**The MATLAB codes for chromatic shift correction by polynomial fitting.**

%-----------------------

% runme.m

%-----------------------

%MAIN ENTRY of the software

%

% PARAMETERS TO BE SET

% train\_filename: csv file containing the x,y locations of each of the

% three colour channels for N samples

%

% d: degree of the polynomial to be used for fitting (set to 1 by default)

%

% val\_filename: csv file containing the x,y locations of each of the

% three colour channels for samples in the validation set

%

% test\_filename: csv file containing the x,y locations of each of the

% three colour channels for samples in the test set

%

clear all; close all; clc;

mean\_err = [];

% set the degree of the polynomial to be used

d = 1;

my\_clock = '20120518\_1800'; % a string that contains the current time stamp, here is an example

csv\_predfname= sprintf('pred\_degree\_%d\_%s.csv', d, my\_clock);

% the prediction result is saved in this file

% Train the model to obtain the coefficients of the polynomial

[drgx\_coeff, drgy\_coeff, dcgx\_coeff, dcgy\_coeff] = my\_train\_new('20120518-BeadsCentroid-1-7-allInOneSheet.csv', d);

% validate the trained model (optional)

mean\_err(d+1,:) = my\_val\_new('20120518-BeadsCentroid-a.csv', drgx\_coeff, drgy\_coeff, dcgx\_coeff, dcgy\_coeff, d);

% Use the trained model on the test data to obtain predictions

[x\_r2gfp\_res, y\_r2gfp\_res, x\_c2gfp\_res, y\_c2gfp\_res] = my\_test\_new('20120518-BeadsCentroid-a.csv', csv\_predfname, drgx\_coeff, drgy\_coeff, dcgx\_coeff, dcgy\_coeff, d);

%-----------------------

% my\_train\_new.m

%-----------------------

function [drgx\_coeff, drgy\_coeff, dcgx\_coeff, dcgy\_coeff] = my\_train\_new(train\_filename, d)

%MY\_TRAIN\_NEW Use training data to calculate the coeffecients of variable

%degree polynomial, which best fits the observations

%

% INPUTS

% train\_filename: csv file containing the x,y locations of each of the

% three colour channels for N samples

% d: degree of the polynomial to be used for fitting

%

% OUTPUTS

% drgx\_coeff, drgy\_coeff: Mx1 vectors containing coefficients for fitting and subsequent correcting of X and Y data for G channel

% dcgx\_coeff, dcgy\_coeff: Mx1 vectors containing coefficients for fitting and subsequent correcting of X and Y data for CY5 channel

%

% M is the number of terms in a polynomial of degree d

%

%

%

% default lookup file name in case it wasn't provided as input

if nargin < 1

train\_filename = 'Beads\_Centroid-1-9.csv';

end

% Load the dataset

my\_dataset = my\_load\_dataset\_csv(train\_filename);

[dummy, c] = size(my\_dataset);

assert(dummy==1);

result\_dataset = cell(1, c);

% Extract out the x,y coordinates for the 3 color channels

% Each is a Nx1 vector

[x\_gfp, y\_gfp, x\_rfp, y\_rfp, x\_cy5, y\_cy5] = my\_create\_vector(my\_dataset);

% Calculating the emperically observed shift

delta\_x\_rfp\_gfp = x\_gfp - x\_rfp;

delta\_y\_rfp\_gfp = y\_gfp - y\_rfp;

% Building the polynomial and the data matrix A

% Example, for d=1: a00 \* x^0\*y^0+ a01 \* x^0\*y^1+ a10 \* x^1\*y^0

% and A = [(x\_rfp.^0).\*(y\_rfp.^0) (x\_rfp.^0).\*(y\_rfp.^1) (x\_rfp.^1).\*(y\_rfp.^0)];

[~, A] = make\_polynomial(d,x\_rfp,y\_rfp);

% QR decomposing A

[Q,R]=qr(A,0);

% Treating like a system of linear equations Ax = B

% where x is the coefficients of the polynomial and B = Q'\*delta

drgx\_coeff = R\(Q'\*delta\_x\_rfp\_gfp);

drgy\_coeff = R\(Q'\*delta\_y\_rfp\_gfp);

% A similar procedure follows for the CY5 correction

delta\_x\_cy5\_gfp = x\_gfp - x\_cy5;

delta\_y\_cy5\_gfp = y\_gfp - y\_cy5;

[~, A] = make\_polynomial(d,x\_cy5,y\_cy5);

[Q,R]=qr(A,0);

dcgx\_coeff = R\(Q'\*delta\_x\_cy5\_gfp);

dcgy\_coeff = R\(Q'\*delta\_y\_cy5\_gfp);

% Storing the results and writing them out to a .csv file

result\_dataset{1}c = [drgx\_coeff, drgy\_coeff, dcgx\_coeff, dcgy\_coeff];

csvwrite(['train\_degree' '\_' int2str(d) '\_coeff.csv'], result\_dataset);

%-----------------------

% my\_val\_new.m

%-----------------------

function [mean\_err] = my\_val\_new(val\_filename, drgx\_coeff, drgy\_coeff, dcgx\_coeff, dcgy\_coeff, d)

%MY\_VAL\_NEW Validate the trained model on a test set.

%

% INPUTS

% val\_filename: csv file containing the x,y locations of each of the

% three colour channels for N samples of the validation data

% d: degree of the polynomial to be used for fitting

% drgx\_coeff, drgy\_coeff, dcgx\_coeff, dcgy\_coeff: learned coefficients

%

% OUTPUTS

% mean\_err: The mean error in estimation of the corrected locations. This

% is a 1x4 vector, containing errors in x,y for the two channels

%

% default lookup file name in case it wasn't provided as input

if nargin < 1

val\_filename = 'Beads\_Centroid-10-11.csv';

end

% Load the dataset

my\_dataset = my\_load\_dataset\_csv(val\_filename);

[dummy, c] = size(my\_dataset);

assert(dummy==1);

mean\_err = zeros(1,4);

% Extract out the x,y coordinates for the 3 color channels

% Each is a Nx1 vector

[x\_gfp,y\_gfp,x\_rfp,y\_rfp,x\_cy5,y\_cy5] = my\_create\_vector(my\_dataset);

% Make the data matrix and multiply with the learned coefficients to get

% the deltas

[~, A] = make\_polynomial(d,x\_rfp,y\_rfp);

delta\_x\_rfp\_gfp\_res = A\*drgx\_coeff;

delta\_y\_rfp\_gfp\_res = A\*drgy\_coeff;

% Add obs. value back to get the predicted location

x\_r2gfp\_res = delta\_x\_rfp\_gfp\_res + x\_rfp;

y\_r2gfp\_res = delta\_y\_rfp\_gfp\_res + y\_rfp;

% Difference between the actual and the predicted locations

x\_tmp\_r2g = x\_gfp-x\_r2gfp\_res;

y\_tmp\_r2g = y\_gfp-y\_r2gfp\_res;

% Calculating the mean errors

mean\_err(1) = mean(abs(x\_tmp\_r2g));

mean\_err(2) = mean(abs(y\_tmp\_r2g));

figure,

subplot(1, 2, 1), scatter(x\_tmp\_r2g, y\_tmp\_r2g);

msg = sprintf('d=%d, rfp2gfp', d); title(msg);

% A similar procedure follows for the CY5 correction

[~, A] = make\_polynomial(d,x\_cy5,y\_cy5);

delta\_x\_cy5\_gfp\_res = A\*dcgx\_coeff;

delta\_y\_cy5\_gfp\_res = A\*dcgy\_coeff;

x\_c2gfp\_res = delta\_x\_cy5\_gfp\_res + x\_cy5;

y\_c2gfp\_res = delta\_y\_cy5\_gfp\_res + y\_cy5;

x\_tmp\_c2g = x\_gfp-x\_c2gfp\_res;

y\_tmp\_c2g = y\_gfp-y\_c2gfp\_res;

mean\_err(3) = mean(abs(x\_tmp\_c2g));

mean\_err(4) = mean(abs(y\_tmp\_c2g));

subplot(1, 2, 2), scatter(x\_tmp\_c2g, y\_tmp\_c2g);

msg = sprintf('d=%d, cy52gfp', d); title(msg);

result\_dataset = [x\_gfp, y\_gfp, x\_r2gfp\_res, y\_r2gfp\_res, x\_c2gfp\_res, y\_c2gfp\_res];

csvwrite(['val\_degree' '\_' int2str(d) '\_pred.csv'], result\_dataset);

% print the errors out

mean\_err

%-----------------------

% my\_test\_new.m

%-----------------------

function [x\_r2gfp\_res, y\_r2gfp\_res, x\_c2gfp\_res, y\_c2gfp\_res] = my\_test\_new(test\_filename, csv\_outfname, drgx\_coeff, drgy\_coeff, dcgx\_coeff, dcgy\_coeff, d)

%MY\_TEST\_NEW Use the learned coeffecients of a polynomial

% to correct for chromatic aberration in the test data

%

% INPUTS

% test\_filename: csv file containing the x,y locations of the test data

% d: degree of the polynomial to be used for fitting

% drgx\_coeff, drgy\_coeff: Mx1 vectors containing the learned coefficients G channel correction

% dcgx\_coeff, dcgy\_coeff: Mx1 vectors containing the learned coefficients

% CY5 channel correction

%

% OUTPUTS

% x\_r2gfp\_res, y\_r2gfp\_res: Corrected coordinates for G channel of size Nx1

% x\_c2gfp\_res, y\_c2gfp\_res: Corrected coordinates for CY5 channel, size Nx1

%

%

% default test file name in case it wasn't provided as input

if nargin < 1

test\_filename = 'Noc-Golgi-Mini-Stack\_Centroid.xls';

end

% Load the test data

my\_dataset = my\_load\_dataset\_csv(test\_filename);

[~, c] = size(my\_dataset);

result\_dataset = cell(1, c);

% Extract out the x,y coordinates for the 3 color channels

% Each is a Nx1 vector

[~,~,x\_rfp,y\_rfp,x\_cy5,y\_cy5] = my\_create\_vector(my\_dataset);

% GFP correction

% first build the polynomial and the data matrix.

% A\*learned coefficients will give delta

[~, A] = make\_polynomial(d,x\_rfp,y\_rfp);

delta\_x\_rfp\_gfp\_res = A\*drgx\_coeff;

delta\_y\_rfp\_gfp\_res = A\*drgy\_coeff;

% Adding back the original location to delta and obtain the corrected loc.

x\_r2gfp\_res = delta\_x\_rfp\_gfp\_res + x\_rfp;

y\_r2gfp\_res = delta\_y\_rfp\_gfp\_res + y\_rfp;

% A similar procedure follows for the CY5 correction

[~, A] = make\_polynomial(d,x\_cy5,y\_cy5);

delta\_x\_cy5\_gfp\_res = A\*dcgx\_coeff;

delta\_y\_cy5\_gfp\_res = A\*dcgy\_coeff;

x\_c2gfp\_res = delta\_x\_cy5\_gfp\_res + x\_cy5;

y\_c2gfp\_res = delta\_y\_cy5\_gfp\_res + y\_cy5;

% Storing the results and writing them out to a .csv file

result\_dataset{1} = [x\_r2gfp\_res, y\_r2gfp\_res, x\_c2gfp\_res, y\_c2gfp\_res];

%csvwrite(['test\_degree' '\_' int2str(d) '\_pred.csv'], result\_dataset);

csvwrite(csv\_outfname, result\_dataset);

%-----------------------

% my\_create\_vector.m

%-----------------------

function [x\_gfp,y\_gfp,x\_rfp,y\_rfp,x\_cy5,y\_cy5] = my\_create\_vector(dataset)

%MY\_CREATE\_VECTOR Extract location data as separate vectors from the loaded

%cell array

%

% INPUTS

% dataset: csv file loaded into memory

%

% OUTPUTS

% x\_gfp, y\_gfp: x,y info for the red channel

% x\_rfp, y\_rfp: x,y info for the green channel

% x\_cy5, y\_cy5: x,y info for the CY5 channel

% Example usage:

% data = my\_load\_dataset\_csv('traindata.csv')

% [x\_gfp,y\_gfp,x\_rfp,y\_rfp,x\_cy5,y\_cy5] = my\_create\_vector(data)

%

%

[~,c] = size(dataset);

x\_gfp = [];

y\_gfp = [];

x\_rfp = [];

y\_rfp = [];

x\_cy5 = [];

y\_cy5 = [];

for i = 1:c

vect = dataset{i};

x\_gfp = [x\_gfp; vect(:,1)];

y\_gfp = [y\_gfp; vect(:,2)];

x\_rfp = [x\_rfp; vect(:,3)];

y\_rfp = [y\_rfp; vect(:,4)];

x\_cy5 = [x\_cy5; vect(:,5)];

y\_cy5 = [y\_cy5; vect(:,6)];

end

%-----------------------

% my\_load\_dataset\_csv.m

%-----------------------

function dataset = my\_load\_dataset\_csv(filename)

%MY\_LOAD\_DATASET\_CSV Read from a given csv file into a cell array for

%subsequent calculations

%

% INPUTS

% filename: csv file containing the x,y locations of each of the

% three colour channels for N samples

%

% OUTPUTS

% dataset: cell array containing the data loaded into memory

%

%

valid\_vect = csvread(filename);

dataset{1} = valid\_vect ;

%-----------------------

% make\_polynomial.m

%-----------------------

function [function\_expr, A] = make\_polynomial(degree,x,y)

%MAKE\_POLYNOMIAL Given the degree of the required polynomial and the data

%points (bi variate), this function builds the polynomial structure and the

%data matrix

%

% INPUTS

% degree: degree of the polynomial to be used for fitting

% x, y: Nx1 vectors containing the data points

%

% OUTPUTS

% function\_expr: The polynomial expression (String)

% A: NxM data matrix where,

% M is the number of terms in a polynomial of degree d

% and Aij is the jth polynomial term (without the coefficient) of the ith

% sample.

%

%

expr = [];

A = [];

for i = 0: degree

for j=0:degree - i

% adding the corresponding term to the existing polynomial

expr = [expr '+a' int2str(i) int2str(j) ' \* ' 'x^' int2str(i) ' \* ' 'y^' int2str(j) ];

A = [A (x.^i).\*(y.^j)];

end

end

% removing the , upfront

expr = expr(1, 2:size(expr,2));

function\_expr = expr ;

end