Adaptive Signal Processing Laboratory (ASPL)
Electrical and Computer Engineering Department
University of Florida

Estimating Failure Modes Using a Multiple-Model Kalman Filter

ASPL Report No. Rep_2004-03-001

Vikas Aggarwal, Karthik Nagarajan, K. Clint Slatton

March 08, 2004

Point of Contact:
Prof. K. Clint Slatton
University of Florida; PO Box 116130; Gainesville, FL 32611
Tel: 352.392.0634, Fax: 352.392.0044, E-mail: slatton@ece.ufl.edu

Estimating Failure Modes Using a Multiple-Model Kalman Filter

Introduction

A Mixture of experts' framework is employed for identifying laser failure modes. Each expert is a Kalman filter which has a specific failure model embedded in it. The filter bank also contains one filter which has the nominal (anomaly free) model embedded within it. The filter residuals are post processed to produce a probabilistic interpretation of the operation of the laser beam. The measurements for the filters are derived from a simulated Shack Hartman plate, which consists of a 2D array of sensors. The mean square error and the probability distribution of the residuals are used as the performance indices.

Caveat: the laser beam pattern and anomalies simulated for this work may not be representative of actual laser phenomena. There are simply used as a vehicle to develop the Kalman based tracking and state estimation.

Shack Hartman plate

Figure.1 below shows the co-ordinate system used for the simulation. A laser beam cross section is given by a slice along the x-y plane at any particular time instant t. The Shack Hartman plate is placed parallel to the x-y plane and measures the instantaneous amplitude and phase of the beam. In this work, only the amplitude is used as the measurement information.

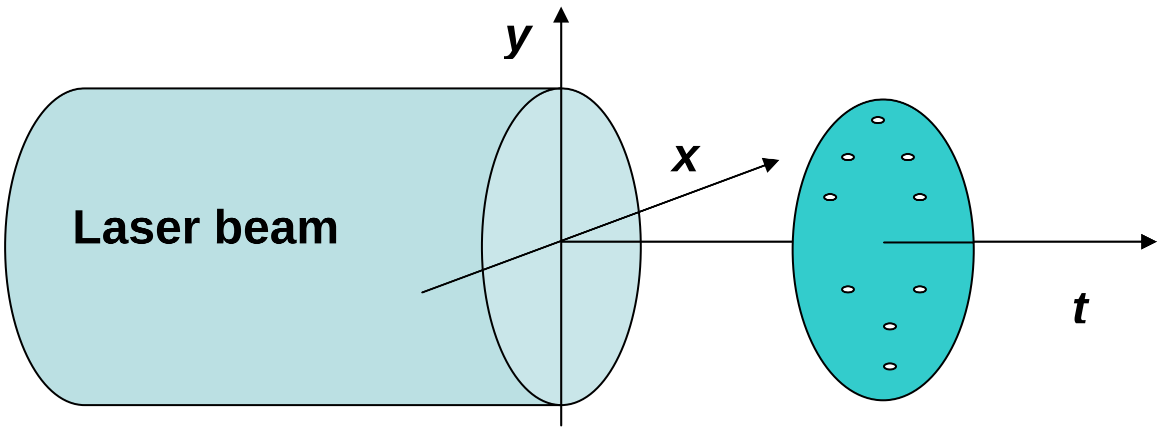
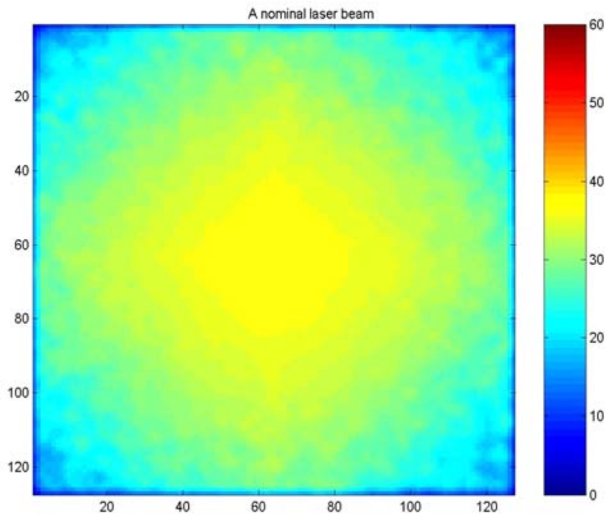


Figure.1 Co-ordinate system used for simulating the laser beam and the SHP

The way the laser beam and Shack Hartman plate are modeled in our simulation is given below (Figure.2). The SHP has been simulated as a circular array of equally spaced sensors. The beam has been modeled to have high intensity at the center that fades away in radial direction. The cross-section of the beam has 127 by 127 pixel points, where the amplitude at each of these points varies in a sinusoidal manner with time. The laser is simulated to have small random fluctuations in the laser amplitude, which is apparent in Figure 2.



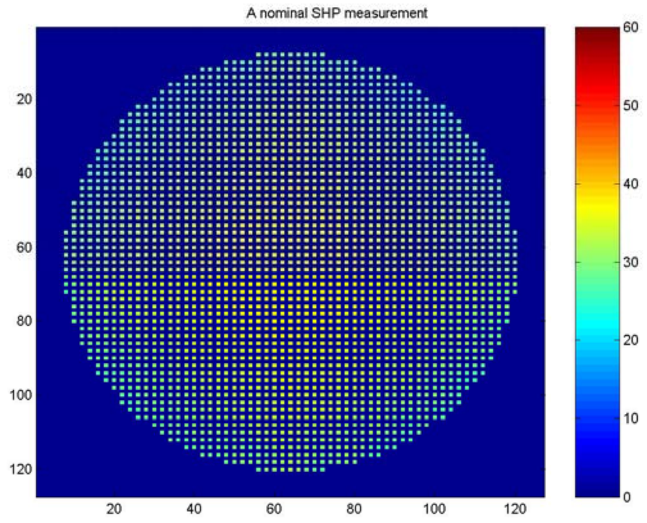
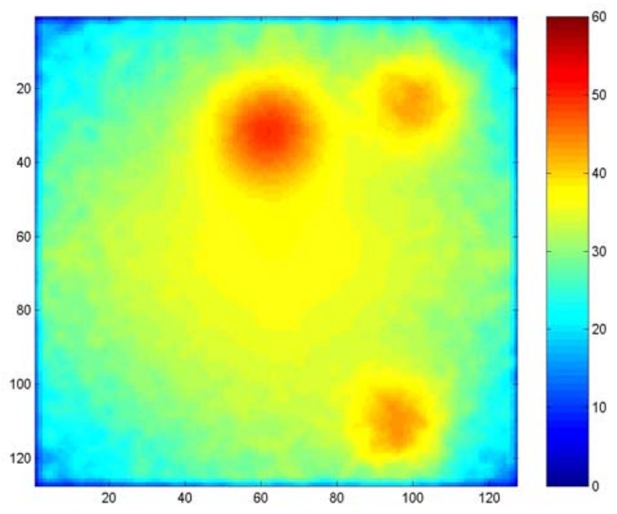


Figure.2 Cross sectional view of the laser beam and the Shack Hartman plate

Model of the occlusion

The occlusions have been modeled as Gaussian windows, where each pixel within the window varies as a Gauss-Markov process. The failure mode is characterized by a sudden increase in the process variance from that of the nominal beam behavior. The occlusions start at a random location and instant of time and are then tracked using the Mixture of Expert network. The figure below on the right shows the effect of failure on one particular pixel of the laser beam, whereas the left figure depicts the cross-sectional view of the beam after the occlusions have developed.



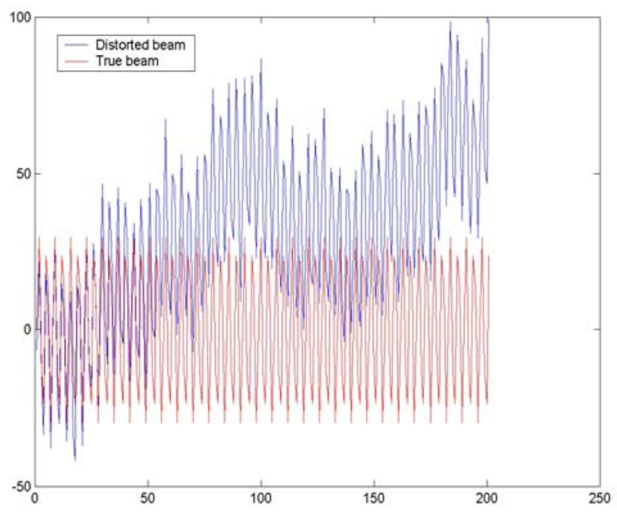


Figure.3 Model of the laser beam occlusion

Fault Detection Architecture

The architecture for FDI is depicted in figure 4 below. The architecture has been previously applied in various applications like modeling robot sensor failures as in [1]. The system comprises of a bank of Kalman filters, each of which has its own set of parameters. Kalman filtering is a well known technique for state and parameter estimation. It is a recursive estimation procedure using sequential measurement data sets. Prior knowledge of the state is improved at each step by taking the prior state estimates and new data for subsequent state estimation. Each filter produces a residual at every time step (also known as the innovation) that is fed to the fault detection and identification module. This module uses the residual to calculate the probability of the different modes of operation and give out a warning signal to indicate the presence of a particular failure mode. The module ideally remains active and starts tracking the occlusion as it begins. The module can also be designed to become active after some threshold measurements sense the beginning of occlusion. The simulation makes use of just one failure mode and hence just two Kalman filters, though multiple failure modes can be included.

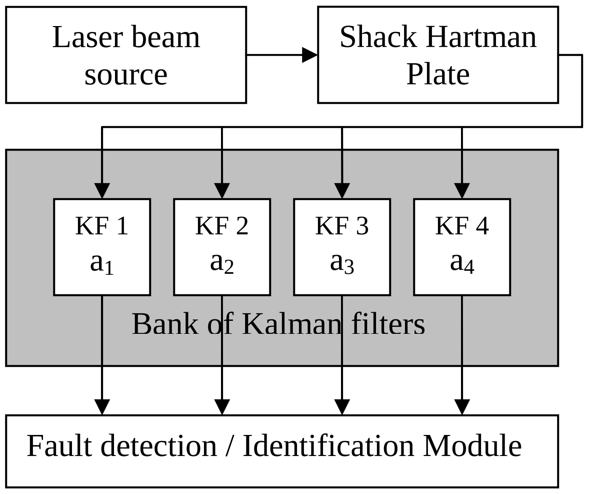


Figure.4 Fault detection Architecture

The residuals r_k are used to calculate the Gaussian conditional probability f of the measured observations at time i^{th} time step $z(t_i)$, given a particular type of failure has occurred using the approach given in [3].

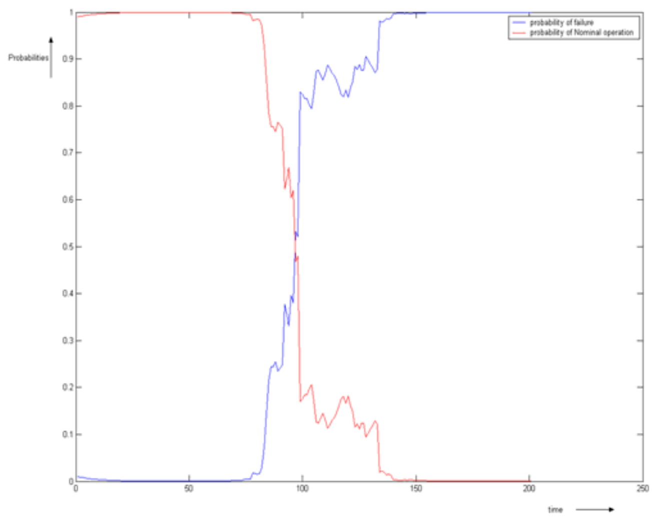
$$f\left(z(t_i)\middle|a_k, Z_{i-1}\right) = \beta \exp\left[\frac{-1}{2}\left(r_k^T(t_i) S_k^{-1} r_k(t_i)\right)\right]$$

where a_k represents the Kalman filter parameters for the k^{th} filter, i.e. the k^{th} failure mode. Z_{i-1} is the set of $m \times 1$observation vectors up through time t_{i-1}. The coefficient β is given by

$$\beta = \frac{1}{(2\pi)^{m/2} |S_k|^{1/2}}$$

where S_k is the covariance and m is the dimension of the observation vector. Once these conditional probabilities f are determined, the probability of each failure mode p_k is calculated using Bayes' rule as for every time step:

$$p_k(t_i) = \frac{f(z(t_i) \mid a_k, Z_{i-1})\, p_k(t_{i-1})}{\sum_{j=1}^{2} f(z(t_i) \mid a_k, Z_{i-1})\, p_j(t_{i-1})}$$



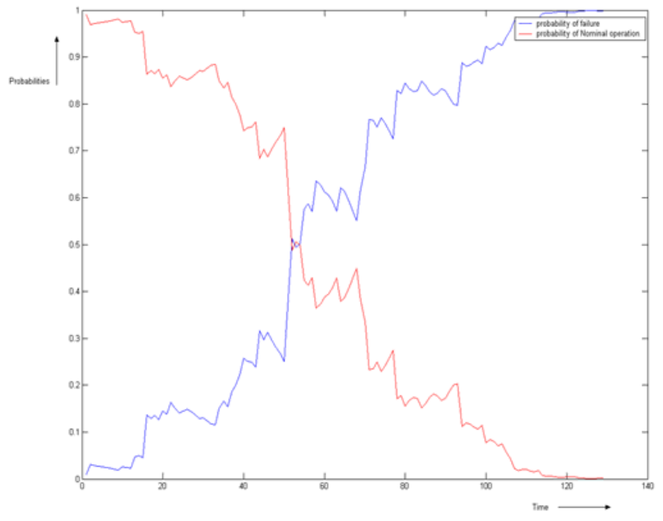


Figure.5 Probability plots. The first anomaly initiated at time step 50. The plot on the left runs for the entire simulation, whereas the plot of the right starts a few time samples after time sample 50.

The left plot shows the probability curves in the scenario when the Kalman filters are running even before the start of the occlusion while the one in the right has the filter running only after the start of the occlusion. The probabilities begin to react to the occlusion very soon after it begins. The complete transition of p_2 from values near zero to values near one depends on the particular variances used in the simulation and in the filters. For many trials the transition occurred more rapidly than in Figure 5.

Estimation of the laser beam

The estimate of the laser beam can also be obtained from the same bank of filters using a learning network. The output of each of the filter is weighted by a mixture of experts network according to the algorithm given in [2]. The same has been provided below for reference.

$$g_i = \frac{e^{u_i}}{\sum_{j=1}^{L} e^{u_j}}$$

$$u_i = z^T a_i$$

$$W_k = H_k P_k^{-1} H_k^T + R_k$$

$$r_k = z_k - H_k \hat{x}_k^{-1}$$

$$f(z_k | \alpha_i) = \frac{1}{\sqrt{2\pi |W_k|}} e^{-\frac{1}{2} r_k^T W_k^{-1} r_k}$$

$$h_i = \frac{f(z_k | \alpha_i) g_i}{\sum_{j=1}^{L} f(z_k | \alpha_j) g_j}$$

$$a_i \leftarrow a_i + \eta (h_i - g_i) z_k$$

$$g_i \leftarrow Weight$$

$$z_k \leftarrow Observation\ Set$$

$$r_k \leftarrow \mathrm{Re}\, siduals$$

Mean square error (MSE) value was used as a performance index to compare the mixture of experts (MOE) estimate to that generated using a single filter. The change in true variance of the beam from nominal condition to an occlusion condition was a factor of 100 ($Q = 0.1$ for nominal and $Q = 10$ for the simulated occlusion). For the Kalman filters tracking the beam, the process noise variance Q parameters were set at 0.1 (tuned to the nominal beam) and 25 (to track an occlusion). Notice that the second Kalman filter was not perfectly tuned to match this particular anomaly. The results are tabulated below. The MOE estimator consistently produces a smaller MSE than either filter alone.

	MSE of MoE	MSE of filter 1	MSE of filter2
Trial 1	4.9612	28.4174	5.5355
Trial 2	4.3716	16.5596	6.2167
Trial 3	4.3338	16.6778	6.1328

Appendix A

Matlab code for laser SHP simulator.

Laserfinal.m: Main File

```
% V.Aggarwal & K. Nagarajan
% 14th Jan 2004
% laserfinal.m : The main file where the laser beam and occlusions are modelled and
% estimated using a mixture of eperts network
% Other files used:
% ex3.m : File that calls the mixture of experts module to get set of weights
% gen_weight10.m :  Contains the mixture of experts module

clear all;
close all;
% Laser basics:
% -------------

c=3e8; % [m/s] vacuum speed of light
lmd=700e-9; % [m] laser wavelength
f=c/lmd;
pdftype=1; % 1=PDF from time=1;2=PDF after occlusion has been detected by threshold comparison
tstart=0;, tstep=1e-8;, tstop=0.2e-5;
t=[tstart:tstep:tstop];
tim=[1:1:length(t)];

s=sin(2*pi*f*t); % The laser modelled as a sine wave

% define SHP array
% -----------------

AR=1; % aperture radius
RadCO=0.90;
ds= AR/63; % node spacing
xA=[-AR:ds:AR];
yA=[-AR:ds:AR];

% wavexy() a 3D matrix stores the nominal laser beam with time as the third dimension
% Any 2D slice of wavexy() represents the crossection of the beam in YX plane
% Here it is a 128 x 128 x 1000 with the sine wave passing through each point
% through time. yx() depicts the points on the SHP with
% sensors. Here we consider a circular array with equally spaced sensors.
for y=1:length(yA)
   for x=1:length(xA)
       waveyx(y,x,:)=s;
       if sqrt(xA(x)^2+yA(y)^2)<AR*RadCO &  (mod(x,2)==0)
           yx(y,x)=1;
       else
           yx(y,x)=NaN;
       end
   end
end

% Now that we have convenient 2D slices (below)distortions that are
% functions of XY  can be applied in a straight forward manner.
% DF distortion function is again a 3D matrix that stores the distortion
% which can be applied to laser beam by multiplying their cross sections
PE=log(t); % phase error that grows in time as log(t)
sE=log(length(t)).*sin(2*pi*f*t + PE); % apply DF as a phase distortion
for y=1:length(yA)
   for x=1:length(xA)
```

```
       DF(y,x,:)=sE;
   end
end

% Just scaling the original beam to raise it to the level of the distortion that will be applied
to it later
BDF=waveyx*2.5;
t0=length(t);

%----------------------------------------------------------------
m=127;
n=127;
mf=floor(m/2);
nf=floor(n/2);
A=rand(m,n);
B=rand(m,n);
C=rand(m,n);

% Define the laser beam to have high intensity in the center that gradually
% reduces radially
for y= -mf:mf
   for x=-nf:nf
       YX(y+mf+1,x+nf+1)=(A(y+mf+1,x+nf+1)*x^2+B(y+mf+1,x+nf+1)*abs(x*y)+C(y+mf+1,x+nf+1)*y^2);
   end;
end;
filter=[1 1 1 1 1; 1 2 2 2 1; 1 2 3 2 1; 1 2 2 2 1; 1 1 1 1 1];
filter=filter./(sum(sum(filter)));

% Until this point, the intensity reduces towards the center. we need it
% the other way round, so subtract the pixel values from the maximum of YX
YX=max(max(YX))-YX;

% Filtering done to smoothen out the intensity changes followed by cropping
% to retain the image size
yxtemp=conv2(YX,filter);
crop=floor(length(filter)/2);
YX=yxtemp(crop+1:m+crop,crop+1:n+crop);
maximum=max(max(YX));
YX=(YX/maximum)*15;

% Define YX1 a 2D matrix that holds the occlusions as they grow with time.
% The occlusions have been defined as gaussian windows
% whose window size increses with time . The variation of the occlusion
% has been defined as a gauss markov process.
YX1=zeros(n,m);
vari=10;    % Variance of the occlusion window
no_occlusion=3;
a=zeros(1,no_occlusion);
b=zeros(1,no_occlusion);
find=1;     % Counter used when locating occlusions in the cross section
win=21;     % The window size of the occlusion
rnd=randn(1,t0); % The random variable which is used to model the Gauss markov occlusion process
rnd(1:100)=rnd(1:100)*sqrt(0.1);
rnd(101:t0)=rnd(101:t0)*sqrt(10);
aviobj = avifile('laser.avi','fps',1);

for c=1:t0
   winf=floor(win/2);
   gss=fspecial('gaussian',win,vari); % the gaussian occlusion window which is filtered and
normalized then
   gss=conv2(gss,filter);
   gss=gss/max(max(gss));
   % k is the number of occlusions in the laser beam
   for k = 1:3  % Find 3 random points to start occlusion
       if c==1
           i(k)= 1 + round((m-1)*rand);
           j(k)= 1 + round((m-1)*rand);
       end
       i1(k)=max(1,i(k)-winf); % find the region of overlap between the occlusion window
       j1(k)=max(1,j(k)-winf); % centered about the point just found and the beam cross section
       i2(k)=min(m,i(k)+winf);
       j2(k)=min(n,j(k)+winf);
       % Growth of the occlusion that follows a gauss-markov model after a
       % time instant 100 before that it is just a normal varying process
       % with zero mean should ideally be kept as a constant ie 0 variance
       if c>100
           YX1(i1(k):i2(k),j1(k):j2(k))=YX1(i1(k):i2(k),j1(k):j2(k)) +
(rnd(c)+0.003*c)*gss(winf+1-(i(k)-i1(k)):winf+1+(i2(k)-i(k)),winf+1-(j(k)-j1(k)):winf+1+(j2(k)-
j(k)));
       else
           YX1(i1(k):i2(k),j1(k):j2(k))=YX1(i1(k):i2(k),j1(k):j2(k)) + (rnd(c))*gss(winf+1-(i(k)-
i1(k)):winf+1+(i2(k)-i(k)),winf+1-(j(k)-j1(k)):winf+1+(j2(k)-j(k)));
       end
   end;

   % The following code snippet finds the random set of co-ordinates where the
   % occlusion spawns by comparing the change in the values between 2 time instants with its
neighbours
   if pdftype==1
       if c >= 2 & find <= no_occlusion
           for cnt1=2:126
               for cnt2=2:126
                   point1=abs(tempYX2(cnt1-1,cnt2-1)-YX1(cnt1-1,cnt2-1));
                   point2=abs(tempYX2(cnt1,cnt2-1)-YX1(cnt1,cnt2-1));
                   point3=abs(tempYX2(cnt1+1,cnt2-1)-YX1(cnt1+1,cnt2-1));
                   point4=abs(tempYX2(cnt1-1,cnt2)-YX1(cnt1-1,cnt2));
                   point5=abs(tempYX2(cnt1+1,cnt2)-YX1(cnt1+1,cnt2));
                   point6=abs(tempYX2(cnt1-1,cnt2+1)-YX1(cnt1-1,cnt2+1));
                   point7=abs(tempYX2(cnt1,cnt2+1)-YX1(cnt1,cnt2+1));
                   point8=abs(tempYX2(cnt1+1,cnt2+1)-YX1(cnt1+1,cnt2+1));
                   point0=abs(tempYX2(cnt1,cnt2)-YX1(cnt1,cnt2));
                   if point0>point1 & point0>point2 & point0>point3 & point0>point4 &
point0>point5 & point0>point6 & point0>point7 & point0>point8
                       if a(:) ~= cnt1 & b(:) ~= cnt2
                           a(find)=cnt1-2;,b(find)=cnt2-2;,find=find+1; % the -2 is done because
the occlusion window hasn't been
                           %cropped and the window size become +4 so the center is displaced by 2
                       end
                   end
               end
           end
       end
       % tempYX2 is used to compare the point values at 2 consecutive times
       tempYX2=YX1;
       % tempYX1 is the random growth in time at one of the occlusion center
       tempYX1(c)=YX1(i(1),j(1));
       % Take a slice of the nominal beam and applies the time varying occlusion to it
       % to get the 3D matrix whose slices represent wavefront at the SHP
       BDFslice(:,:,c)=(BDF(:,:,c) .* YX) + YX1;
       clims=[0 60];
       win=round(win+0.5);     % increase the wiondow with a rate of 0.5
       % Stores the laser cross section as frames in the video file laser.avi
       if(mod((c-2),7)==0)
           h = imagesc(abs(BDFslice(:,:,c)),clims);
           set(h,'EraseMode','xor');
           axis equal;
           frame = getframe(gca);
           aviobj = addframe(aviobj,frame);
       end
   % Case for starting the tracking procedure after the occlusion  has crossed a threshold level

   elseif pdftype==2
       if c >= 2
           for cnt1=2:126
               for cnt2=2:126
                   point1=abs(tempYX2(cnt1-1,cnt2-1)-YX1(cnt1-1,cnt2-1));
                   point2=abs(tempYX2(cnt1,cnt2-1)-YX1(cnt1,cnt2-1));
                   point3=abs(tempYX2(cnt1+1,cnt2-1)-YX1(cnt1+1,cnt2-1));
                   point4=abs(tempYX2(cnt1-1,cnt2)-YX1(cnt1-1,cnt2));
                   point5=abs(tempYX2(cnt1+1,cnt2)-YX1(cnt1+1,cnt2));
                   point6=abs(tempYX2(cnt1-1,cnt2+1)-YX1(cnt1-1,cnt2+1));
                   point7=abs(tempYX2(cnt1,cnt2+1)-YX1(cnt1,cnt2+1));
```

```
                    point8=abs(tempYX2(cnt1+1,cnt2+1)-YX1(cnt1+1,cnt2+1));
                    point0=abs(tempYX2(cnt1,cnt2)-YX1(cnt1,cnt2));
                    if point0>point1 & point0>point2 & point0>point3 & point0>point4 &
point0>point5 & point0>point6 & point0>point7 & point0>point8 & find <= no_occlusion &
abs(point0) > 6
                        if a(:) ~= cnt1 & b(:) ~= cnt2
                            a(find)=cnt1-2;,b(find)=cnt2-2;,find=find+1;
                            flag=c; % variable used to store the time instant from which the
occlusion cross the safety threshold
                        end
                    end
                end
            end
        end
        % tempYX2 is used to compare the point values at 2 consecutive times
        tempYX2=YX1;
        if flag~=0
            tempYX11(c-flag+1)=YX1(i(1),j(1));
        end
        BDFslice(:,:,c)=(BDF(:,:,c) .* YX) + YX1;
        clims=[0 60];
        win=round((win+0.5));
   end
end;
aviobj = close(aviobj);

% Plots the laser cross section at two points in time to visualize the
% occlusion growth
figure(1);
imagesc(abs(BDFslice(:,:,30)),clims);
title('Laser cross-section at the start of occlusion');
colorbar;

figure(2);
imagesc(abs(BDFslice(:,:,191)),clims);
title('Laser cross-section after growth of occlusion');
colorbar;

% Plots the SHP cross section after occlusion
figure(3);
imagesc(abs(BDFslice(:,:,191).*yx),clims);
title('SHP cross-section after the growth of occlusion');
colorbar;

for j1= 1:t0
    temp2(j1)=BDFslice(i(1),j(1),j1);           % the variable stores the randomly varying
sinusoidal beam
    temp1(j1)=BDF(i(1),j(1),j1).*YX(i(1),j(1)); % the variable stores the nominal sinusoidal beam
end;

% The laser sine wave with and without effect of occlusion
figure(4);
plot(temp2);hold on; plot(temp1,'r');
legend('Laser beam with occlusion','Laser beam without occlusion');
title('Laser sine wave in the presence and absence of occlusion');
xlabel('Time scale');
ylabel('Intensity of the laser beam');

% ------------------------------------------------
% Kalman filter for detection the failure mode in the laser beam


if pdftype==1
    Q1=0.1;,Q2=25;,R1=9;,R2=9;,Rnoise=3;
    % call the mixture of experts module to generate the weights
    [xnet,Mean_weight1,Mean_weight2,z,filter1,filter2,h1,h2]=ex3(t,tempYX1,Q1,Q2,R1,R2,Rnoise);
    aviobj = avifile('pdf.avi','fps',7);

    figure(11);
    for c = 1:201
        h = plot(tim(1:c),h1(1:c),'b',tim(1:c),h2(1:c),'r');
        axis([1 201 -0.2 1.2]);
        set(h,'LineWidth',2,'EraseMode','xor');
        frame = getframe(gca);
        aviobj = addframe(aviobj,frame);
    end
    aviobj = close(aviobj);

    figure(5);
    subplot(211);
    plot(tempYX1);hold on;plot(xnet,'-r');
    legend('Occlusion growth','MoE estimate of the occlusion');
    title('MoE estimate of the occlusion growth');
    xlabel('Time');
    ylabel('Intensity of the occlusion');
    subplot(212);
    plot(tempYX1);hold on;plot(z,'-r');
    legend('Occlusion growth','Observation made on the occlusion');
    title('Observation made on the occlusion growth');
    xlabel('Time');
    ylabel('Intensity of the occlusion');

    figure(6);
    subplot(211);
    plot(tempYX1);hold on;plot(xnet,'-r');
    legend('Occlusion growth','MoE estimate of the occlusion');
    title('MoE estimate of the occlusion growth');
    xlabel('Time');
    ylabel('Intensity of the occlusion');
    subplot(212);
    plot(tempYX1);hold on;plot(filter1,'-r');
    legend('Occlusion growth','Smooth filter estimate of the occlusion');
    title('Smooth filter estimate of the occlusion growth');
    xlabel('Time');
    ylabel('Intensity of the occlusion');

    figure(7);
    subplot(211);
    plot(tempYX1);hold on;plot(xnet,'-r');
    legend('Occlusion growth','MoE estimate of the occlusion');
    title('MoE estimate of the occlusion growth');
    xlabel('Time');
    ylabel('Intensity of the occlusion');
    subplot(212);
    plot(tempYX1);hold on;plot(filter2,'-r');
    legend('Occlusion growth','Rough filter estimate of the occlusion');
    title('Rough filter estimate of the occlusion growth');
    xlabel('Time');
    ylabel('Intensity of the occlusion');

    figure(8);
    plot(Mean_weight1,'b');hold on;plot(Mean_weight2,'r');
    legend('Weight for Smooth filter','Weight for Rough filter ');
    title('Weight tracks of the two filters in the bank');
    xlabel('Time');
    ylabel('Weight values assigned');

    figure(9);
    plot(h1,'b');hold on;plot(h2,'r');
    legend('PDF for Smooth filter','PDF for Rough filter');
    title('PDFs of the two filters in the bank');
    xlabel('Time');
    ylabel('Probabilities assigned');

    xhatsine= s*2.5*YX(i(1),j(1)) + xnet;
    zsine=s*2.5*YX(i(1),j(1)) + z;

    figure(10);
    subplot(211);
    plot(xhatsine,'r');hold on;plot(temp2,'b');
    legend('Laser beam estimate by MoE','Corrupted laser beam');
    title('Laser beam estimate by MoE and the original');
    xlabel('Time');
    ylabel('Intensity of laser beam');
    subplot(212);
    plot(zsine,'r');hold on;plot(temp2,'b');
    legend('Observation of Laser beam with measurement noise','Corrupted laser beam');
    title('Observation on the laser beam with measurement noise and the original corrupted
version');
    xlabel('Time');
    ylabel('Intensity of laser beam');

    MSE_for_MoE=mean((tempYX1-xnet).^2);
    MSE_for_smooth_filter=mean((tempYX1-filter1).^2);
    MSE_for_rough_filter=mean((tempYX1-filter2).^2);
    MSE_for_MoE
    MSE_for_smooth_filter
    MSE_for_rough_filter
end


% PDF after the occlusion has started
if pdftype==2
    Q1=3;,Q2=6;,R1=0.01;,R2=0.01;,Rnoise=0.1;
    tforfilter=[1:length(tempYX11)];
    % call the mixture of experts module to generate the weights

[xnet1,Mean_weight1,Mean_weight2,z,filter1,filter2,h1,h2]=ex3(tforfilter,tempYX11,Q1,Q2,R1,R2,Rno
ise);

    figure(9);
    plot(h1,'b');hold on;plot(h2,'r');
    legend('PDF for Rough filter','PDF for Smooth filter');
    title('PDFs of the two filters in the bank after the occlusion has started');
    xlabel('Time');
    ylabel('Probabilities assigned');
end;
```

ex3.m: Subroutine

```
% V. Aggarwal & K. Nagarajan
% 25th Jan 2004
% ex3.m : This file contains the module that calls the MoE module multiple times
% Files accesed : gen_weight10.m -> To get the weights generated from MoE
% Files accesed from: laserfinal.m
% The code has been used from the previous work on MofE
function [xnet,Mean_weight1,Mean_weight2,z,xhat1,xhat2,h1,h2]=ex3(k,x,Q1,Q2,R1,R2,Rnoise)

randn('state',sum(100*clock)); % 07-Oct-01
v=randn(size(k))*Rnoise;
z=x+v;

for iter=1:30
    [weight1, weight2, xhat1, xhat2, h1, h2]=gen_weight10(k,z,Q1,Q2,R1,R2);
    Accu_weight1(iter,:) = weight1;
    Accu_weight2(iter,:) = weight2;
end

Mean_weight1 = mean(Accu_weight1,1);
Mean_weight2 = mean(Accu_weight2,1);

xnet=xhat1.*Mean_weight1 + xhat2.*Mean_weight2;
```

gen_weight10.m: Subroutine

```
% V. Aggarwal & K. Nagarajan
% 25th Jan 2004
% gen_weight10.m : Contains the function to calculate the weights generated
% from MoE network
% Files accessed from: ex3.m
function [g1, g2, xhat1, xhat2, h1, h2]=gen_weight10(k,z,Q1,Q2,R1,R2);

PhiTrue=1;
% ----------------------------------------------
% Define 2 Kalman filters
% ----------------------------------------------

% Filter 1
xp=zeros(size(k));, Pp=zeros(size(k));
Phi=ones(size(k))*PhiTrue;
Q=zeros(size(k));
Q(1:length(k))=Q1;
R=zeros(size(k));
R(1:length(k))=R1;
H=ones(size(k));

for cnt=1:1:length(k)
  K(cnt)=Pp(cnt)*H(cnt)/(H(cnt)*Pp(cnt)*H(cnt)+R(cnt));
  xhat1(cnt)=xp(cnt)+K(cnt)*(z(cnt)-H(cnt)*xp(cnt));
  in(cnt)=z(cnt)-H(cnt)*xp(cnt);
  P(cnt)=(1-K(cnt)*H(cnt))*Pp(cnt);
  if cnt<length(k)
    Pp(cnt+1)=Phi(cnt)*P(cnt)*Phi(cnt)+Q(cnt);
    xp(cnt+1)=Phi(cnt)*xhat1(cnt);
  end
end

% Filter 2
xpna=zeros(size(k));, Ppna=zeros(size(k));
Qna=zeros(size(k));
Qna(1:length(k))=Q2;
Rna=zeros(size(k));
Rna(1:length(k))=R2;

for cnt=1:1:length(k)
  Kna(cnt)=Ppna(cnt)*H(cnt)/(H(cnt)*Ppna(cnt)*H(cnt)+Rna(cnt));
  xhat2(cnt)=xpna(cnt)+Kna(cnt)*(z(cnt)-H(cnt)*xpna(cnt));
  inna(cnt)=z(cnt)-H(cnt)*xpna(cnt);
  Pna(cnt)=(1-Kna(cnt)*H(cnt))*Ppna(cnt);
  if cnt<length(k)
    Ppna(cnt+1)=Phi(cnt)*Pna(cnt)*Phi(cnt)+Qna(cnt);
    xpna(cnt+1)=Phi(cnt)*xhat2(cnt);
  end
end

% Probability generation using Magill's filter bank approach
h1(1)=.99; % Initial probabilities, higher for the 1st filter to indicate the nominal operation
h2(1)=.01;
for id =2:1:length(k)

    W(id)=H(id)*Pp(id)*H(id)+ R(id);
    f1(id) = normpdf(z(id),H(id)*xp(id), sqrt(W(id)));

    Wna(id)=H(id)*Ppna(id)*H(id)+Rna(id);
    f2(id) = normpdf(z(id),H(id)*xpna(id), sqrt(Wna(id)));

    f(id) = f1(id)*h1(id-1) + f2(id)*h2(id-1) ;
    h1(id) = (f1(id)*h1(id-1))/f(id);
    h2(id) = (f2(id)*h2(id-1))/f(id);
end

% Weight generation using the MofE concept
```

```
a1(1)=0;
a2(1)=0;
myu = 0.01;
for id =1:1:length(k)
    u1(id) = z(id)*a1(id);,u2(id) = z(id)*a2(id);
    g1(id) =exp(u1(id))/(exp(u1(id))+exp(u2(id)));
    g2(id) =exp(u2(id))/(exp(u1(id))+exp(u2(id)));

    W(id)=H(id)*Pp(id)*H(id)'+ R(id);
    f1(id) = normpdf(z(id),H(id)*xp(id), sqrt(W(id)));

    Wna(id)=H(id)*Ppna(id)*H(id)'+Rna(id);
    f2(id) = normpdf(z(id),H(id)*xpna(id), sqrt(Wna(id)));

    f(id) = f1(id)*g1(id) + f2(id)*g2(id);
    h1temp1(id) = (f1(id)*g1(id))/f(id);
    h2temp2(id) = (f2(id)*g2(id))/f(id);
    a1(id+1) = a1(id) + myu*(h1temp1(id)-g1(id))*z(id);
    a2(id+1) = a2(id) + myu *(h2temp2(id)-g2(id))*z(id);
end
```

References

[1] Roumeliotis, S.I.; Sukhatme, G.S.; Beke, "Sensor fault detection and identification in a mobile robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Volume: 3 , 13-17 Oct. 1998, Pages:1383 - 1388 vol.3.

[2] Chaer, W.S.; Bishop, R.H.; Ghosh, J**., "**A mixture-of-experts framework for adaptive Kalman filtering," *IEEE Transactions on Systems, Man and Cybernetics*, Volume: 27 , Issue: 3 , June 1997, Pages:883 – 896.

[3] Maybeck, P.S.; Hanlon, P.D., "Performance enhancement of a multiple model adaptive estimator," *Aerospace and Electronic Systems, IEEE Transactions on* , Volume: 31 , Issue: 4 , Oct. 1995, Pages:1240 – 1254.

[4] Menke, T.E.; Maybeck, P.S., "Sensor/actuator failure detection in the Vista F-16 by multiple model adaptive estimation," *Aerospace and Electronic Systems, IEEE Transactions on* , Volume: 31 , Issue: 4 , Oct. 1995, Pages:1218 – 1229.