struct.affineCandidates_6xNp(:,idx);  %keep only good candidates (resample)
    end

%2. apply motion model (brownian, so just add randomness)
    cond_struct.affineCandidates_6xNp           ...
                            =   cond_struct.affineCandidates_6xNp + rn2.*repmat(options_struct.affsig(:),[1,Np]);

    %extract the candidate snippets from the image based on motion model above
    PFcandidateSnippets_0to1 =  warpimg(I_0to1, affparam2mat(cond_struct.affineCandidates_6xNp), sz);  %now create actual snippet candidates

%------------------------------
%PROCESSING
%------------------------------
%3a. weighting (find how well the algorithm model explains each snippet, find distances)
    if (algo_code==1) %i.e. iPCA

        diff_0to1           =   repmat(algo_struct.mean(:),[1,Np]) - reshape(PFcandidateSnippets_0to1,[D,Np]); %diff_0to1: (sw)(sh) x Np
        coefdiff            =   0;

        if (size(algo_struct.basis,2) > 0)
            coef            =   algo_struct.basis'*diff_0to1;
            diff_0to1       =   diff_0to1 - algo_struct.basis*coef;
            if (isfield(cond_struct,'coef'))
                coefdiff    =   (abs(coef)-abs(cond_struct.coef))*algo_struct.reseig./repmat(algo_struct.eigval,[1,Np]);
            else
                coefdiff    =   coef .* algo_struct.reseig ./ repmat(algo_struct.eigval,[1,Np]);
            end
            cond_struct.coef=   coef;
        end




    elseif(algo_code==2) %i.e. bPCA
        diff_0to1           =   [];
        for i = 1:Np
            Itst            =   255*PFcandidateSnippets_0to1(:,:,i);
            algo_struct     =   bPCA_3_test(Itst(:), algo_struct);
            diff_0to1(:,i)  =   algo_struct.tstout_err_Dx1/255;
        end



    elseif(algo_code==3) %i.e. RVQ
        diff_0to1           =   [];
        for i = 1:Np
            Itst            =   255*PFcandidateSnippets_0to1(:,:,i);
            algo_struct     =   RVQ_testing(Itst(:), algo_struct);
            diff_0to1(:,i)  =   algo_struct.tstout_err_Dx1/255;
        end



    elseif(algo_code==4) %i.e. TSVQ
        diff_0to1           =   [];
        for i = 1:Np
            Itst            =   255*PFcandidateSnippets_0to1(:,:,i);
            algo_struct     =   TSVQ_3_test(Itst(:), algo_struct);
            diff_0to1(:,i)  =   algo_struct.tstout_err_Dx1/255;
        end
```

```
    end

%3b. raise distances to exponentials
    if (~isfield(options_struct,'errfunc'))
        options_struct.errfunc ...
                            =   [];
    end

    switch (options_struct.errfunc)
        case 'robust';
            cond_struct.weights ...
                            =   exp(-sum(diff_0t1.^2./(diff_0t1.^2+options_struct.rsig.^2))./options_struct.condenssig)';
        case 'ppca';
            cond_struct.weights ...
                            =   exp(-(sum(diff_0t1.^2) + sum(coefdiff.^2))./options_struct.condenssig)';
        otherwise;
            cond_struct.weights ...
                            =   exp(-sum(diff_0t1.^2)./options_struct.condenssig)';
    end

%4. pick MAP estimate
    cond_struct.weights     =   cond_struct.weights ./ sum(cond_struct.weights);            %normalize weights
    [maxprob,maxidx]        =   max(cond_struct.weights);                                   %MAP estimate: pick best index
    cond_struct.tstout_bestAffineParams ...
                            =   affparam2mat(cond_struct.affineCandidates_6xNp(:,maxidx));  %MAP estimate: pick best affine ↘
                            →parameters based on best index
    cond_struct.tstout_bestPFcandidate_0t1 ...
                            =   PFcandidateSnippets_0t1(:,:,maxidx);                        %MAP estimate: pick best candidate↘
                            → snippet based on best index


%-----------------------------
%POST-PROCESSING
%-----------------------------
%error and reconstruction
    cond_struct.err_0t1     =   reshape(diff_0t1(:,maxidx), sz);                            %get reconstruction error
    if      (algo_code==1)
        cond_struct.err_0t1 =   -cond_struct.err_0t1;
    end
    cond_struct.recon       =   cond_struct.tstout_bestPFcandidate_0t1 - cond_struct.err_0t1; %get reconstructed image

%metrics
    cond_struct.tstout_snr  =   UTIL_METRICS_compute_SNR       (cond_struct.tstout_bestPFcandidate_0t1, cond_struct.err_0t1);
    cond_struct.tstout_rmse =   UTIL_METRICS_compute_rms_value (cond_struct.err_0t1(:)*255);
```

Listing 9: TRK_condensation.m.

B.6 Utility functions

B.6.1 RVQ

```
%% This function takes a grayscale or RGB image and creates a posneg image.
%
% Refer to readme.pdf for a description of posneg images.
%
% Copyright (C) Salman Aslam.  All rights reserved.
% Date created          : Feb 4, 2011
% Date last modified    : July 9, 2011.
%%

function Iposneg = RVQ__create_posnegImage(I, cfn_Iraw, bView, bSave)

%--------------------
%INITIALIZATIONS
%--------------------
    sz                      =   size(I);

    h                       =   sz(1);
    w                       =   sz(2);
    Nc                      =   6;                  %number of channels in raw image

%--------------------
%PRE-PROCESSING
%--------------------
%extract planar channels, if grayscale, make all channels the same
    if (length(sz)==3)                              %RGB image
        posR                =   I(:,:,1);
        posG                =   I(:,:,2);
        posB                =   I(:,:,3);
    elseif (length(sz)==2)                          %gray scale image
        posR                =   I;
        posG                =   posR;
        posB                =   posR;
    end

%vectorize (the reason is that we'll be stacking these in planar form)
    posR                    =   posR';
    posG                    =   posG';
    posB                    =   posB';

    posR                    =   posR(:);
    posG                    =   posG(:);
    posB                    =   posB(:);

%create negative channels
    negR                    =   255-posR;
    negG                    =   255-posG;
    negB                    =   255-posB;

%--------------------
%PROCESSING
%--------------------
%create posneg image
    Iposneg                 =   [posR; posG; posB; negR; negG; negB];

%--------------------
%POST-PROCESSING
%--------------------
%save it
    if (bSave)
        fid                 =   fopen(cfn_Iraw, 'w');
                                fwrite(fid, Iposneg);
                                fclose(fid);
    end

%view it
    if (bView)
        M                   =   reshape(Iposneg, w, h*Nc);
                                imshow(M')
                                title(cfn_Iraw);
                                pause
    end
```

Listing 10: "RVQ__create_posnegImage.m."

B.6.2 Signal metrics

```
%% This function computes the power signal of an input signal.
%
% Sig_Dx1: signal vector
% Pow_Dx1: power vector (normalized since we assume resistance R=1)

% Copyright (C) Salman Aslam.  All rights reserved.
% Data created        : April 15, 2011
% Date last modified : July 7, 2011
%%

function Pow_Dx1 =   UTIL_METRICS_compute_powerSignal(Sig_Dx1)

    Pow_Dx1                 =   Sig_Dx1.^2;
```

Listing 11: UTIL_METRICS_compute_powerSignal.m.

```
%% This function computes the total energy of an input signal.
%
```

```
% Sig_Dx1: signal vector
%
% Copyright (C) Salman Aslam.  All rights reserved.
% Data created       : April 15, 2011
% Date last modified : July 7, 2011
%%
function energy = UTIL_METRICS_compute_energy(Sig_Dx1)

    energy                  =   sum (  UTIL_METRICS_compute_powerSignal(Sig_Dx1) );
```

Listing 12: UTIL_METRICS_compute_energy.m.

```
%% This function computes the power of a signal.
%
% Sig_Dx1: signal vector

% Copyright (C) Salman Aslam.  All rights reserved.
% Data created       : March 17, 2011
% Date last modified : July 7, 2011
%%
function power = UTIL_METRICS_compute_power(Sig_Dx1)

    power                   =   UTIL_METRICS_compute_energy(Sig_Dx1) / length(Sig_Dx1);    %norm(s,2) is the same as sqrt(sum(↘
    →s.^2)), or square root of energy
```

Listing 13: UTIL_METRICS_compute_power.m.

```
%% This function computes the rms energy of a signal.
%
% Sig_Dx1: signal vector

% Copyright (C) Salman Aslam.  All rights reserved.
% Data created       : March 17, 2011
% Date last modified : July 7, 2011
%%

function rms_value = UTIL_METRICS_compute_rms_value(Sig_Dx1)

    rms_value               =   sqrt( UTIL_METRICS_compute_power(Sig_Dx1) );
```

Listing 14: UTIL_METRICS_compute_rms_value.m.

```
%% This function computes SNR given a signal and an error signal.
%
% Sig_Dx1: signal vector
% Err_Dx1: error (noise) vector

% Copyright (C) Salman Aslam.  All rights reserved.
% Data created       : March 17, 2011
% Date last modified : July 7, 2011
%%

function SNR = UTIL_METRICS_compute_SNR(Sig_Dx1, Err_Dx1)

%signal
    power_signal            =   UTIL_METRICS_compute_power (Sig_Dx1); %Sig_Dx1 can also be a row vector

%error
    power_noise             =   UTIL_METRICS_compute_power (Err_Dx1); %Err_Dx1 can also be a row vector

%SNR
    SNR                     =   power_signal / power_noise;
```

Listing 15: UTIL_METRICS_compute_SNR.m.

```
%% This function computes SNR in dB given a signal and an error signal.
%
% Copyright (C) Salman Aslam.  All rights reserved.
% Data created       : March 20, 2011
% Date last modified : July 7, 2011
%%

function SNRdB = UTIL_METRICS_compute_SNRdB(Sig_Dx1, Err_Dx1)  %although shown as column vectors, they can be row vectors as ↘
→well

    SNR                     =   UTIL_METRICS_compute_SNR (Sig_Dx1, Err_Dx1);
    SNRdB                   =   10*log10            (SNR);
```

Listing 16: UTIL_METRICS_compute_SNRdB.m.

```
%% This function computes the PSNR of an input signal.
%
% Err_Dx1: error (noise) vector
%
% Copyright (C) Salman Aslam.  All rights reserved.
% Data created       : March 20, 2011
% Date last modified : July 7, 2011
%%
function psnr_dB = UTIL_METRICS_compute_PSNRdB(max_signal_value, Err_Dx1)

    power_noise             =   UTIL_METRICS_compute_power(Err_Dx1);
    psnr_dB                 =   10*log10(max_signal_value^2 / power_noise);
```

Listing 17: UTIL_METRICS_compute_PSNRdB.m.

```
%% This function computes training error, i.e., loss
%
% The training error computed is for DM2, i.e., all the training data.
%
% For computing SNRdB, the following approach is taken:
% (a) for each training vector, compute the error vector algo_struct.tstout_err_Dx1
% (b) concatenate all training vectors into S_NDx1 to make one giant signal
% (c) concatenate all error vectors into E_NDx1 to make one giant error signal
% (d) compute SNRdB using these giant signals
%
% The reason for this approach is that it's similar to the approach taken
% in Explorer, Dr Barnes' software.
%
% Copyright (C) Salman Aslam.  All rights reserved.
% Data created       : April 20, 2011
% Date last modified : July 7, 2011
%%

function algo_struct = UTIL_METRICS_compute_training_error(DM2, algo_struct, algo_code)

%----------------
%INITIALIZATIONS
%----------------
    [D, N]                  =   size(DM2);
    S_NDx1                  =   [];             %all N D-dimensional training vectors are concatentated to form a large signal
    E_NDx1                  =   [];             %all N D-dimensional error vectors are concatentated to form a large signal
    algo_struct.trgout_descriptors = [];        %the above step is carried out to be compatible with the way Explorer computes↘
    → training error

%--------------------
%PRE-PROCESSING
%--------------------
%!!caution, this needs checking
    if      (algo_code==1)  str='algo_struct_____________=___bPCA_3_test_______(tst_vec_Dx1,_algo_struct);';  %probabilistic ↘
    →PCA
%!!end caution
    elseif  (algo_code==2)  str='algo_struct_____________=___bPCA_3_test_______(tst_vec_Dx1,_algo_struct);';  %batch PCA
    elseif  (algo_code==3)  str='algo_struct_____________=___RVQ__testing_____(tst_vec_Dx1,_algo_struct);';  %RVQ
    elseif  (algo_code==4)  str='algo_struct_____________=___TSVQ_3_test_______(tst_vec_Dx1,_algo_struct);';  %TSVQ
    end

%--------------------
%PROCESSING
%--------------------
        for i=1:N

        %pick out a single vector from DM2, this is one training example
        tst_vec_Dx1         =   DM2(:,i);

        %evaluate error against model
                            eval(str);       %so for RVQ, test against the RVQ codebook

        %concatenate signals (training vectors) and errors
        S_NDx1              =   [S_NDx1; tst_vec_Dx1];                  %save the SNR
        E_NDx1              =   [E_NDx1; algo_struct.tstout_err_Dx1]; %save the error
    end

    %compute SNR for the large signal
    snr_dB                  =   UTIL_METRICS_compute_SNRdB      (S_NDx1, E_NDx1); %S_NDx1 is now one big signal
    rmse                    =   UTIL_METRICS_compute_rms_value  (E_NDx1);

%--------------------
%POST-PROCESSING
%--------------------
%save results
    algo_struct.trgout_snr  =   snr_dB;
    algo_struct.trgout_rmse =   rmse;
```

Listing 18: UTIL_METRICS_compute_training_error.m.

B.6.3 Indexing

```
%Copyright (C) Salman Aslam.  All rights reserved.
%Date: July 7, 2011.
%%
%Description:
%- This file takes an index and image width (Iw) and returns (x,y) coordinates.
%%

function [y,x] = UTIL_idx_to_xy(idx, Iw)

%go to C style indexing (0-based)
    idx = idx-1;

%compute x, y in C-style
    y = idivide(  int32(idx), int32(Iw), 'floor'  );
    x = mod(int32(idx), int32(Iw));

%back to Matlab style (1-based)
    y = y+1;
    x = x+1;
```

Listing 19: UTIL_idx_to_xy.m.

B.6.4 Plotting

```
%Downloaded from web (don't know from where) and modified.  All rights reserved.
%Date: July 7, 2011.
%%
%Description:
%This file displays several images in one figure.
%%

function UTIL_PLOT_tightsubplot(numRows, numCols, idx, I_rgb)

    colormap('gray');

    [row, col]      =   UTIL_idx_to_xy(idx, numCols);
    row             =   double(row-1);
    col             =   double(col-1);
    x               =   col*(1/numCols);
    y               =   (numRows-row-1)*(1/numRows);


    w               =   1/numCols-.001;
    h               =   1/numRows-0.001;
                        subplot('position', [x, y, w, h]);

                        imagesc((I_rgb));
                        axis equal
                        axis off;
```

Listing 20: UTIL_IMG_tightsubplot.m.

B.6.5 Files

```
%% This function throws an exception if the fileID is invalid, i.e. -1

% Copyright (C) Salman Aslam.  All rights reserved.
% Date created          : Feb 19, 2011
% Date last modified    : July 9, 2011.
%%

function UTIL_FILE_checkFileOpen(fid, cfn)

    try
        if (fid == -1)
            err = MException('ResultChk:FileNotOpen', ['salman: ' cfn ' could not be opened']);
            throw(err);
        end

    catch err
        disp(err.message);
        rethrow(err);
    end
```

Listing 21: UTIL_FILE_checkFileOpen.m

```
function UTIL_FILE_deleteFile(cfn)   %cfn is complete file name

    if      (ispc)      [status, result] = system(['del ' cfn]);
    elseif  (isunix)    [status, result] = unix(['rm ' cfn]);
    end
```

Listing 22: UTIL_FILE_deleteFile.m

References