Ben Stonebraker
October 20, 2019
Image Processing – Project 2B

Image Processing Project 2B

In this project we analyzed the invariant moments, mean, and standard deviations of objects the computer thought were hay bales based on the segmentation code provided. Firstly, we used the code provided in detectionHayBalesSimple.m (Appendix A) and modified it so that we could calculate the invariant moments, mean, and standard deviation in inv_moments.m (Appendix B). Then once the image statistics were calculated, the user was asked to determine whether the section was a haybale or not, and the modified detectionHayBalesSimple.m was able to put the data in the appropriate dataset. The 2 datasets were then returned to the user in main.m (Appendix C), and then the crossplot and Mahalanobis Distance plot were generated based on that data. The statistics generated are in a table located in Appendix D.

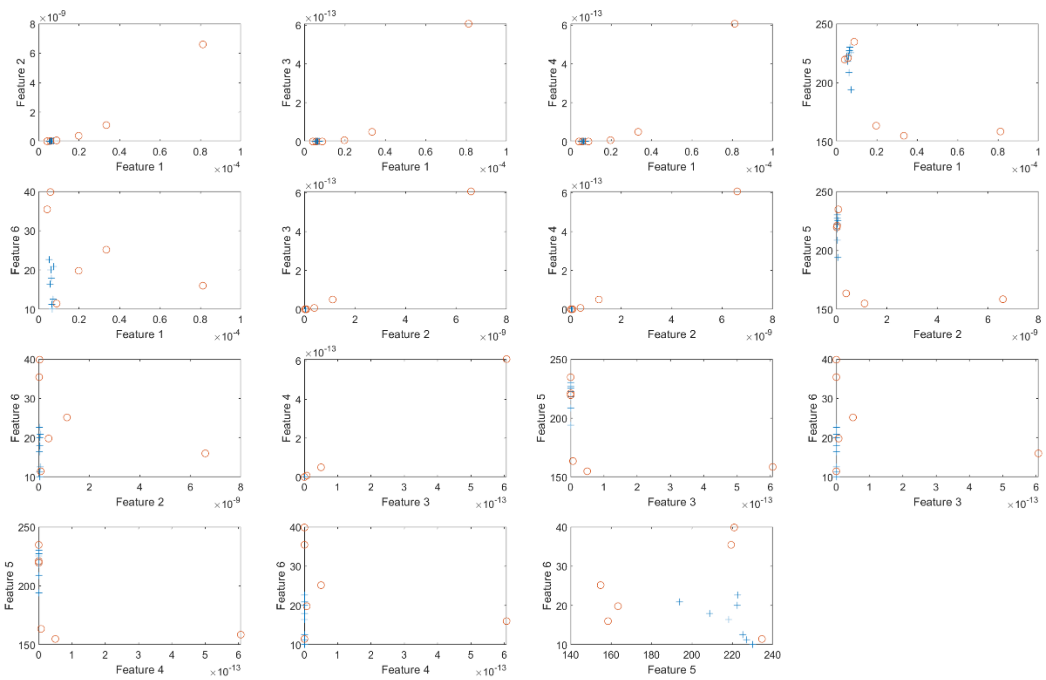

Fig. 1 – Crossplot of the two data sets. Where Features 1-6 correspond to Phi1-4, mean, and standard deviation respectively for IMG_0034.png

Ben Stonebraker
October 20, 2019
Image Processing – Project 2B

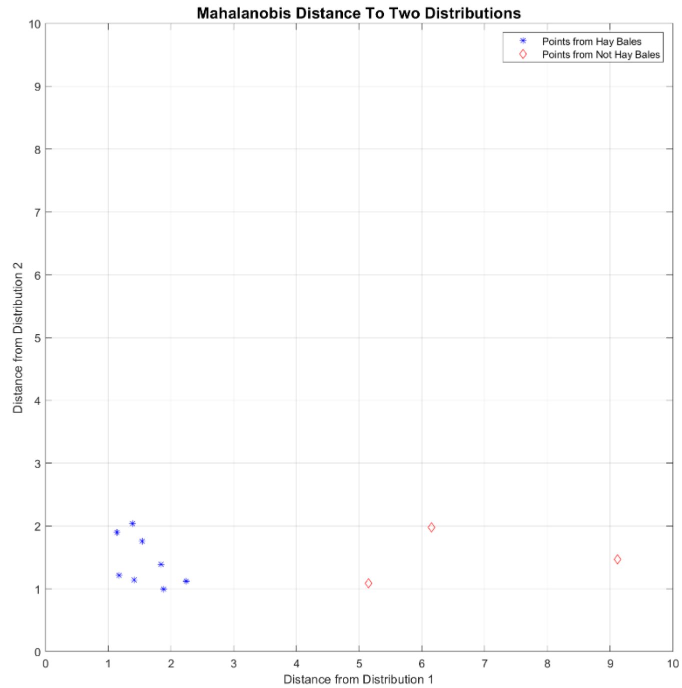

Fig. 2 Mahalanobis Distance Plot, where Distribution 1 is the Hay Bale sample and Distribution 2 Not a Hay Bale for IMG_0034.png

The cross plot ended up being fairly useless as one can barely see any real clustering of either data sets. To solve this, we used the Mahalanobis distance plot to try and better segregate the 2 data sets. As shown in Figure 2, the data is much more clustered together and a clear dividing line can be made to segregate the two data sets.

This process was then repeated for IMG_0038.png. The crossplot and the Mahalanobis distance plot are shown below, and the code/table is located in Appendices E-G. The detectHayBalesSimple.m was replaced with a modified version, detect2.m (Appendix E). The only real changes were changing the filter parameters and then using a different size for the area filter. Then main2.m (Appendix F) was used to process and plot the data, though there were no real significant changes besides the image file name and which detection function it called. It's worth noting the crossplot is especially unhelpful since we have 37 non-haybales and only 5 haybales, cluttering any haybale points that would be visible.

Ben Stonebraker
October 20, 2019
Image Processing – Project 2B

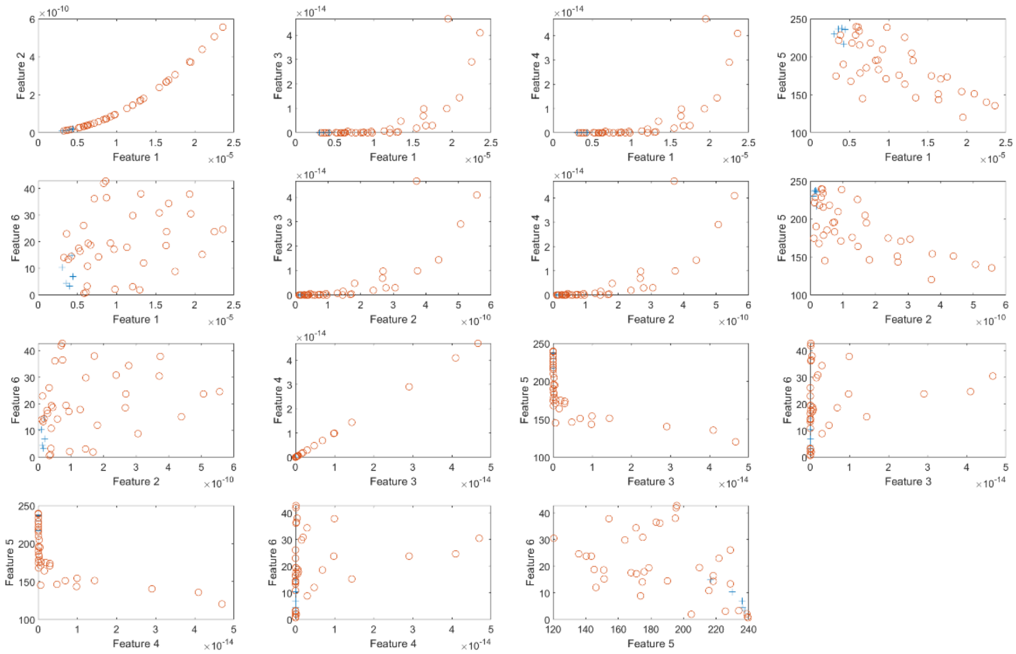

Fig. 3 – Crossplot of data for IMG_0038.png where Features 1-6 are Phi1-4, Mean, and Standard Deviation respectively.

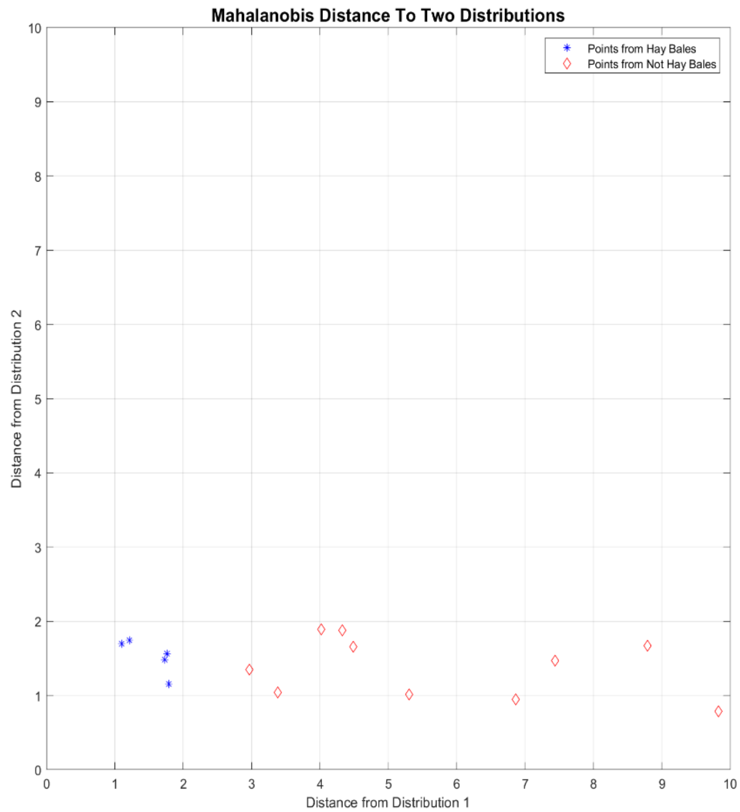

Fig. 2 - Mahalanobis Distance Plot, where Distribution 1 is the Hay Bale sample and Distribution 2 Not a Hay Bale for IMG_0038.png

Appendix A

```
function [hay_stats,not_hay_stats] = detectionHayBalesSimple(file)

close all
IMG_0034 = imread(file);
%take the fft
IMG = fft2(IMG_0034);
e_IMG = edge( IMG_0034, 'canny' );

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FILTER IMAGE
% filtering of FT across rows.
FLT_IMG = BW_Filtering( IMG, 2, 1/50 );
% filtering of FT across columns.
FLT_IMG = BW_Filtering( FLT_IMG, 1, 1/50 );
FLT_IMG = HP_Filtering( FLT_IMG, 2, 1/250 );
FLT_IMG = HP_Filtering( FLT_IMG, 1, 1/250 );
flt_img = real( ifft2( FLT_IMG ) );
figure(1);
imagesc( flt_img ), colormap('gray');

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%SET THRESHOLDS
%0.7 * max + 0.3 * min
thresholdMax = max(max(flt_img));
thresholdMin = min(min(flt_img));
t_img = zeros(size(flt_img));
t_img(flt_img>(thresholdMax*0.7+thresholdMin*0.3)) = 1;
figure(2);
imagesc( t_img ), colormap('gray');

%%

% create connnect or Labeled image.
[L_img,Labels] = SizeOfRegions( t_img, 500, 1500 );
figure(3);
imagesc( L_img ), colormap('gray');

% prepare to work through connected sets.
[N,M] = size( IMG_0034 );
D_img = zeros( N, M );
m = 5; % counts figures.
hayCnt = 1;
notHayCnt = 1;
% loop through each labelled set.
for k = 1:Labels
    [Row, Col] = find( L_img == k );  % find list of pixels labeled k
    % if size of region is appropriate.
    if length( Row ) > 500 && length( Row ) < 1500
        % compute center of blob.
        Center = round( sum( [Row Col] )/length(Row) );
        % extract image around blob.
        r = max(1,Center(1)-70):min(N,Center(1)+70);
        c = max(1,Center(2)-70):min(M,Center(2)+70);
```

```
        Local = IMG_0034( r, c );
        Mask = t_img( r, c );
        % display local image.
        figure(m)
        subplot( 221 ), imagesc( Local ), colormap('gray');
        % create threshold from top and bottom band of pixels.
        I = 1:5*141;
        Mu_O = mean( double( [Local(I) Local(end-I+1)] ) );
        Std_O = std( double( [Local(I) Local(end-I+1)] ) );

        % Create threshold.
        Thres = Mu_O + 2.5*Std_O;
        figure(100); % figure to display process.
        Local = GrowAroundBlobs( Mask, edge(Local,'canny',[],2), ...
                                 Local, Thres );
        im_stats = inv_moments(Local);
        confirm = input("Is this a haybale? (1/0)");
        if confirm == 1
            hay_stats(hayCnt) = im_stats;
            hayCnt = hayCnt + 1;
        else
            not_hay_stats(notHayCnt) = im_stats;
            notHayCnt = notHayCnt + 1;
        end
        figure(m); % move back to current figure.
        m = m + 1;  % move to next figure
        % Display segmented image.
        subplot( 222 ), imagesc( Local ), colormap('gray');
        %pause(2)
        close 100 % remove demo figure
        % save off segmented image.
        D_img( r, c ) = Local;
    else
      t_img( [Row Col] ) = 0;
    end
end

figure;
subplot(211),imagesc( D_img ), colormap('gray');
D_img( D_img > 0 ) = 1;
subplot(212),imagesc( D_img ), colormap('gray');

% %%
% e1_IMG = imdilate( e_IMG, ones(5,5) );
% e2_IMG = imerode( e1_IMG, ones(5,5) );
% %DD_img = Fill2Edge_Matlab( e2_IMG, D_img );
% %DD_img = imfill( D_img, find(  );
%
% figure;
% imagesc( DD_img ), colormap('gray');
return
```

Appendix B:

```
function image_stats = inv_moments(image)

[m,n] = size(image);
if m ~= n
    if m > n
        padImage = padarray(image, [2/2, (m-n+2)/2]);
    else
        padImage = padarray(image, [(n-m+2)/2, 2/2]);
    end
else
    padImage = image;
end
image = padImage;
[m, n] = size(padImage);

Dx = [];Dy = [];
Dx = uint8(repmat(1:n, m ,1));
Dy = uint8(repmat((1:n)',1,m));

m00 = sum(sum(image));
m10 = sum(sum(Dx.*image));
m01 = sum(sum(Dy.*image));
m20 = sum(sum((Dx.^2).*image));
m02 = sum(sum((Dy.^2).*image));
m22 = sum(sum((Dy.^2).*(Dx.^2).*image));
m03 = sum(sum((Dy.^3).*image));
m30 = sum(sum((Dx.^3).*image));
m11 = sum(sum(Dx.*Dy.*image));
m12 = sum(sum(Dx.*(Dy.^2).*image));
m21 = sum(sum((Dx.^2).*(Dy).*image));

x_bar = m10 / m00;
y_bar = m01 / m00;

mu30 = m30 - 3*x_bar*m20 + 2*m10*(x_bar^2);
mu12 = m12 - 2*y_bar*m11 - x_bar*m02+2*(y_bar^2)*m10;
mu21 = m21 - 2*x_bar*m11 - y_bar*m20+2*(x_bar^2)*m01;
mu03 = m03 - 3*y_bar*m02 + 2*(y_bar^2)*m01;

nu20 = sum(sum((Dx-x_bar).^2.*image))/(m00^2);
nu02 = sum(sum((Dy-y_bar).^2.*image))/(m00^2);
nu11 = sum(sum((Dy-y_bar).*(Dx-x_bar).*image))/(m00^2);
nu12 = mu12/(m00^((1+2)/2 + 1));
nu21 = mu21/(m00^((1+2)/2 + 1));
nu03 = mu03/(m00^((0+3)/2 + 1));
nu30 = mu30/(m00^((3+0)/2 + 1));

Phi1 = nu20 + nu02;
Phi2 = (nu20 - nu02)^2 + 4*nu11^2;
Phi3 = (nu30  - 3*nu12)^2 + (3*nu21-nu03)^2;
Phi4 = (nu30 + nu12)^2 + (nu21 + nu03)^2;
```

Ben Stonebraker
October 20, 2019
Image Processing – Project 2B

```
image_stats = struct;

image_stats.Phi1 = Phi1;
image_stats.Phi2 = Phi2;
image_stats.Phi3 = Phi3;
image_stats.Phi4 = Phi4;
%% Not Inv moments
image_stats.mean = mean(mean(image(find(image))));
image_stats.std = std(double(image(find(image))));
```

Appendix C:

```
[hay_stats,not_hay_stats] = detectionHayBalesSimple('IMG_0034.png');
close all
%% Part 2
haymat = [];
not_haymat = [];
for i=1:1:length(hay_stats)
    imstat = hay_stats(i);
    haymat(i,:) =  [imstat.Phi1, imstat.Phi2, imstat.Phi3, imstat.Phi4,
imstat.mean, imstat.std;];
end
for i=1:1:length(not_hay_stats)
    imstat = not_hay_stats(i);
    not_haymat(i,:) =  [imstat.Phi1, imstat.Phi2, imstat.Phi3, imstat.Phi4,
imstat.mean, imstat.std;];
end
Crossplot(haymat,not_haymat);
hay_cov = cov(haymat);
not_hay_cov = cov(not_haymat);
hay_mean = mean(haymat);
not_hay_mean = mean(not_haymat);
%% Mahalanobis Part
d0_0 = MahalanobisDist(haymat',hay_cov,hay_mean');
d0_1 = MahalanobisDist(haymat',not_hay_cov,not_hay_mean');
d1_1 = MahalanobisDist(not_haymat',not_hay_cov,not_hay_mean');
d1_0 = MahalanobisDist(not_haymat',hay_cov, hay_mean');
figure(2)
set(2, 'units', 'normalized','outerposition', [0 0 1 1] );
plot( d0_0, d0_1, 'b*', d1_0, d1_1, 'rd' );
hold on
grid on
title( 'Mahalanobis Distance To Two Distributions');
set(findall(2,'type','text'),'fontSize',14);
legend( 'Points from Hay Bales', 'Points from Not Hay Bales' );
xlabel( 'Distance from Distribution 1' );
ylabel( 'Distance from Distribution 2' );
axis('equal')
ylim([0 10]);
xlim([0 10]);
```

Appendix D:

Hay Bales:

Phi1	Phi2	Phi3	Phi4	Mean	Std. Dev.
6.80715490558697e-06	4.63373579086567e-11	4.88971463778958e-18	4.88971463778958e-18	230.185289957567	10.0745787311848
6.57229758903253e-06	4.31950955988029e-11	5.85631745297849e-18	5.85631745297844e-18	227.220226214238	11.3011702115839
7.15726567525681e-06	5.12264519462093e-11	9.00247034087161e-18	9.00247034087157e-18	225.282763532764	12.5476353327220
5.63561848619419e-06	3.17601957219337e-11	7.76662945985196e-18	7.76662945985196e-18	218.241578947368	16.3742453610653
6.31046466116929e-06	3.98219642398665e-11	2.10587371879264e-17	2.10587371879264e-17	208.728840970350	17.8966318630964
5.99077851285991e-06	3.58894271901440e-11	6.66168482095960e-18	6.66168482095960e-18	222.538685282141	20.0305816456851
5.29900931754632e-06	2.80794997474427e-11	4.57970202360055e-18	4.57970202360055e-18	222.619464469619	22.6226098215505
7.26293801404461e-06	5.27502685958542e-11	7.71645010603021e-17	7.71645010603022e-17	193.831460674157	20.9470361739146

Not Hay Bales:

Phi1	Phi2	Phi3	Phi4	Mean	Std. Dev.
8.83599503101088e-06	7.80748081880489e-11	6.55093400517333e-18	6.55093400517333e-18	234.680343511450	11.4464792147995
5.75734332372828e-06	3.26751205008032e-11	6.67422751246153e-18	6.74326930798550e-18	220.961645358533	39.8460633149147
4.16365697043419e-06	1.73360393674452e-11	2.86274609372200e-18	2.86274609372200e-18	219.426493710692	35.4249315926238
3.34354937823673e-05	1.11793224447072e-09	4.99098504793670e-14	4.99098504793670e-14	154.864779874214	25.1609737512074
1.97412512781933e-05	3.89717002028768e-10	6.95473641621525e-15	6.95473641621524e-15	163.449844881075	19.7983968827542
8.12070164574254e-05	6.59457952191656e-09	6.06927390819953e-13	6.06927390819953e-13	158.496000000000	15.9990582052137

Ben Stonebraker
October 20, 2019
Image Processing – Project 2B

Appendix E:

```
function [hay_stats,not_hay_stats] = detectionHayBalesSimple(file)

close all
IMG_0034 = imread(file);
%take the fft
IMG = fft2(IMG_0034);
e_IMG = edge( IMG_0034, 'canny' );

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%FILTER IMAGE
% filtering of FT across rows.
FLT_IMG = BW_Filtering( IMG, 2, 1/69 );
% filtering of FT across columns.
FLT_IMG = BW_Filtering( FLT_IMG, 1, 1/55 );
FLT_IMG = HP_Filtering( FLT_IMG, 2, 1/690 );
FLT_IMG = HP_Filtering( FLT_IMG, 1, 1/550 );
flt_img = real( ifft2( FLT_IMG ) );
figure(1);
imagesc( flt_img ), colormap('gray');

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%SET THRESHOLDS
%0.7 * max + 0.3 * min
thresholdMax = max(max(flt_img));
thresholdMin = min(min(flt_img));
t_img = zeros(size(flt_img));
t_img(flt_img>(thresholdMax*0.7+thresholdMin*0.3)) = 1;
figure(2);
imagesc( t_img ), colormap('gray');

%%

% create connnect or Labeled image.
[L_img,Labels] = SizeOfRegions( t_img, 1500, 2500 );
figure(3);
imagesc( L_img ), colormap('gray');

% prepare to work through connected sets.
[N,M] = size( IMG_0034 );
D_img = zeros( N, M );
m = 5; % counts figures.
hayCnt = 1;
notHayCnt = 1;
% loop through each labelled set.
for k = 1:Labels
    [Row, Col] = find( L_img == k );  % find list of pixels labeled k
    % if size of region is appropriate.
    if length( Row ) > 1500 && length( Row ) < 2500
        % compute center of blob.
        Center = round( sum( [Row Col] )/length(Row) );
        % extract image around blob.
        r = max(1,Center(1)-70):min(N,Center(1)+70);
        c = max(1,Center(2)-70):min(M,Center(2)+70);
```

```
        Local = IMG_0034( r, c );
        Mask = t_img( r, c );
        % display local image.
        figure(m)
        subplot( 221 ), imagesc( Local ), colormap('gray');
        % create threshold from top and bottom band of pixels.
        I = 1:5*141;
        Mu_O = mean( double( [Local(I) Local(end-I+1)] ) );
        Std_O = std( double( [Local(I) Local(end-I+1)] ) );

        % Create threshold.
        Thres = Mu_O + 2.5*Std_O;
        figure(100); % figure to display process.
        Local = GrowAroundBlobs( Mask, edge(Local,'canny',[],2), ...
                                 Local, Thres );
        im_stats = inv_moments(Local);
        confirm = input("Is this a haybale? (1/0)");
        if confirm == 1
            hay_stats(hayCnt) = im_stats;
            hayCnt = hayCnt + 1;
        else
            not_hay_stats(notHayCnt) = im_stats;
            notHayCnt = notHayCnt + 1;
        end
        figure(m); % move back to current figure.
        m = m + 1;  % move to next figure
        % Display segmented image.
        subplot( 222 ), imagesc( Local ), colormap('gray');
        %pause(2)
        close 100 % remove demo figure
        % save off segmented image.
        D_img( r, c ) = Local;
    else
      t_img( [Row Col] ) = 0;
    end
end

figure;
subplot(211),imagesc( D_img ), colormap('gray');
D_img( D_img > 0 ) = 1;
subplot(212),imagesc( D_img ), colormap('gray');

% %%
% e1_IMG = imdilate( e_IMG, ones(5,5) );
% e2_IMG = imerode( e1_IMG, ones(5,5) );
% %DD_img = Fill2Edge_Matlab( e2_IMG, D_img );
% %DD_img = imfill( D_img, find(  );
%
% figure;
% imagesc( DD_img ), colormap('gray');
return
```

Appendix F:

```
clear all
[hay_stats,not_hay_stats] = detect2('IMG_0038.png');
close all
%% Part 2
haymat = [];
not_haymat = [];
for i=1:1:length(hay_stats)
    imstat = hay_stats(i);
    haymat(i,:) =  [imstat.Phi1, imstat.Phi2, imstat.Phi3, imstat.Phi4,
imstat.mean, imstat.std;];
end
for i=1:1:length(not_hay_stats)
    imstat = not_hay_stats(i);
    not_haymat(i,:) =  [imstat.Phi1, imstat.Phi2, imstat.Phi3, imstat.Phi4,
imstat.mean, imstat.std;];
end
Crossplot(haymat,not_haymat);
hay_cov = cov(haymat);
not_hay_cov = cov(not_haymat);
hay_mean = mean(haymat);
not_hay_mean = mean(not_haymat);
%% Mahalanobis Part
d0_0 = MahalanobisDist(haymat',hay_cov,hay_mean');
d0_1 = MahalanobisDist(haymat',not_hay_cov,not_hay_mean');
d1_1 = MahalanobisDist(not_haymat',not_hay_cov,not_hay_mean');
d1_0 = MahalanobisDist(not_haymat',hay_cov, hay_mean');
figure(2)
set(2, 'units', 'normalized','outerposition', [0 0 1 1] );
plot( d0_0, d0_1, 'b*', d1_0, d1_1, 'rd' );
hold on
%plot( [0 10], [0 10], 'g' );
grid on
title( 'Mahalanobis Distance To Two Distributions');
set(findall(2,'type','text'),'fontSize',14);
legend( 'Points from Hay Bales', 'Points from Not Hay Bales' );
xlabel( 'Distance from Distribution 1' );
ylabel( 'Distance from Distribution 2' );
axis('equal')
ylim([0 10]);
xlim([0 10]);
```

Ben Stonebraker

Appendix G:

Hay Bales

Phi1	Phi2	Phi3	Phi4	Mean	Std. Dev.
4.29114725126337e-06	1.84139447320252e-11	3.83154103513258e-18	3.83154103513258e-18	216.739525691700	14.8616573838209
4.45067472004874e-06	1.98085054636809e-11	7.00596321843783e-19	7.00596321843783e-19	236.138199513382	7.00496246891768
4.02645794609404e-06	1.62123635916639e-11	4.11649112843612e-19	4.11649112843616e-19	237.899463806971	3.40739809891007
3.03779684258996e-06	9.22820965684954e-12	4.44950920459264e-19	4.44950920459264e-19	229.950236220472	10.3699091080648
3.61490423262779e-06	1.30675326110703e-11	3.64179496145301e-19	3.64179496145304e-19	236.377425742574	4.48669954068184

Not Hay Bales:

Phi1	Phi2	Phi3	Phi4	Mean	Std. Dev.
8.61802614019764e-06	7.37828938201223e-11	1.12567364994339e-16	1.13997333684380e-16	195.836921069798	42.8085881054173
9.22981684399019e-06	8.51895189736050e-11	6.16948644831575e-17	6.16948644831573e-17	209.746019108280	19.5044990232470
1.20784103428735e-05	1.45887996410835e-10	1.54970581925730e-15	1.54970581925730e-15	164.047163798598	29.8676983848859
1.29620314385703e-05	1.68014259014484e-10	2.33265687786965e-16	2.33265687786965e-16	204.808102345416	1.95560612798897
6.30309656496154e-06	3.97290263072299e-11	1.32026233767484e-17	1.32026233767484e-17	215.580126364159	10.8691822163854
6.24444652292416e-06	3.89931123776597e-11	2.53594573234486e-18	2.53594573234487e-18	233.888144675151	3.33640253524337
5.79301044702939e-06	3.35589700393917e-11	3.48402332528486e-18	3.48402332528485e-18	228.713012477718	26.1047053064387
6.09858544854504e-06	3.71927444732053e-11	1.18173095836678e-18	1.18173095836678e-18	239.246406570842	0.977589162978613
9.68562062123156e-06	9.38112468184260e-11	5.74301952532782e-16	5.74301952532782e-16	171.130144605117	17.1927361388178
5.33608795870039e-06	2.82980688924601e-11	6.52005643457501e-18	6.57322119763375e-18	218.229729729730	16.4689896002976
3.27927646120105e-06	1.06962161064251e-11	1.85267767018974e-17	1.86720594408310e-17	174.744218224946	14.1013042310568
9.80305708809090e-06	9.60999282723692e-11	5.20670131656623e-18	5.20670131656627e-18	238.839912280702	2.16108359843158

7.70058494165243e-06	5.87612420792751e-11	1.93732024373107e-17	1.96605150119540e-17	218.215785662563	14.3559737571627
8.35582000981648e-06	6.98197280364495e-11	1.07603169402057e-16	1.07603169402058e-16	195.434918648310	41.9565384124311
8.71725823318989e-06	7.43503486894763e-11	2.32564197520525e-16	2.35362297270458e-16	183.215617715618	36.5583380704778
7.15804434687281e-06	5.12375988717977e-11	1.14459689798025e-16	1.14459689798025e-16	185.525120772947	36.1969679663423
4.21813383263721e-06	1.77926530300387e-11	1.84477299410956e-17	1.84477299410956e-17	190.119581464873	14.5124324631339
5.16987869496484e-06	2.67276457206513e-11	1.01947123725564e-16	1.01947123725564e-16	167.836664762993	17.5731268658922
1.31164685932557e-05	1.72041748357864e-10	4.27325474104747e-16	4.27325474104747e-16	194.956989247312	38.0291898848031
5.85862854813462e-06	3.43235284650180e-11	9.89968242081781e-19	9.89968242081782e-19	239.627968337731	0.659119395765833
1.20636167735659e-05	1.44771854238975e-10	4.11438708221061e-17	4.14144861878540e-17	225.707376058041	3.11234070375175
1.63922626860233e-05	2.68706275967592e-10	9.78473759304650e-15	9.78473759304650e-15	143.446428571429	23.7973417631657
1.74900801239094e-05	3.05902902740769e-10	2.99984745026002e-15	2.99984745026002e-15	173.650465356774	8.87536528805827
1.34648705295497e-05	1.81302738377536e-10	4.78171997444663e-15	4.78171997444664e-15	146.284180790960	12.0420888292002
1.63418382584982e-05	2.67055677666915e-10	6.89989145157584e-15	6.89989145157585e-15	151.095098756401	18.6062230250377
1.66607247364966e-05	2.77579748745309e-10	2.96557850081447e-15	2.96557850081447e-15	170.824594852240	34.4337471695191
1.54739186662008e-05	2.38806577753547e-10	1.91916209796751e-15	1.93100061372075e-15	174.968283582090	30.8707271415825
1.93448258292064e-05	3.74222286362331e-10	9.93586945957125e-15	9.93586945957126e-15	154.252707581227	37.8766986629705
2.25110824672805e-05	5.06748833848705e-10	2.90524366331154e-14	2.90524366331154e-14	140.358260869565	23.8097769834477
1.94979775500625e-05	3.71294247484560e-10	4.66425884889040e-14	4.69601979204996e-14	120.389109472490	30.5128977669652
2.35872104174248e-05	5.56356495275873e-10	4.09413892000302e-14	4.09413892000302e-14	135.768115942029	24.6700224245993
6.68127407400640e-06	4.46394232519901e-11	6.13765013606076e-16	6.13765013606076e-16	145.171452236334	18.7711440435877
3.84372585421678e-06	1.47957980515476e-11	1.01702845838331e-18	1.01702845838330e-18	228.742699289661	13.3841513469724
2.09539889309015e-05	4.39069652116341e-10	1.44029418968417e-14	1.44029418968417e-14	151.316086547507	15.2346923858157
6.36654214915228e-06	4.05328589369325e-11	1.13212646580347e-16	1.13212646580347e-16	178.718500797448	19.4272523494811

3.62376566623037e-06	1.31316776037501e-11	1.57655415280832e-18	1.57655415280833e-18	221.714634998254	23.0064847965771
1.13385531304516e-05	1.28562787092074e-10	7.34169281933506e-16	7.34169281933507e-16	175.883081155433	17.9468152829413