

## Contents

---

- =====%%
- ===== Initilisations =====%%
- ===== User Inputs and GUI =====%%
- ===== Path Additions =====%%
- ===== Run SBR =====%%
- ===== Read Crop Data =====%%
- ===== Train SVM =====%%
- ===== Body =====%%
- ===== Preprocessing =====%%
- ===== Set Snake At Centre =====%%
- ===== Call Snake =====%%
- ===== Feedback =====%%
- ===== Calculate SLR =====%%

=====%%

---

Author : Deepak R, Abhishek Vasant Nayak Date : 1 September 2015 Institute : M S Ramaiah Institute Of Technology  
Purpose : Ear Detection using Snakes (Active Contour Model)

```
%%=====%%  
close all;  
clear all;  
global i j length perFolder hit counter;
```

===== Initilisations =====%%

---

```
i = 1;  
j = 1;  
hit = 0; % Counter to count successful ear localization rate  
counter = 0; % Counter to count total number of images tested  
FileReadPath = 'Database\Snakes_Pie2\s'; % File location of output from Snake Based Background  
Removal  
length = 30;  
perFolder = 13;  
file = zeros(390,4);  
lookFlag = 0;  
SBRrunFlag = 0;
```

===== User Inputs and GUI =====%%

---

```
choice = questdlg('Would you like to run Snake-based Background Removal (it will take 6 and a Half  
minuted to run SBR)?', ...  
    'Yes', ...  
    'No');
```

```

switch choice
    case 'Yes'
        SBRrunFlag = 1;
    case 'No'
        disp('Skipping SBR and using Precompiled SBR Database');

end

choice = questdlg('Would you like to look at the detected Images?', ...
    'Yes', ...
    'No');

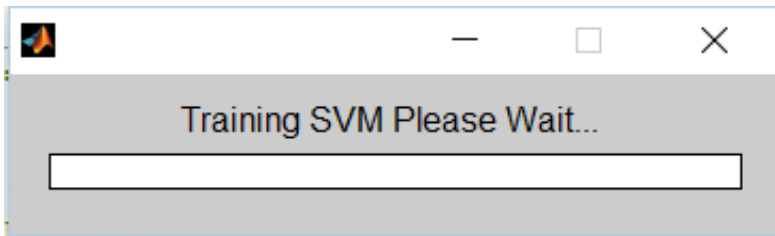
switch choice
    case 'Yes'
        lookFlag = 1;
    case 'No'
        disp('Not Showing Detected Images');

end

h = waitbar(0, 'Training SVM Please Wait...');

```

Skipping SBR and using Precompiled SBR Database  
 Not Showing Detected Images



#### ===== Path Additions =====%%

```

addpath('SnakeFunctions');           % Adds snake functions to path
addpath('SvmFunctions');             % Adds SVM functions to path
addpath('Database');                 % Adds Databases to path

```

#### ===== Run SBR =====%%

```

if(SBRrunFlag == 1)
    disp('Running SBR');
    addpath('CustomDatabaseCreator'); % Adds CMU Cropping data to path used in display
    run CreateCmpuSnake;
    FileReadPath = 'Database\Snakes_Pie\s';
end

```

#### ===== Read Crop Data =====%%

```

if(lookFlag == 1)

```

```

addpath('CMUCropData');
addpath('DisplayFunction');
temp = textread('CMUCropData.txt', '%d', 'delimiter', '\n');
for i = 1:4
    file(:,i) = (temp((390*(i-1))+1:(390*i)))';
end
end

i = 1;

```

## ===== Train SVM =====%%

```
SVM = svmTrain();
```

## ===== Body =====%%

```

while i<=length
    while j<=perFolder

```

## ===== Preprocessing =====%%

```

I = imread(strcat(FileReadPath,num2str(i), '\', num2str(j), '.jpg'));
I = im2double(I);
I = imresize(I,[size(I,1)*3,size(I,2)*3]);

```

## ===== Set Snake At Centre =====%%

```

[n,m] = size(I);
r = n/2;
c = m/2;
siz = 40;
x = [ n/2+5,n/2,n/2-5,n/2];
y = [ m/2,m/2-5,m/2,m/2+5];
P=[x(:) y(:)]; % Make an array with the coordinates

```

## ===== Call Snake =====%%

```

Options=struct;
Options.Verbose=true;
Options.Iterations=70;
[O,J]=Snake2D(I,P,Options); % Call the Snake to Detect the ear

```

## ===== Feedback =====%%

```

%First reiteration
res = svmEvaluate(SVM,imcrop(I,[min(O(:,2))-35,min(O(:,1))-80,55,100]));

```

```

% Test if the ear detected by the Snake-based Ear Localization is
% an Ear or not.

if(res == 0)
    x = [ n/2+5,n/2,n/2-5,n/2];
    y = [ m/2+60,m/2+55,m/2+60,m/2+55];
    P=[x(:) y(:)];
    Options = struct;
    Options.Verbose = true;
    Options.Iterations = 70;
    [O,J]=Snake2D(I,P,Options);
end

%Second reiteration
res = svmEvaluate(SVM,imcrop(I,[min(O(:,2))-35,min(O(:,1))-80,55,100]));

if(res == 0)
    x = [ n/2-25,n/2-30,n/2-35,n/2-30];
    y = [ m/2,m/2,m/2,m/2];
    P=[x(:) y(:)];
    Options = struct;
    Options.Verbose = true;
    Options.Iterations = 70;
    [O,J]=Snake2D(I,P,Options);
end

% Used for display purpose
if(lookFlag == 1)
    I3 = imcrop(imread(strcat('F:\Database\Pie Mixed\s',num2str(i),'\',num2str(j),'.jpg
')),[9 12 631 474]);
    imshow(I3);
    hold on;
    [posi,rcor] = getCor(file,i,j,[min(O(:,2))-35,min(O(:,1))-50,55,80]);
    % gets coordinates from rescaled version for the original
    rectangle('Position',posi,'EdgeColor','y','LineWidth',1);
    hold off;
    waitforbuttonpress;
end

```

## ===== Calculate SLR =====%%

```

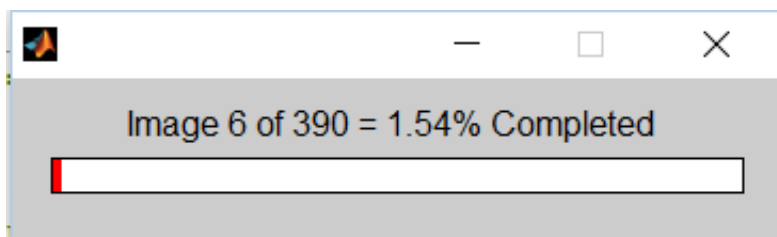
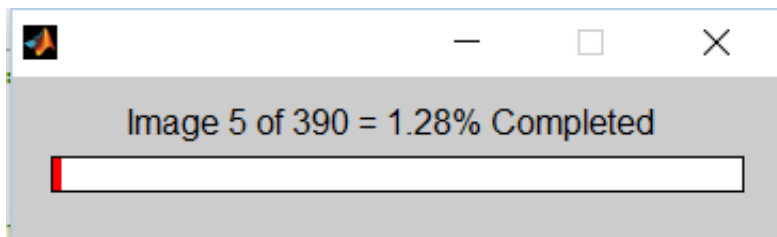
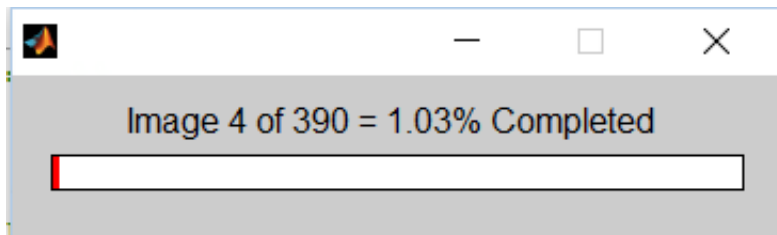
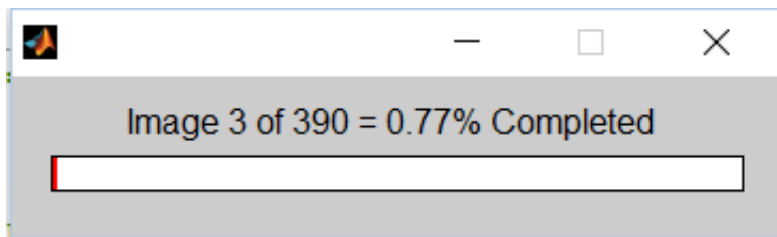
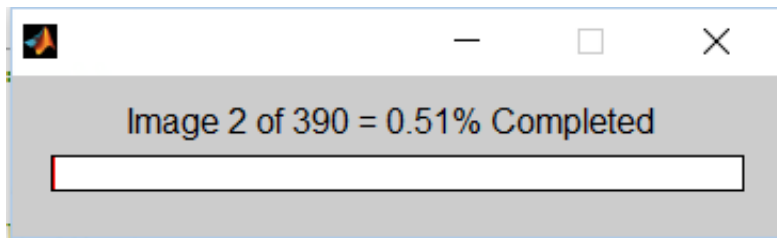
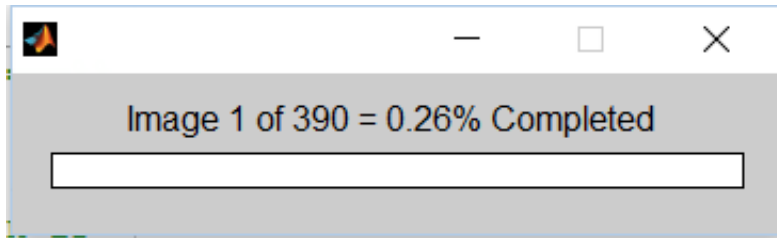
e1 = (imcrop(I,[min(O(:,2))-35,min(O(:,1))-50,55,100]));
ee = entropy(e1);

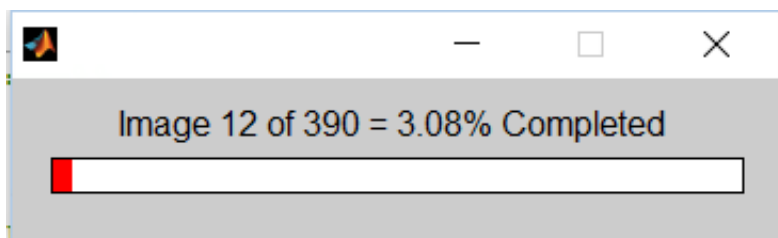
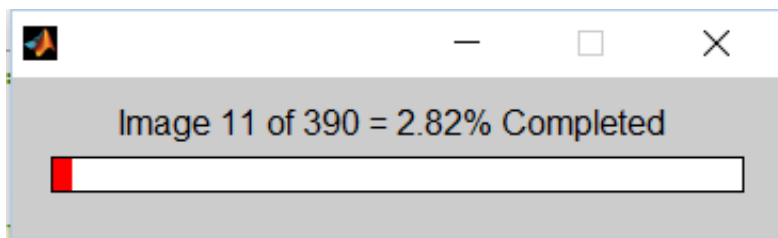
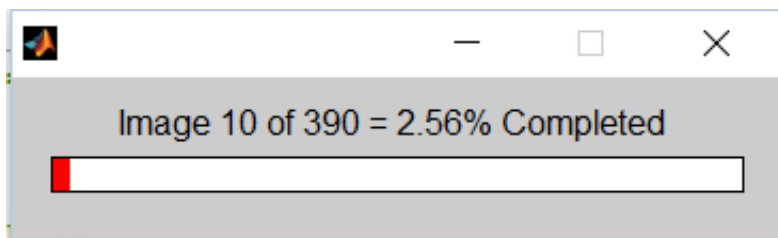
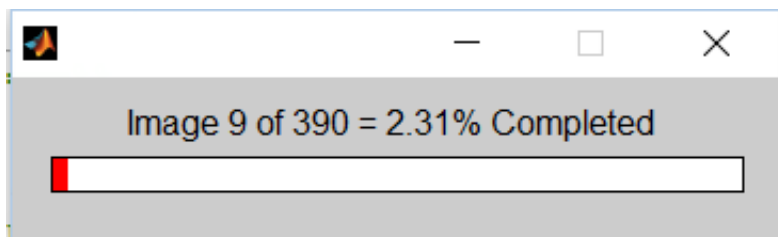
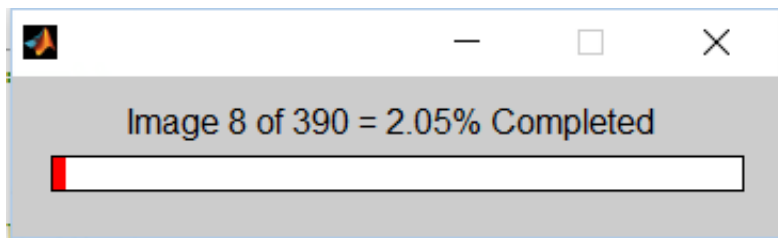
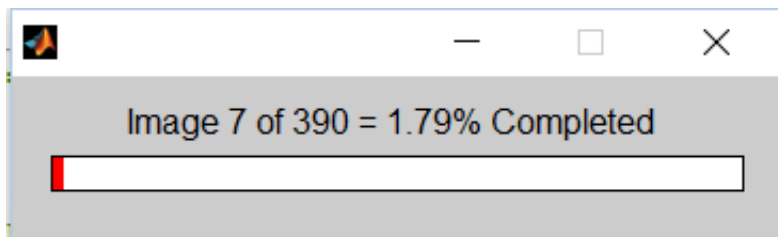
% Thresholding to find SLR Refer Madan R.G., Rahul K., Manikantan K, Ramachandran S,
% Entropy based binary swarm optimization and classification for ear detection,
% Eng.Appl. Artif. Intelli., 27, pp. 115-128, 2013. Here a different
% threshold was found and the upper limit 6.8 is because the background in
% CMU-PIE face database is falsely accepted when this upper limit is not
% set.

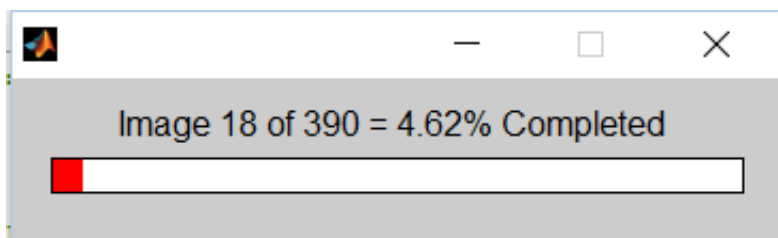
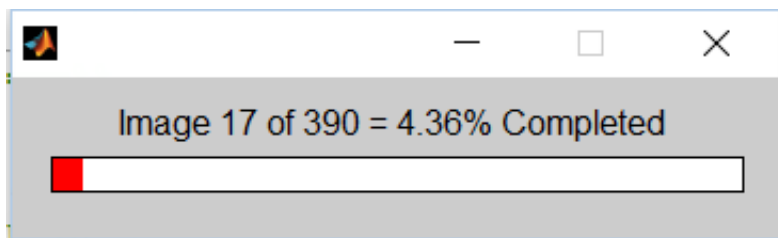
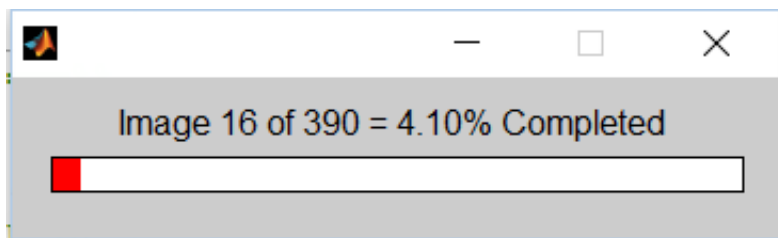
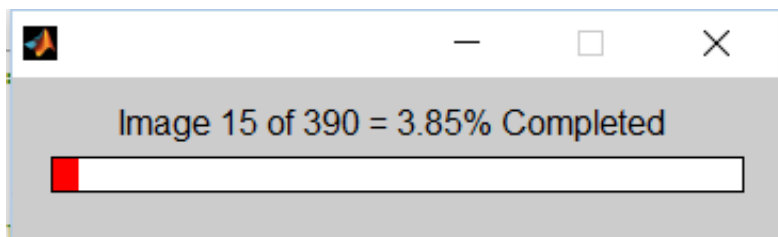
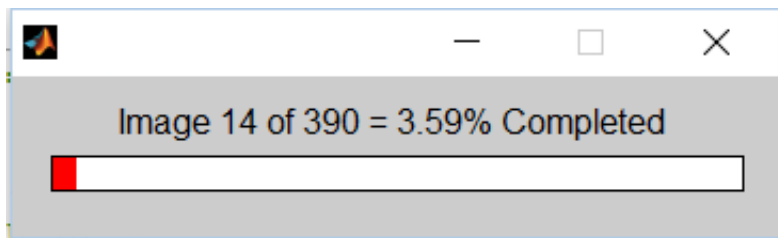
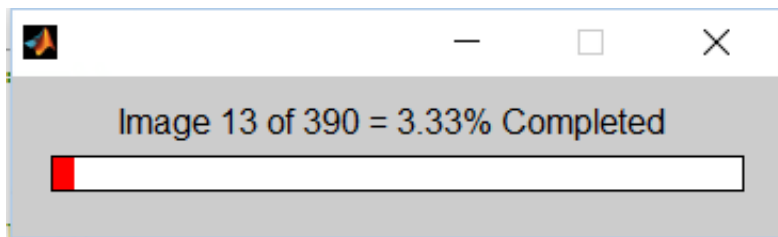
if( ee >= 4.7 && ee < 6.8)
    hit = hit+1;
end
counter = counter+1;
j = j+1;

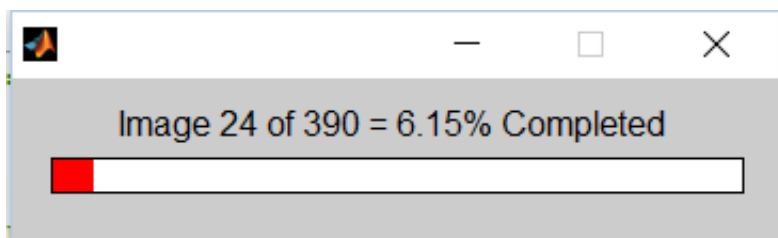
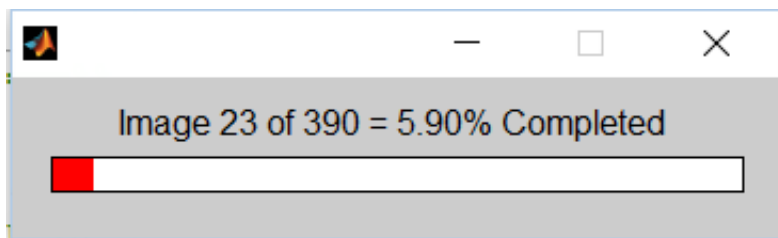
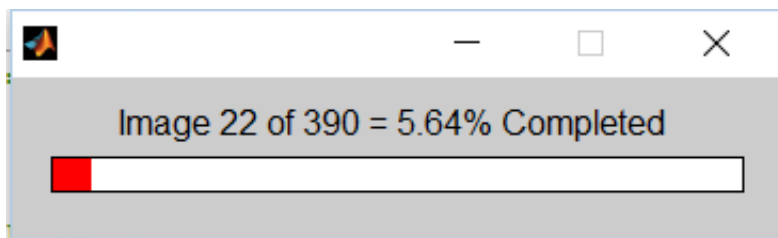
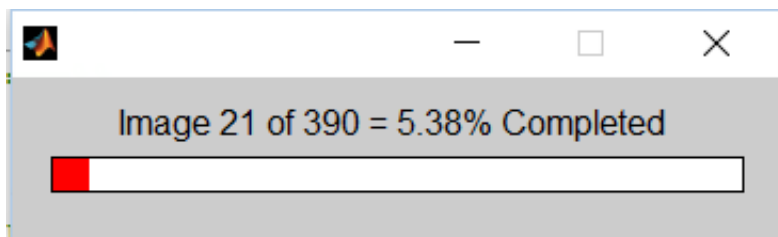
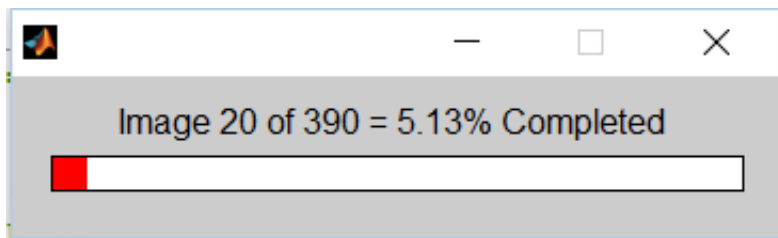
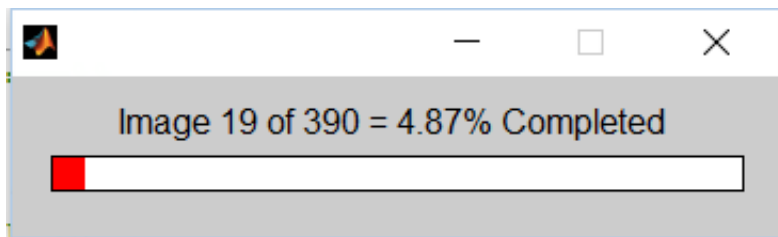
```

```
perc = ((i-1)*13+j-1)/390;  
waitbar(perc,h,sprintf('Image %d of 390 = %3.2f%% Completed',((i-1)*13+j-1),perc.*100));
```

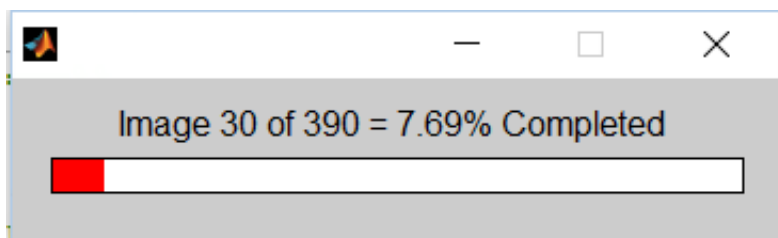
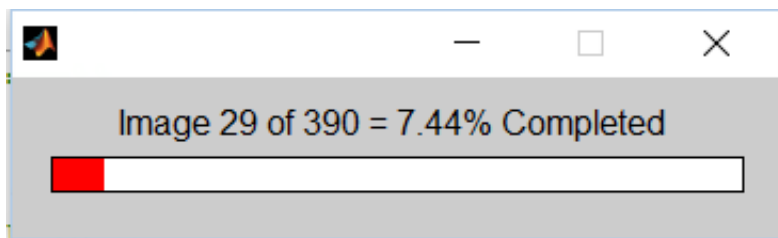
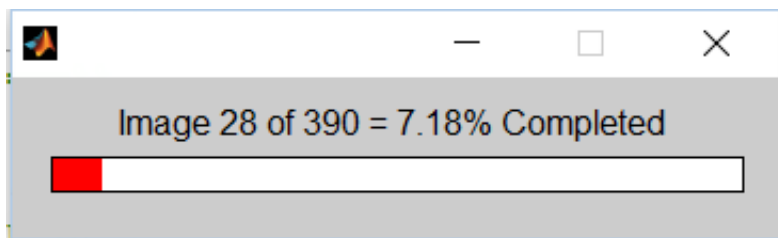
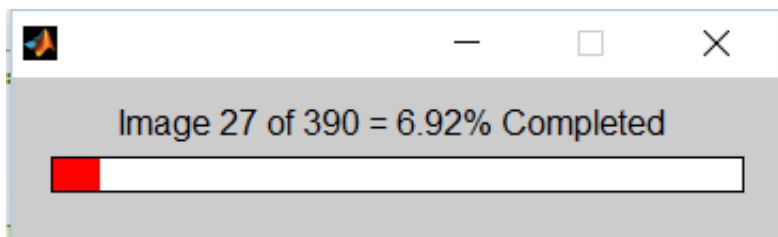
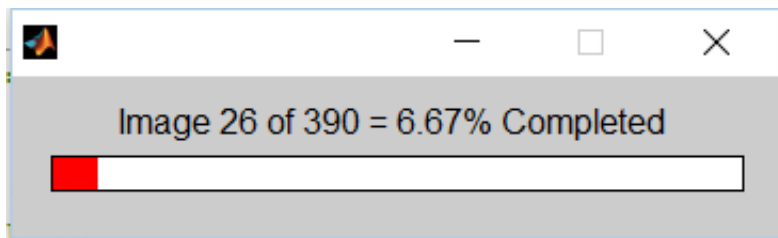
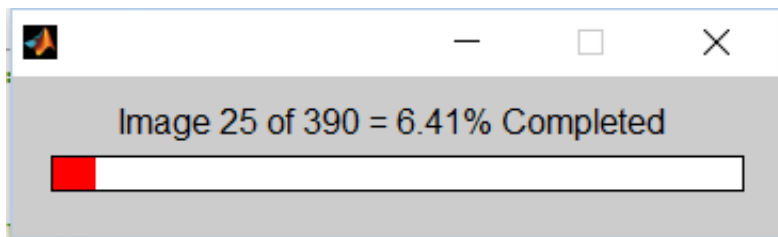


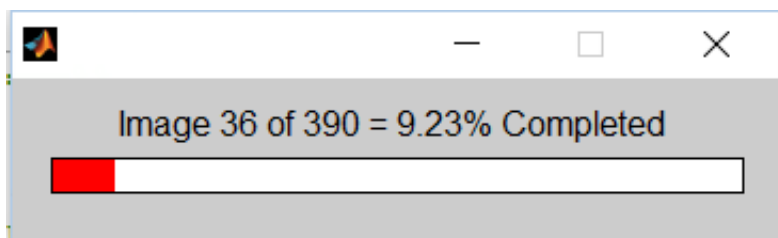
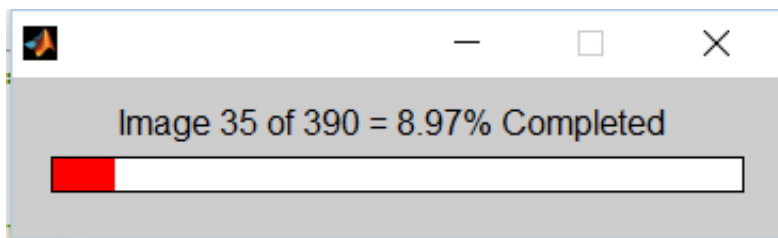
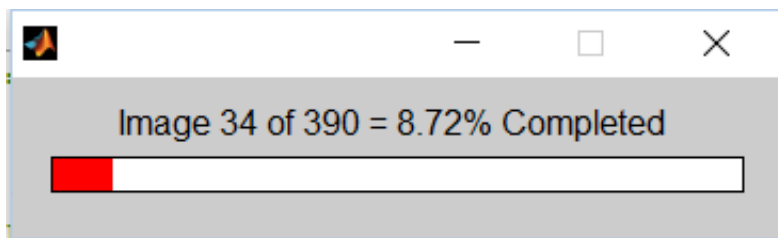
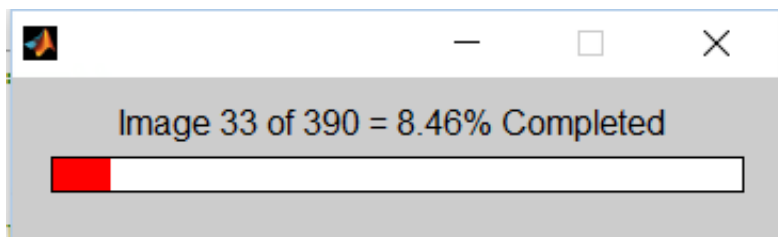
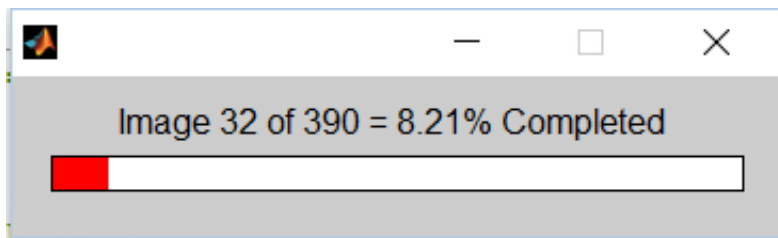
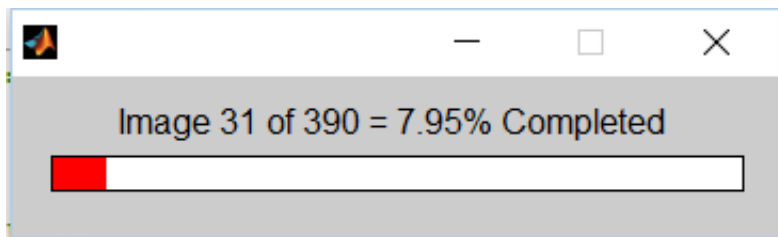


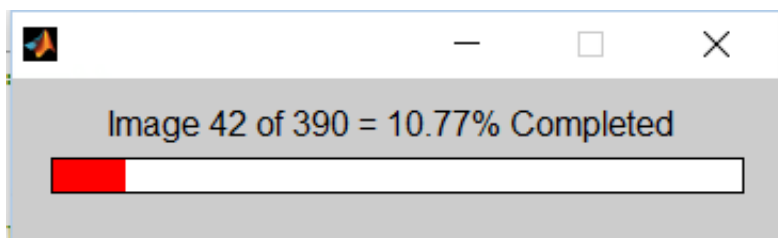
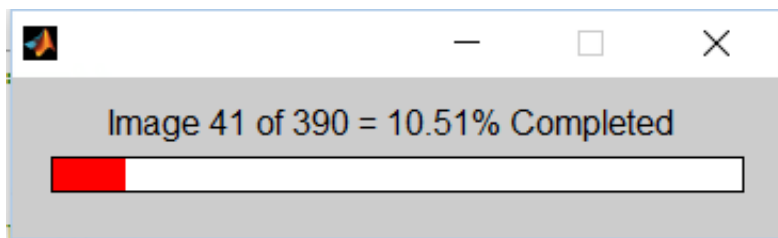
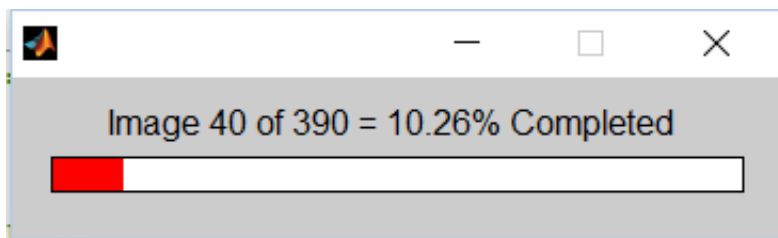
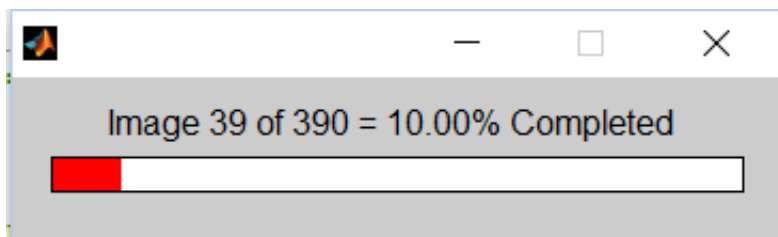
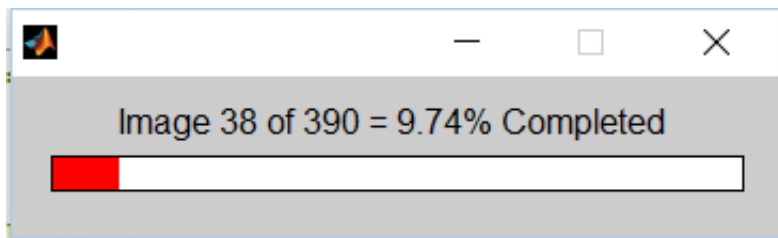
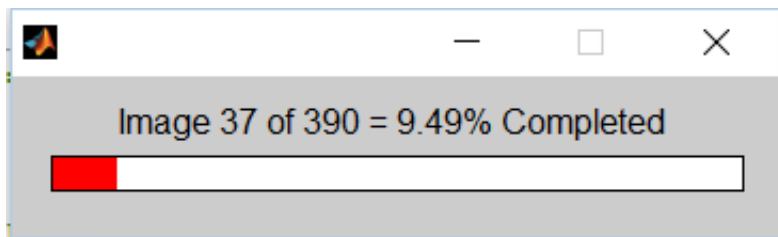


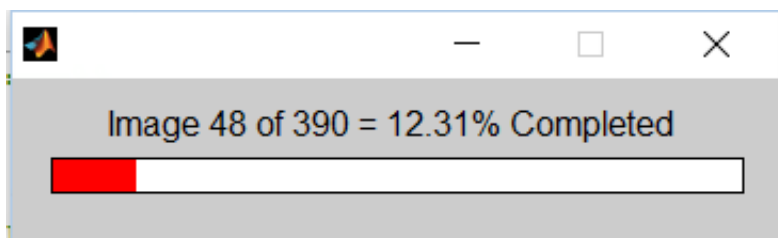
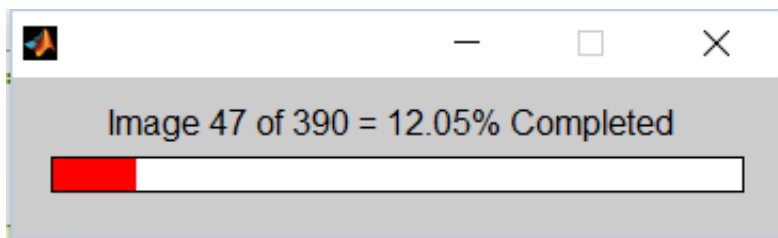
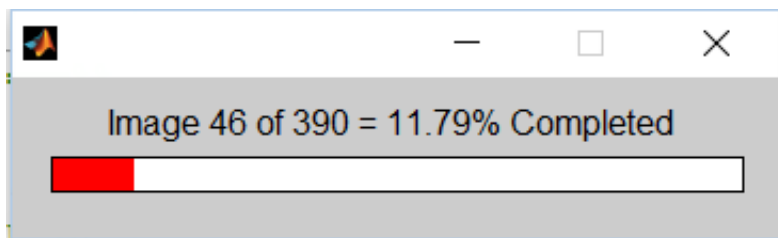
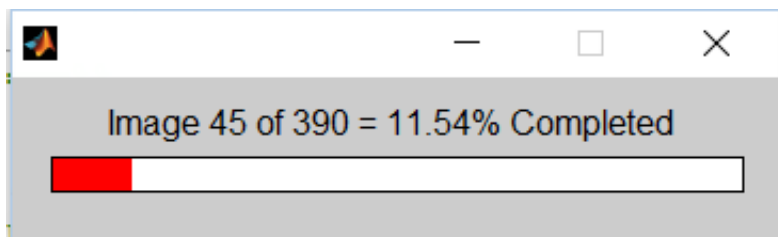
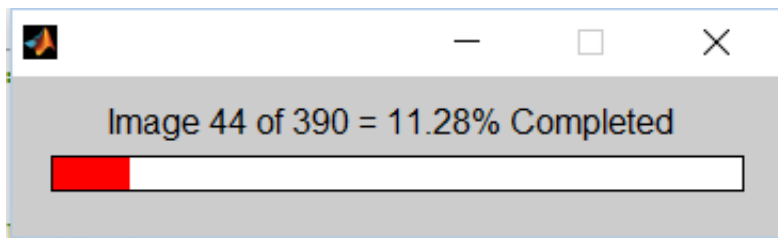
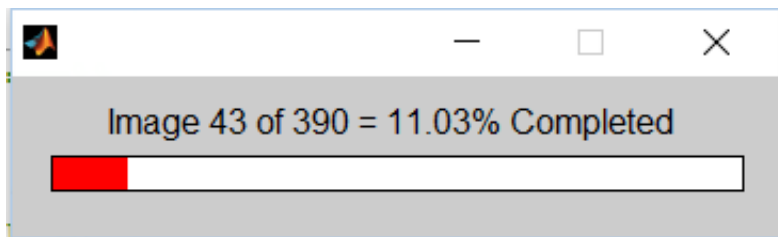


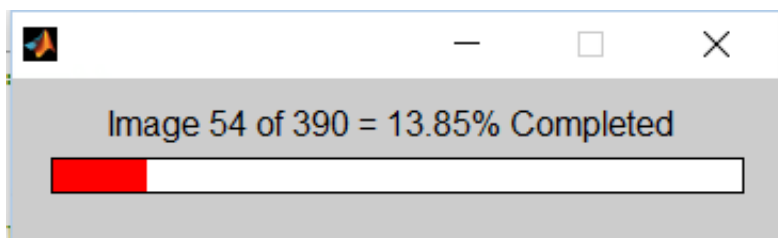
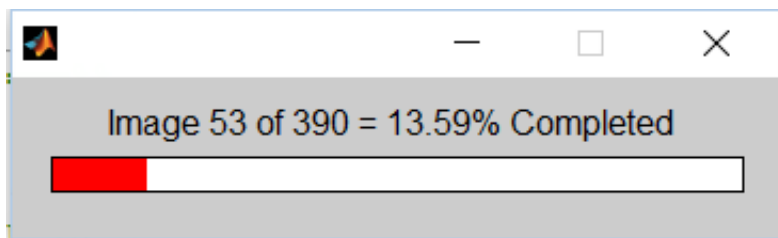
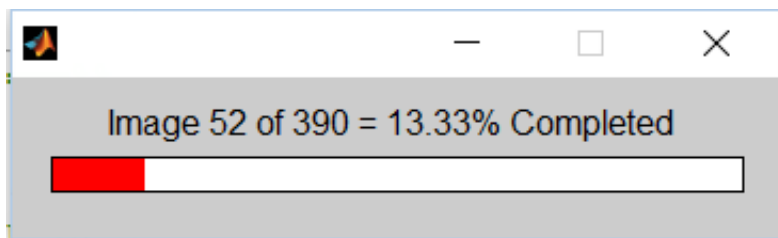
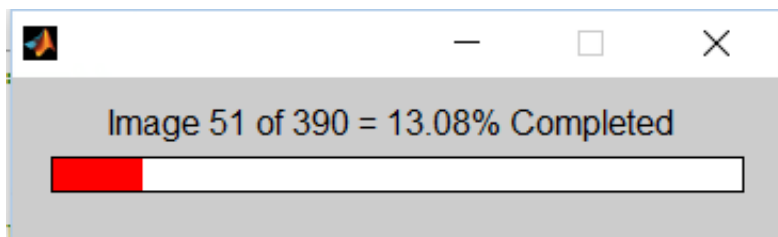
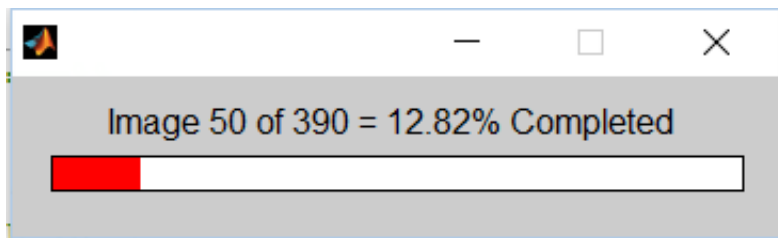
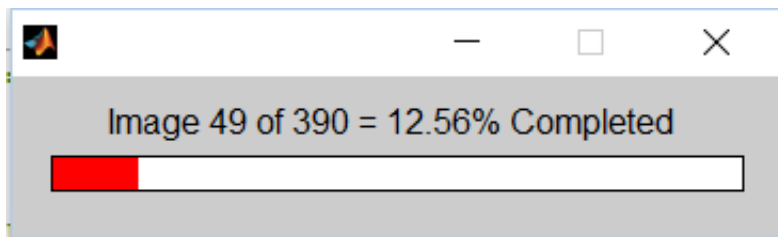


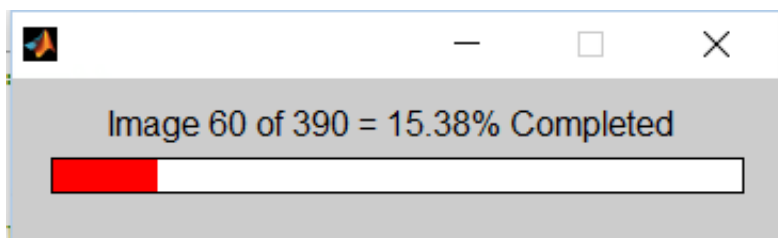
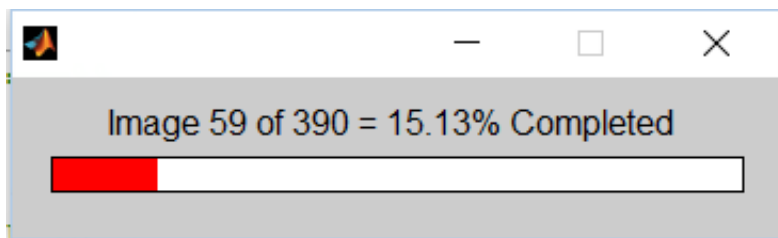
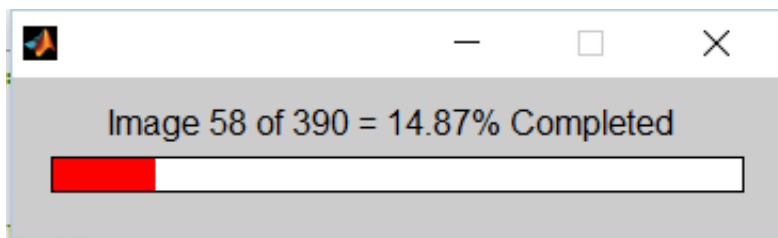
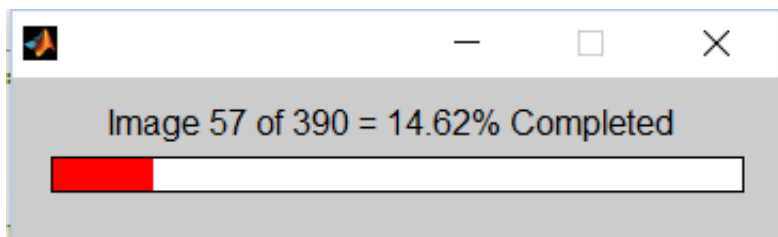
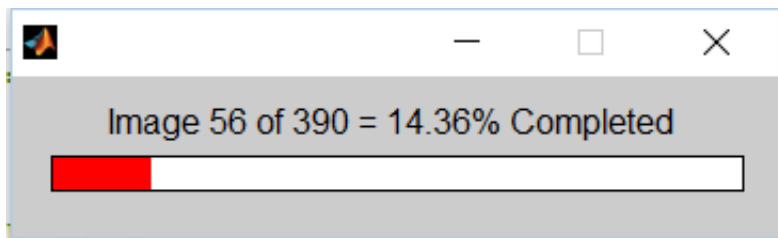
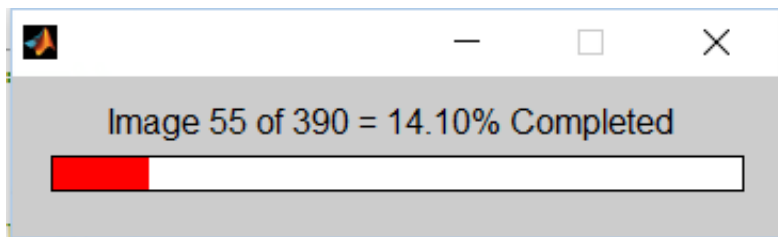


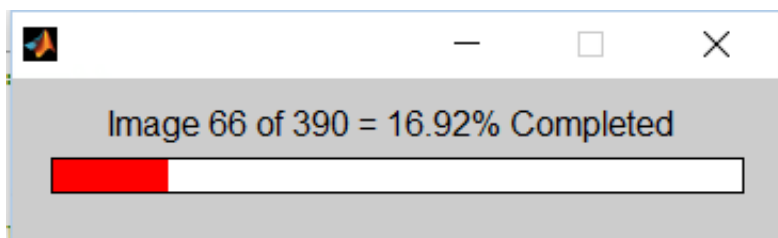
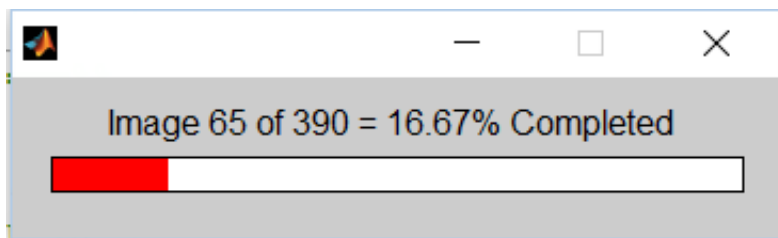
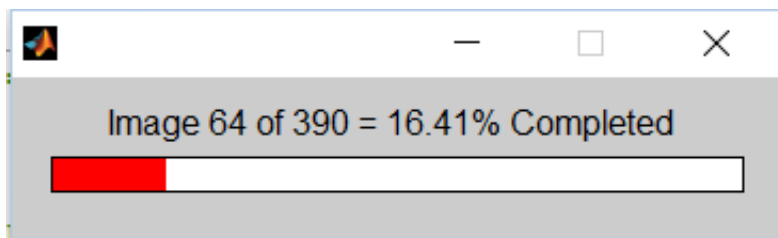
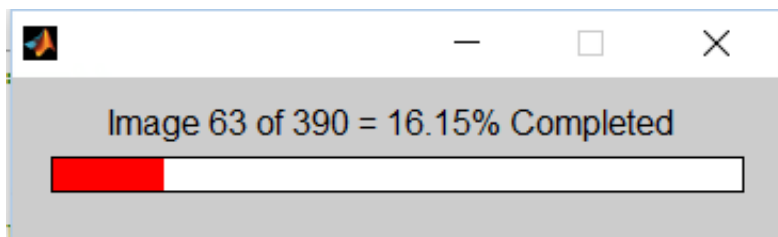
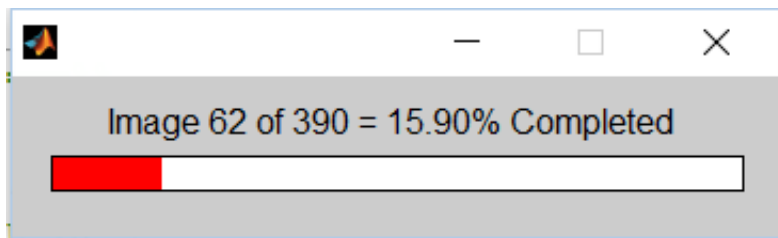
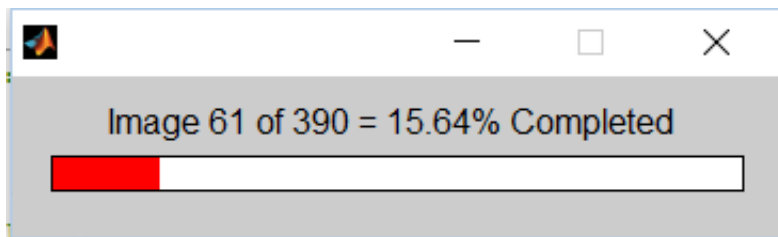


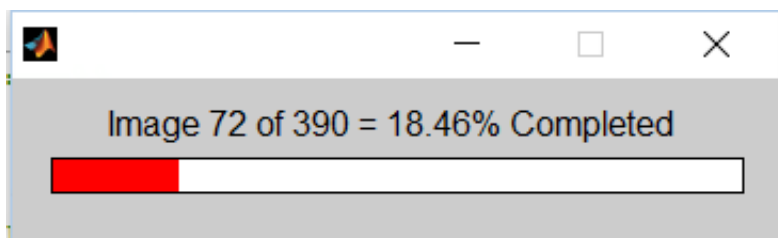
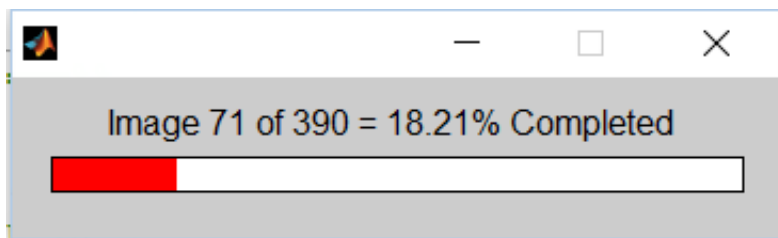
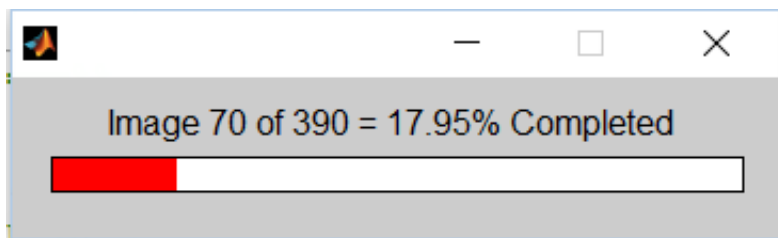
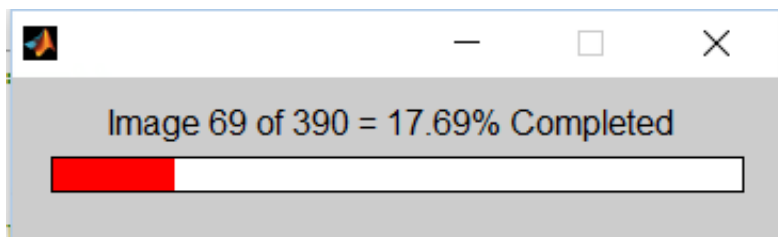
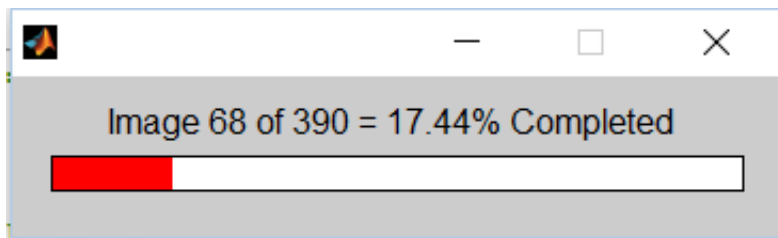
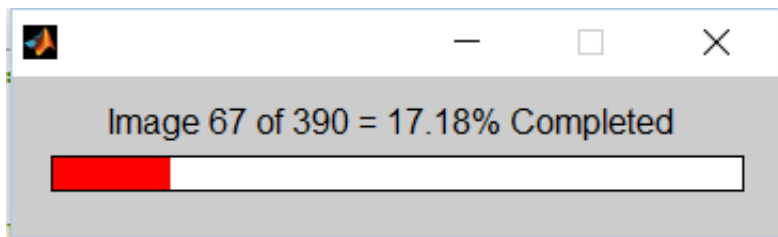




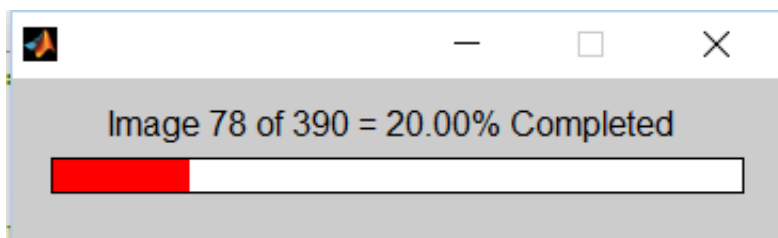
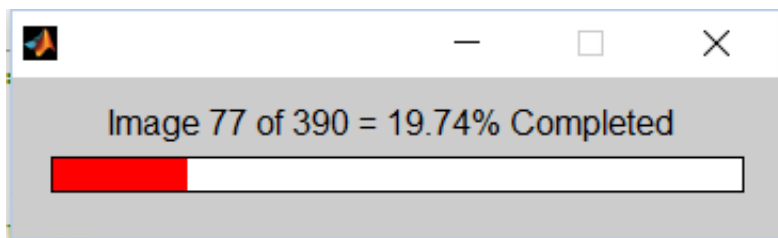
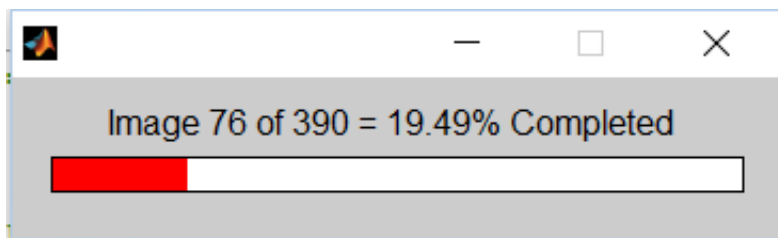
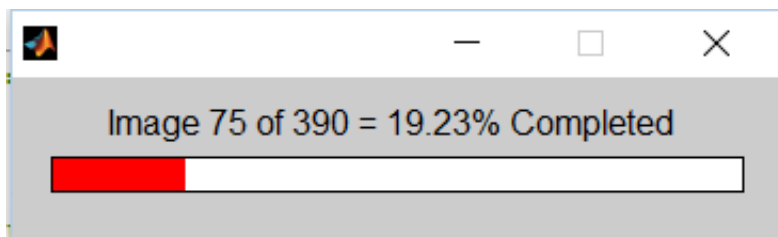
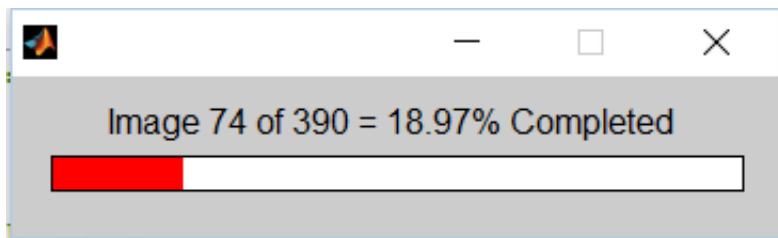
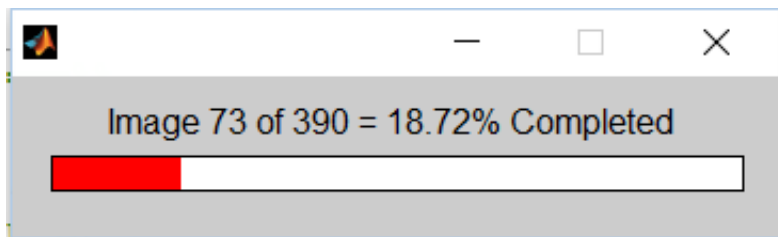


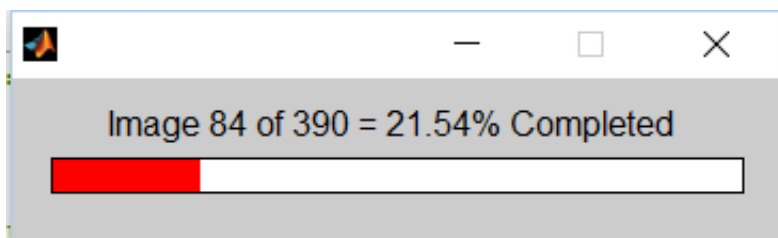
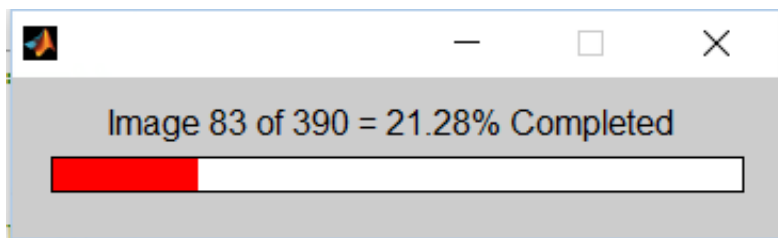
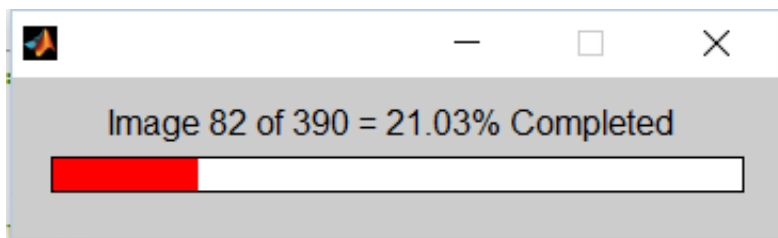
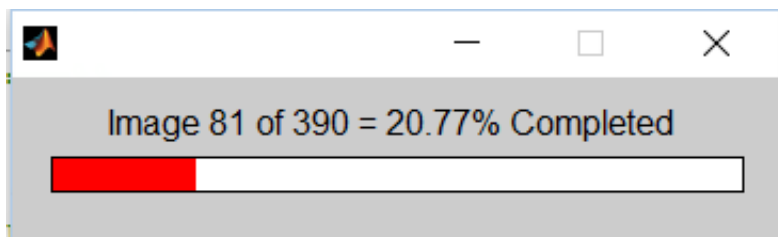
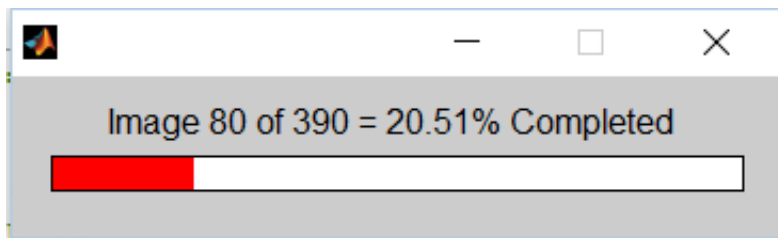
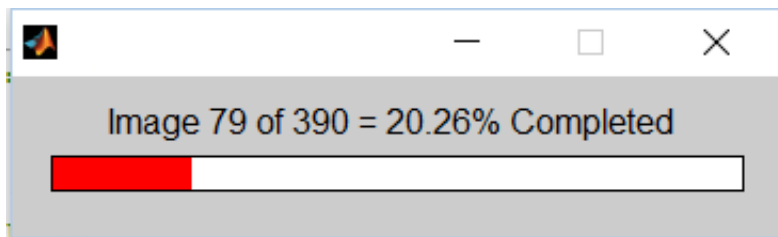


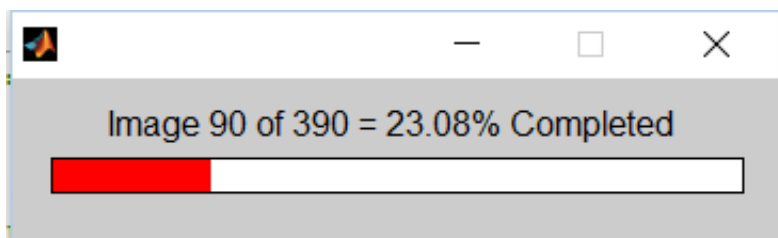
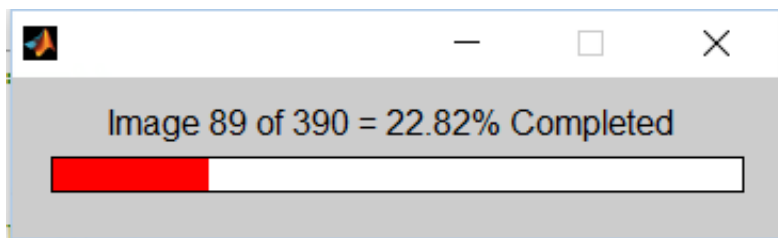
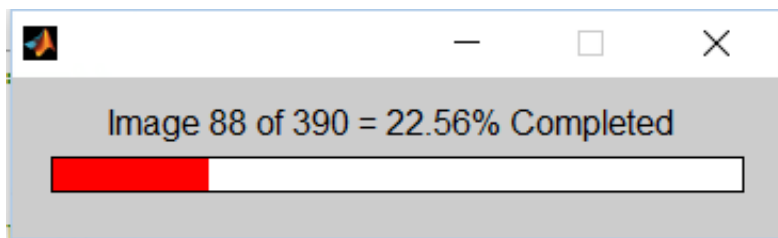
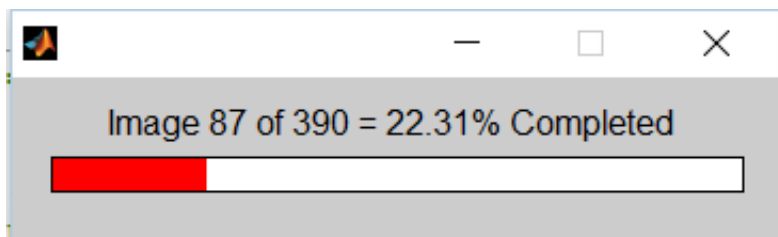
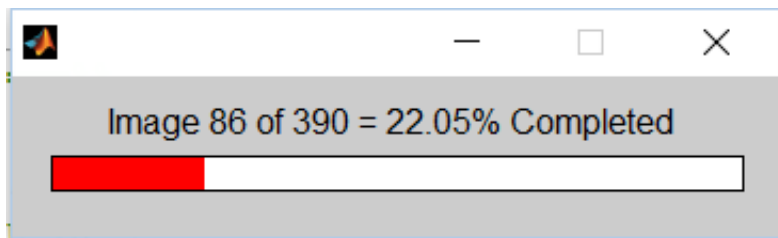
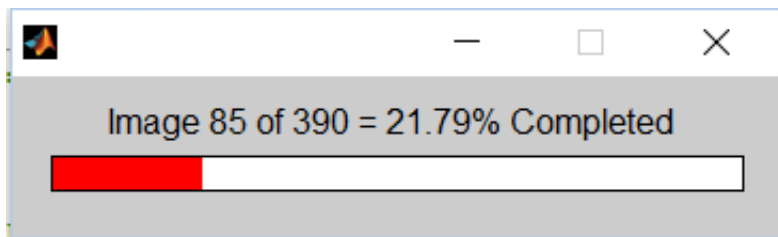


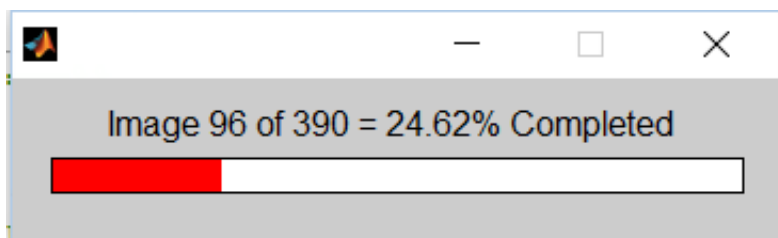
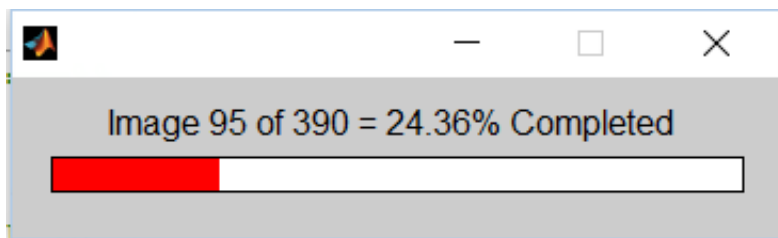
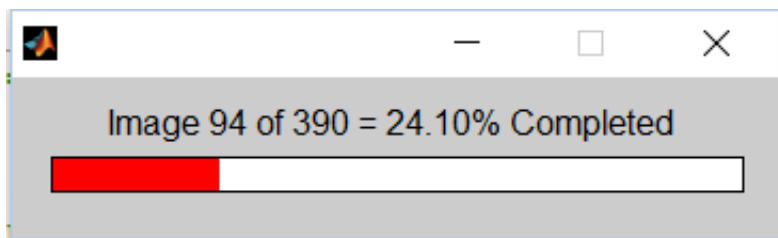
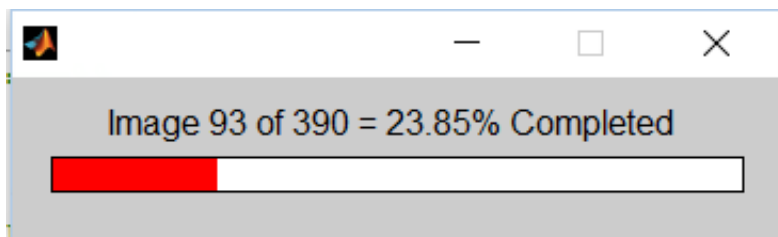
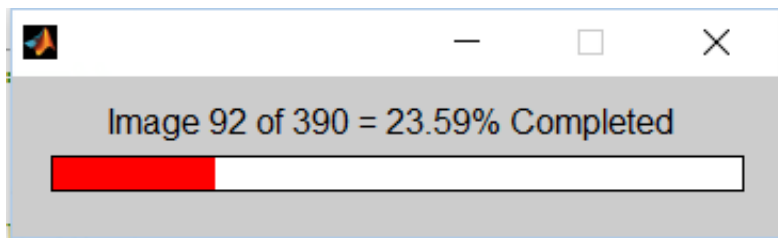
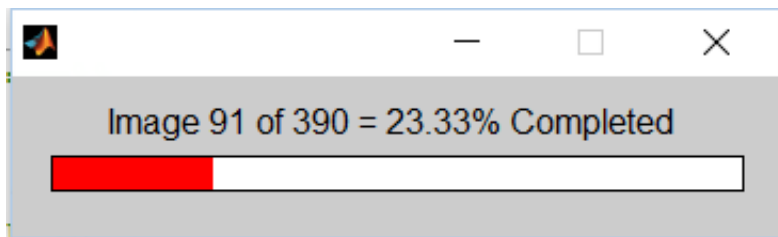


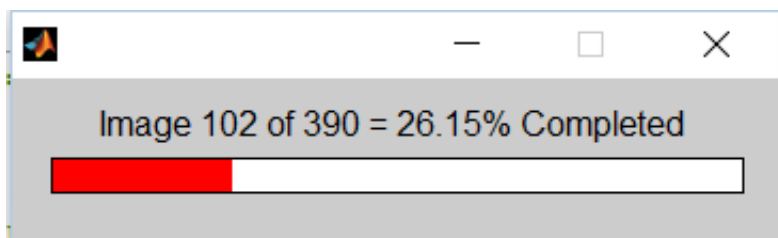
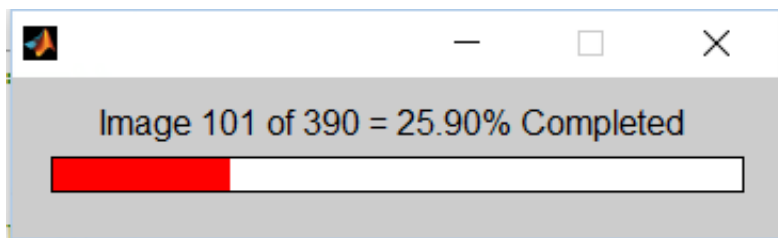
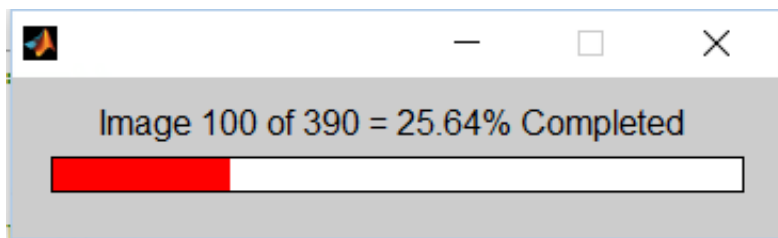
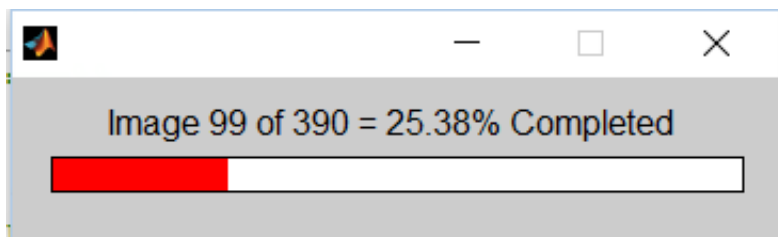
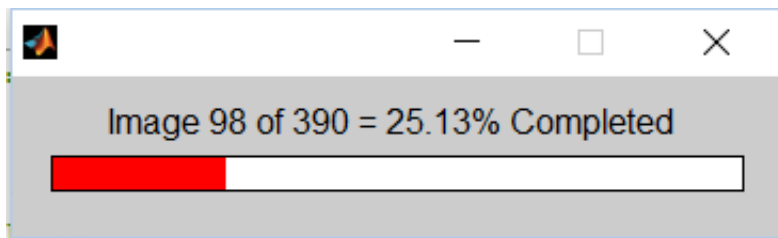
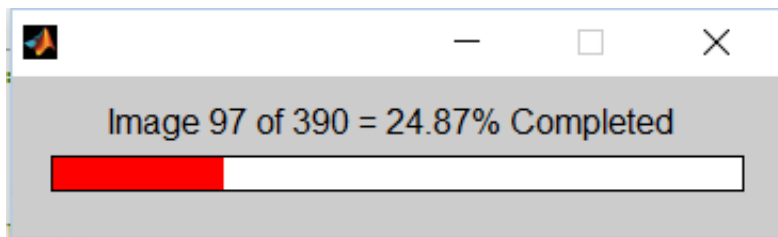


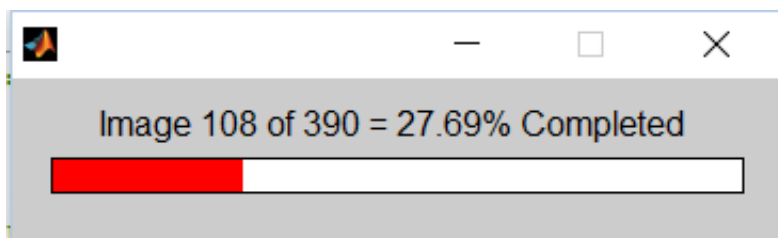
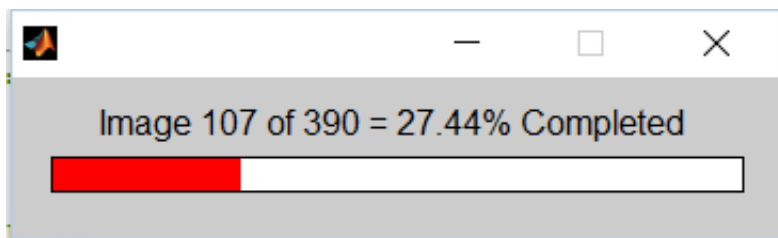
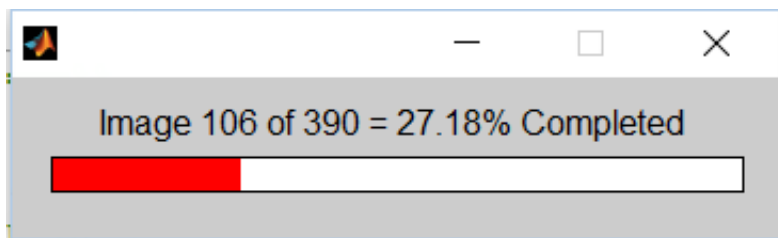
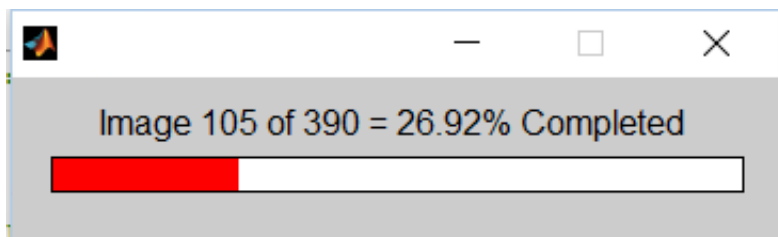
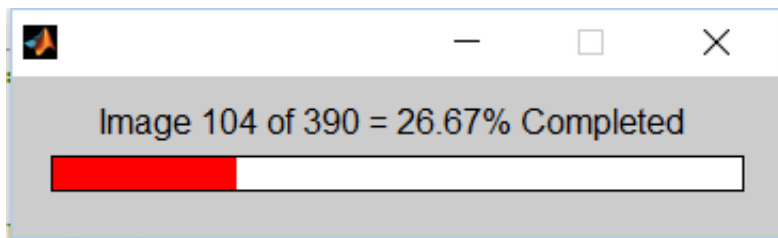
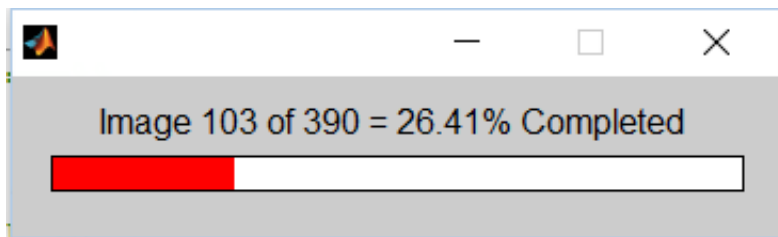


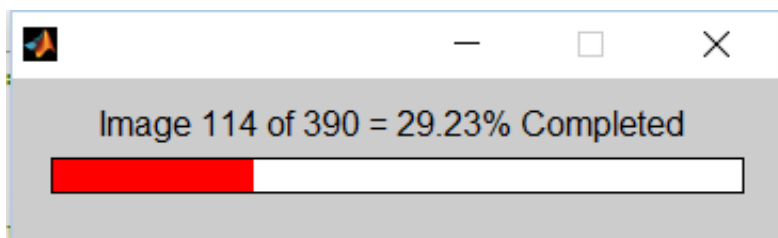
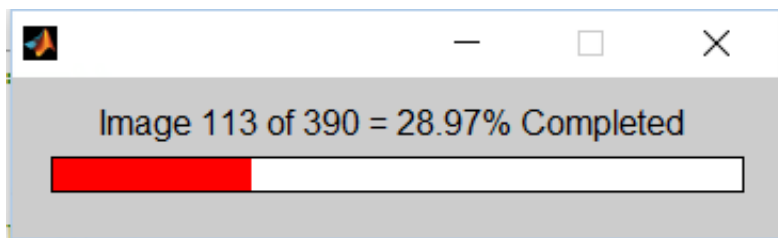
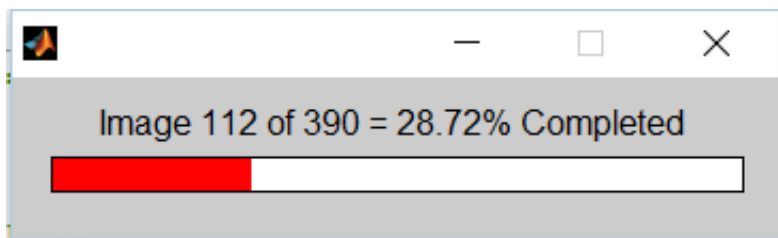
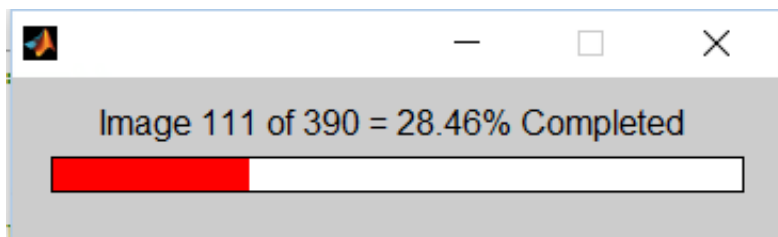
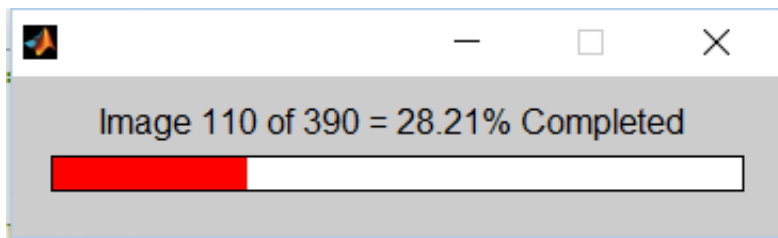
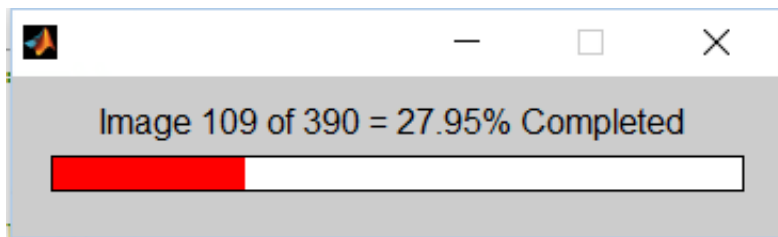


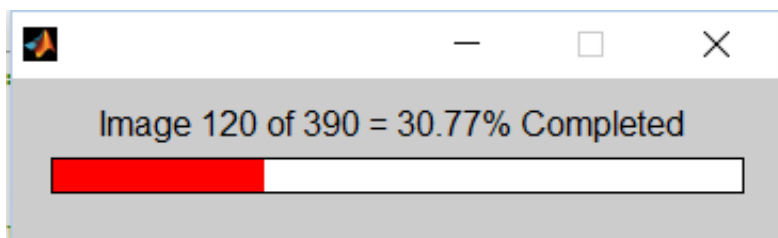
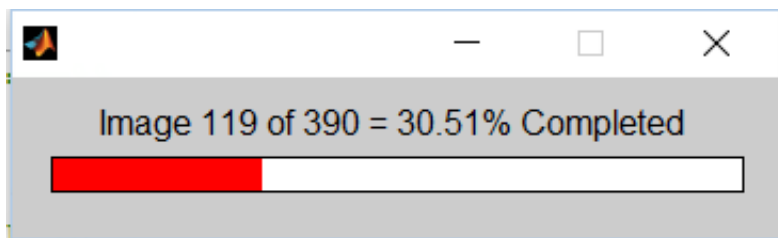
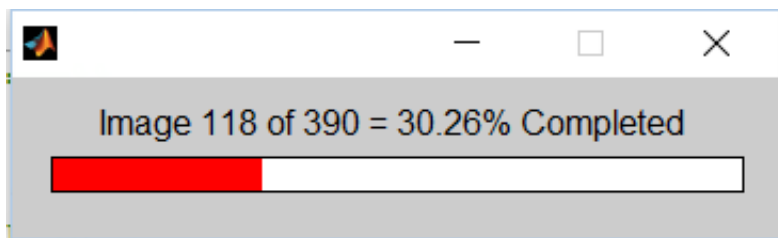
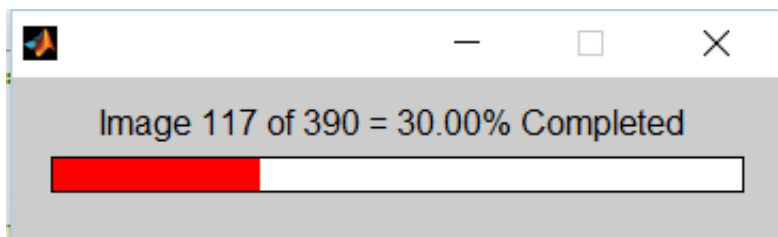
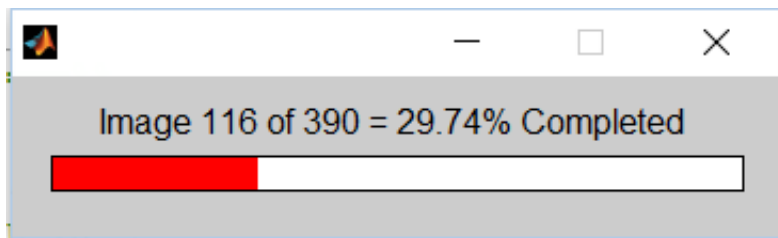
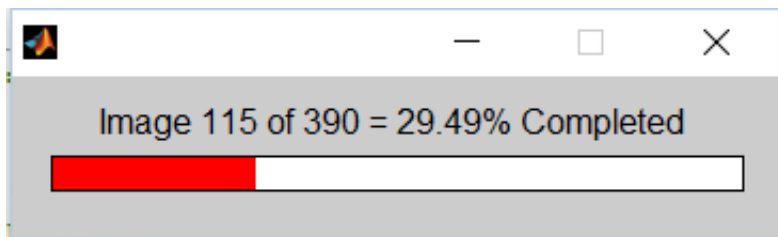




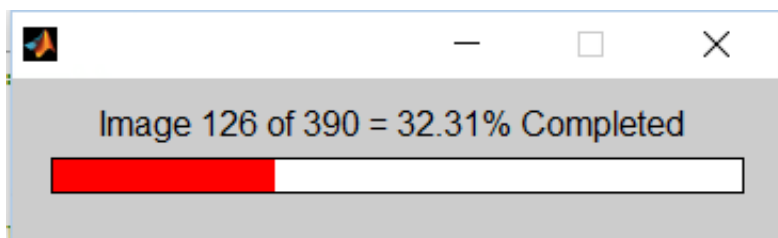
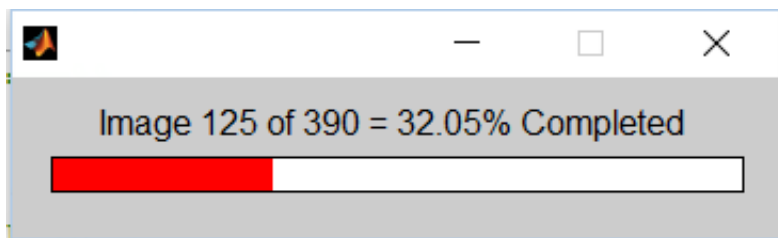
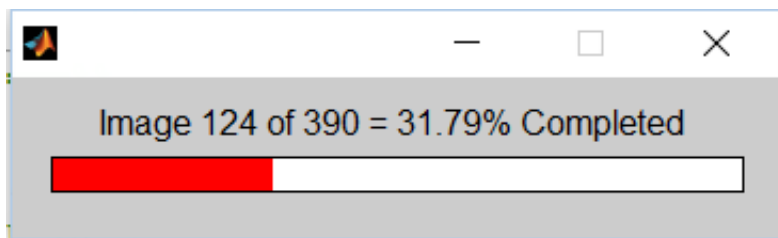
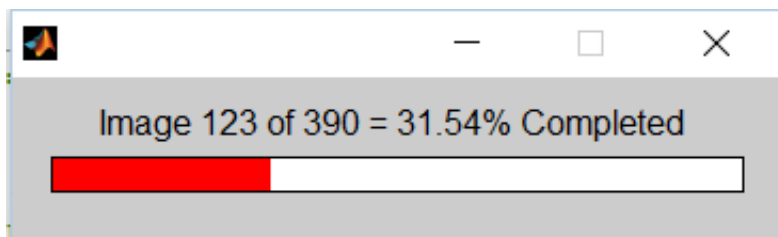
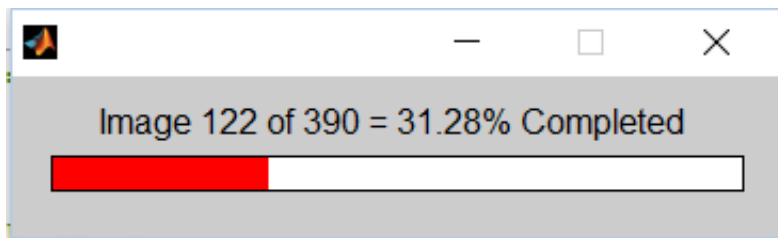
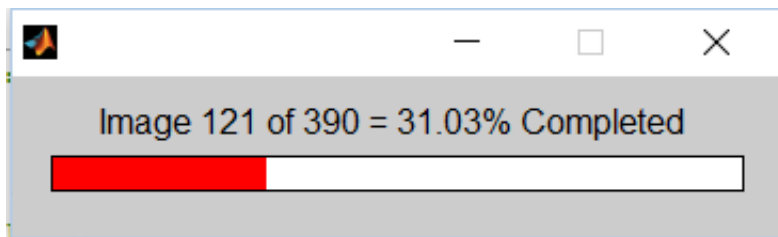


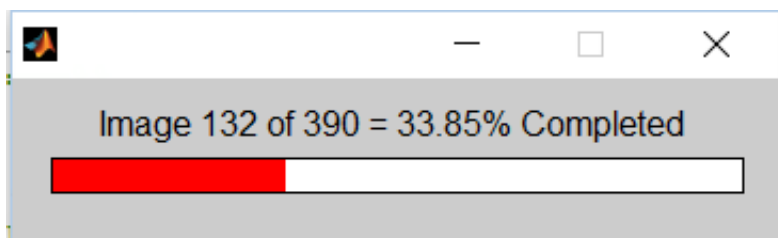
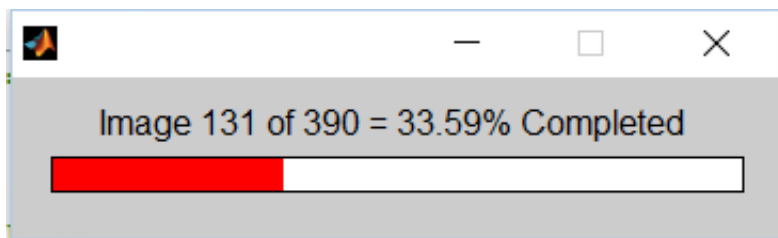
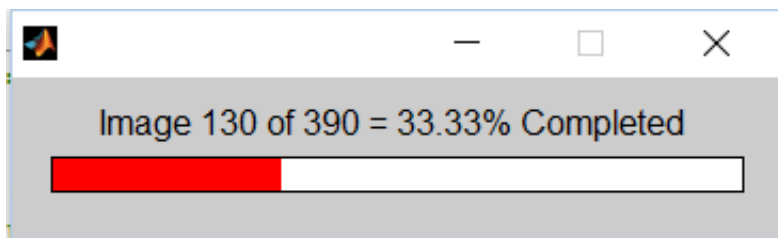
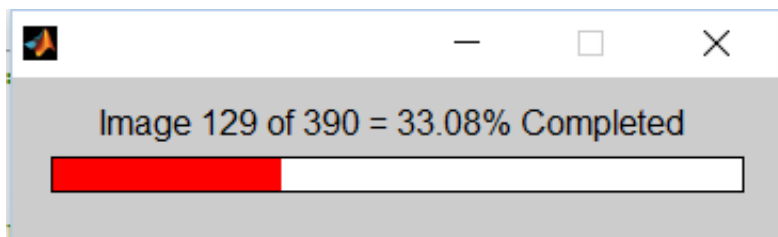
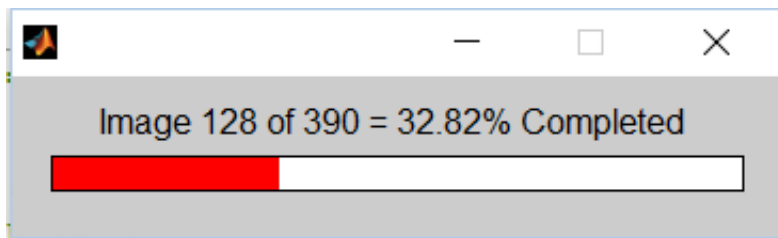
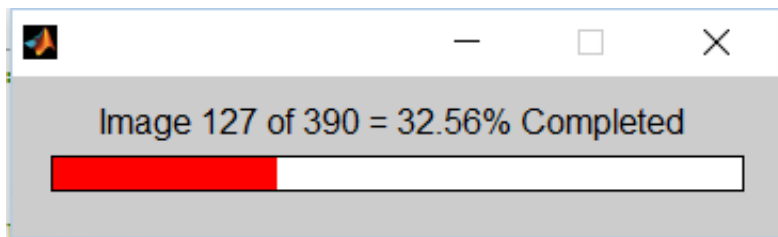


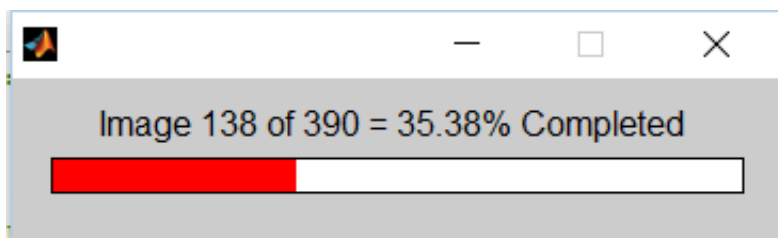
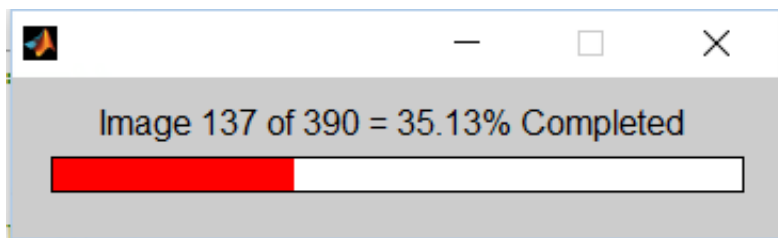
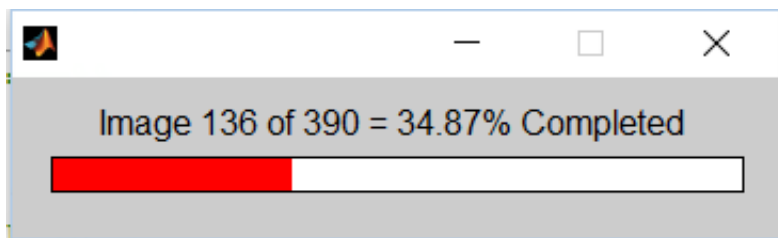
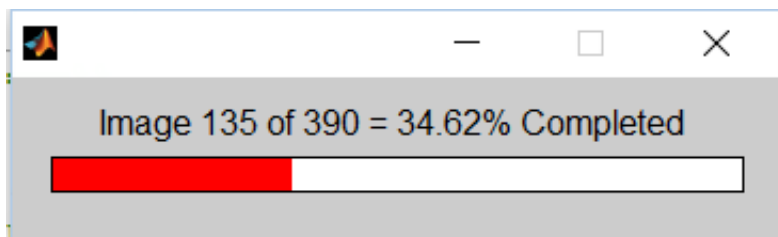
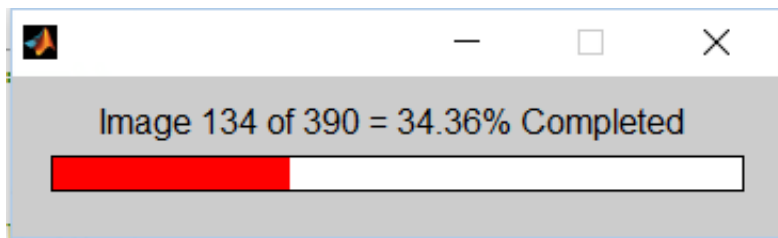
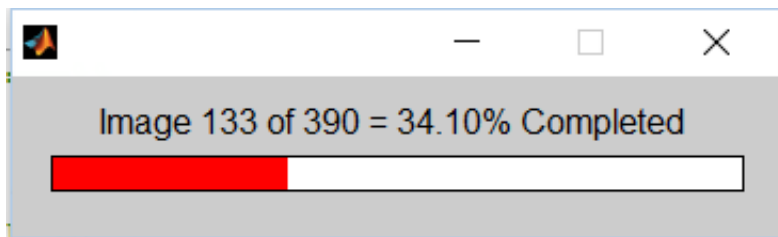


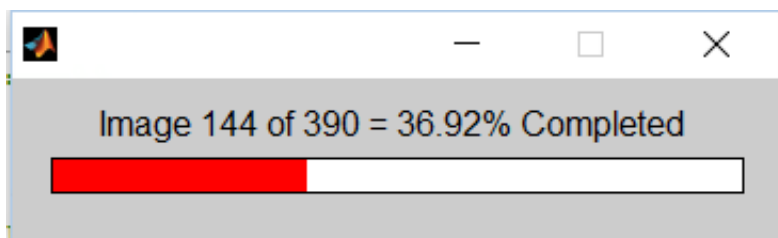
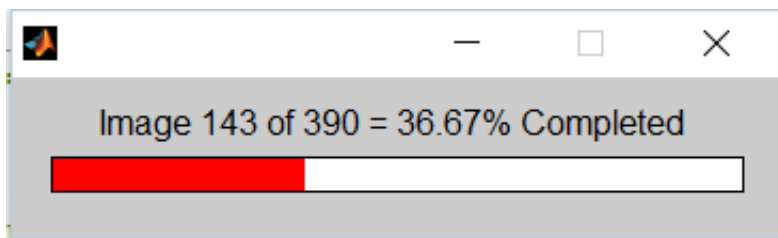
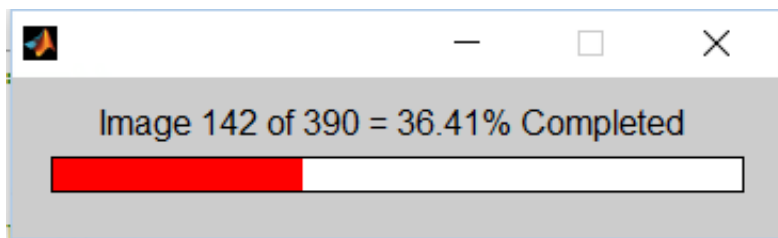
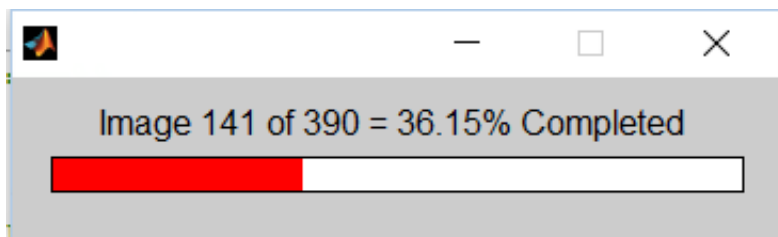
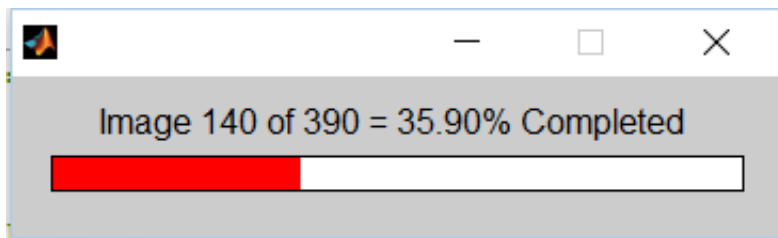
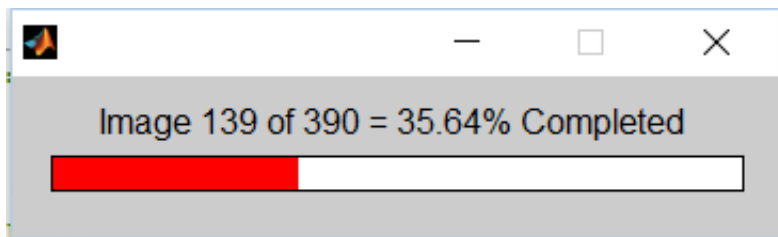


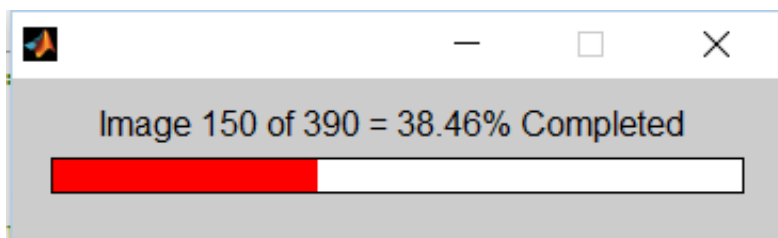
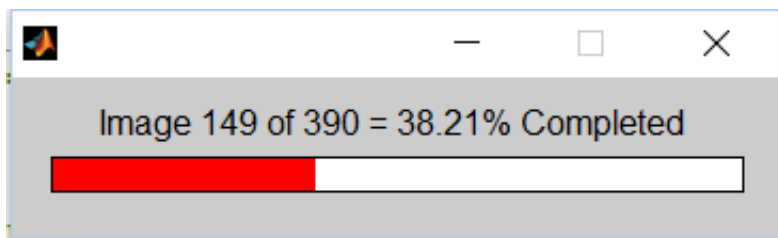
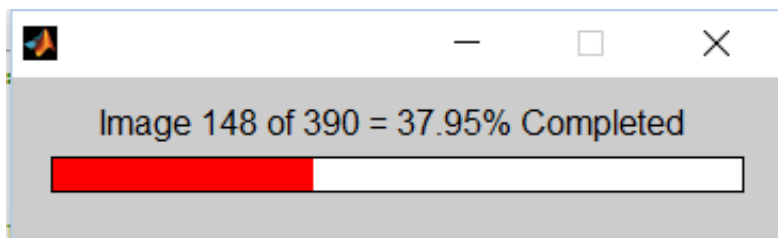
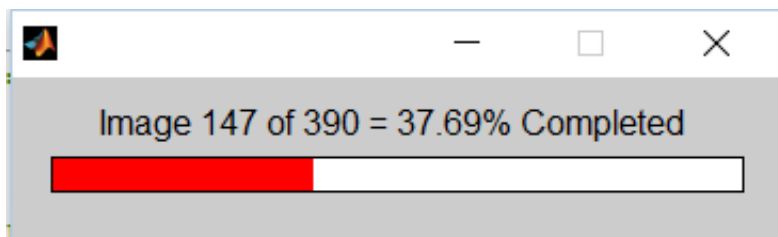
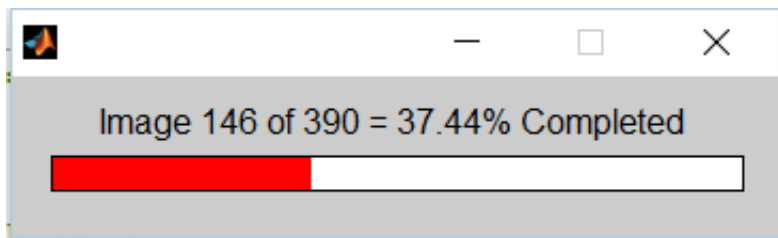
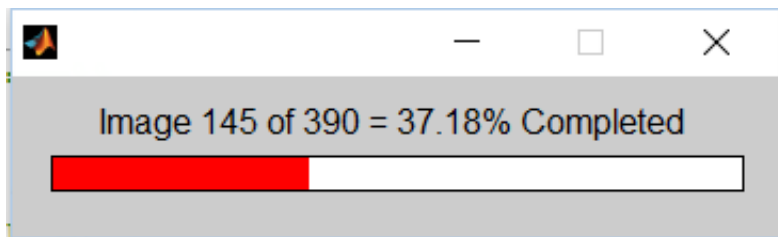


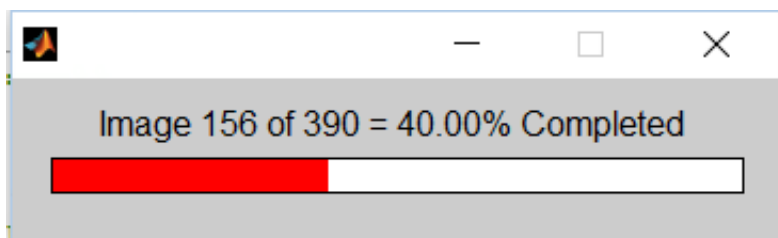
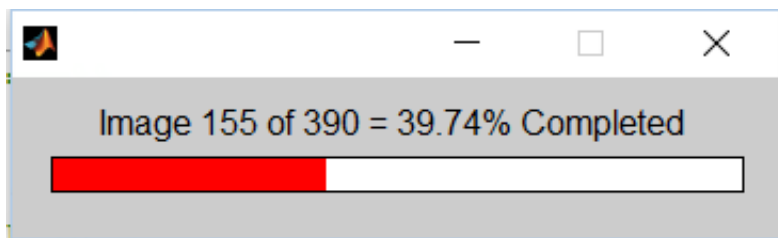
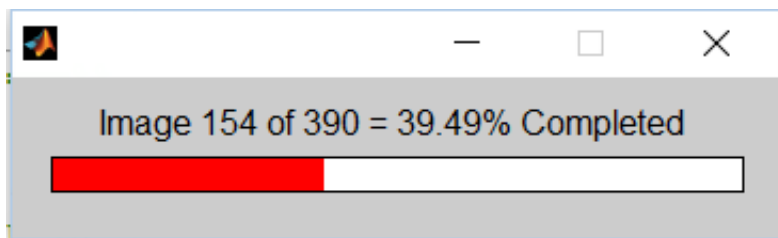
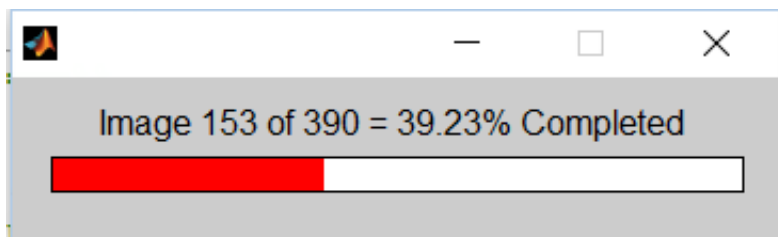
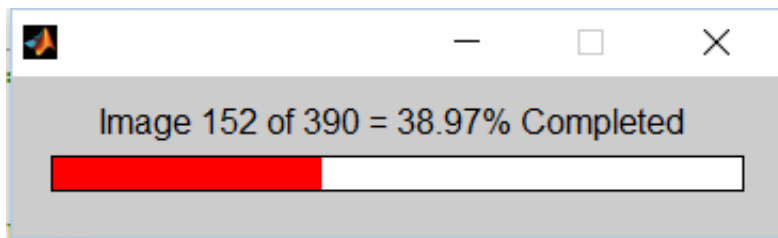
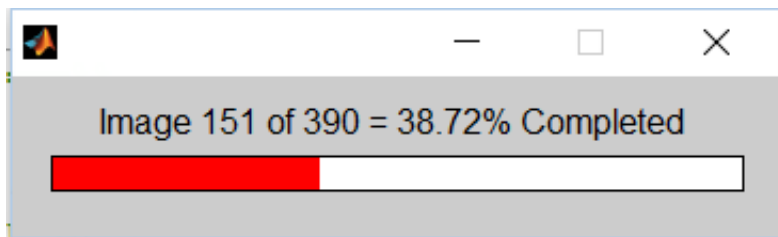


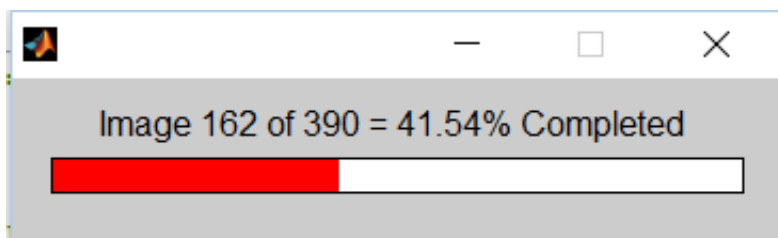
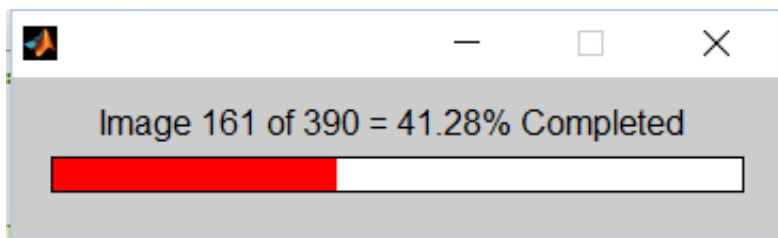
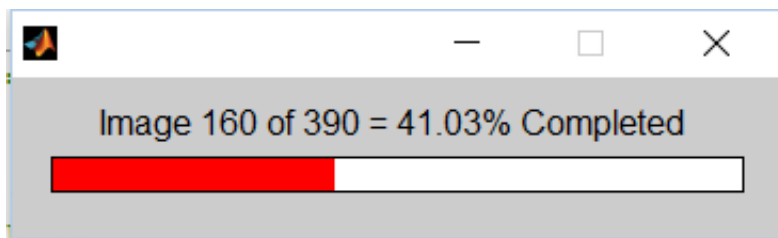
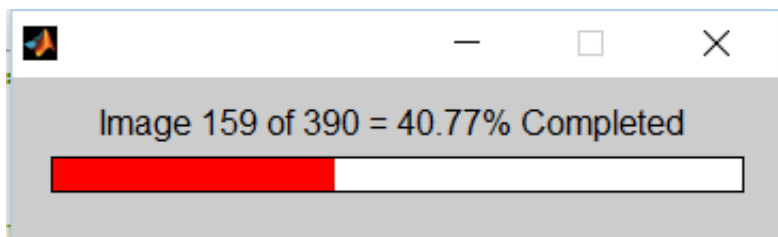
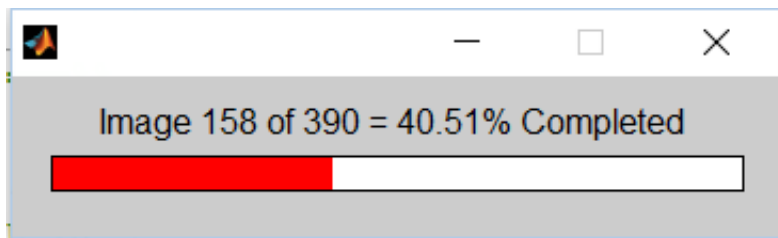
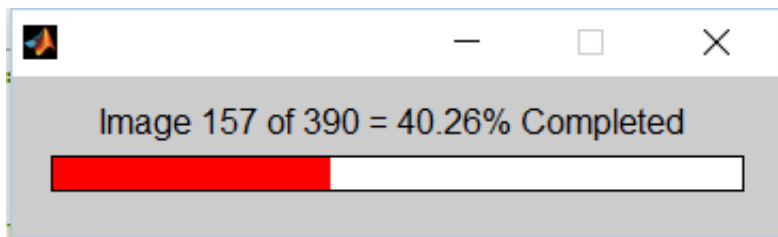


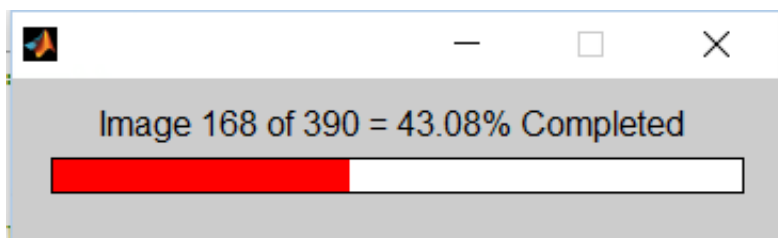
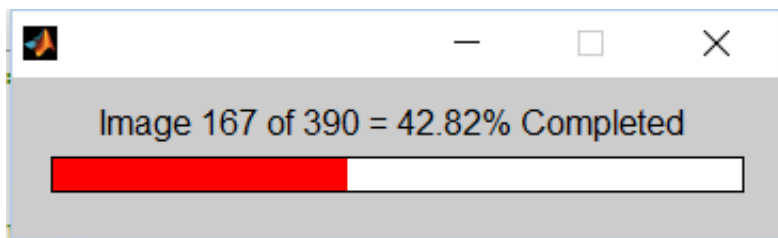
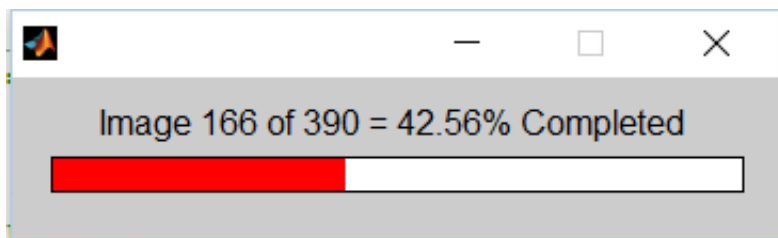
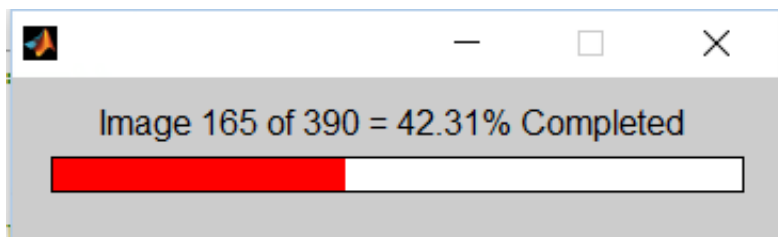
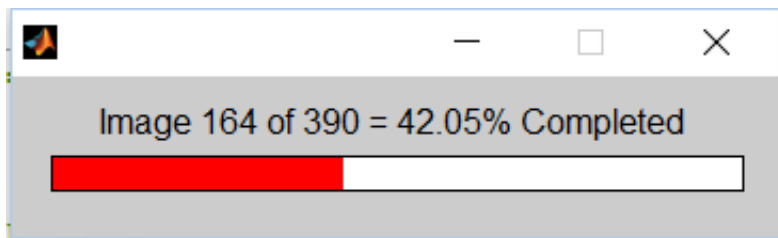
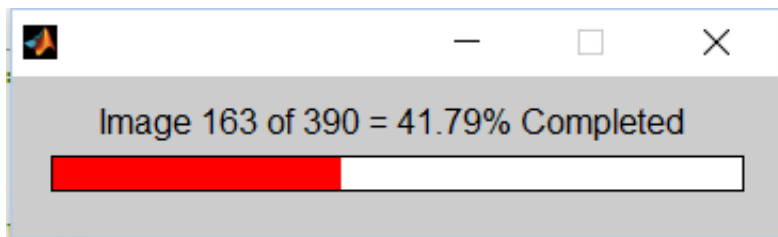




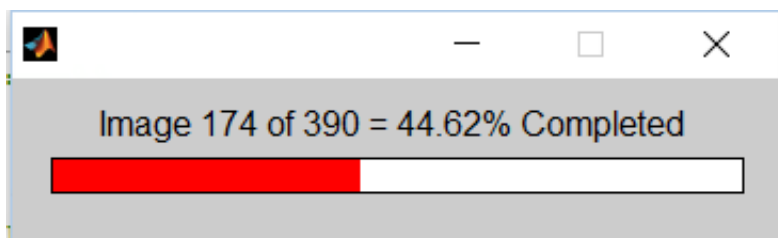
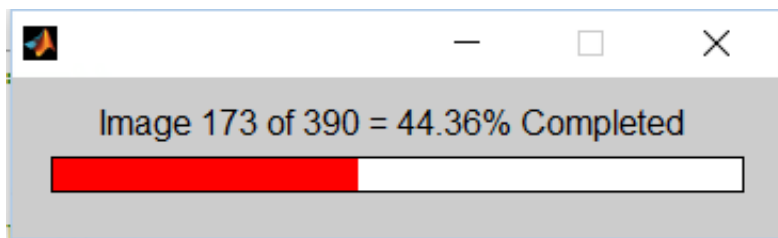
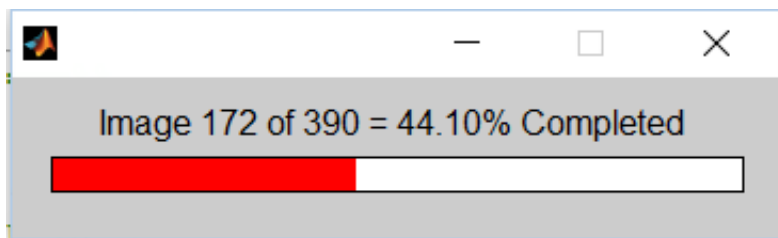
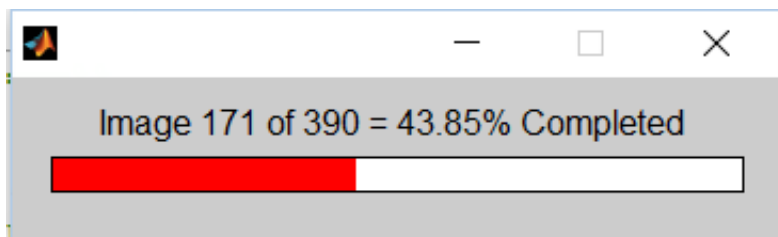
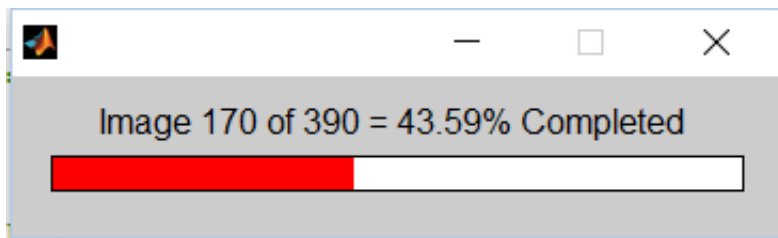
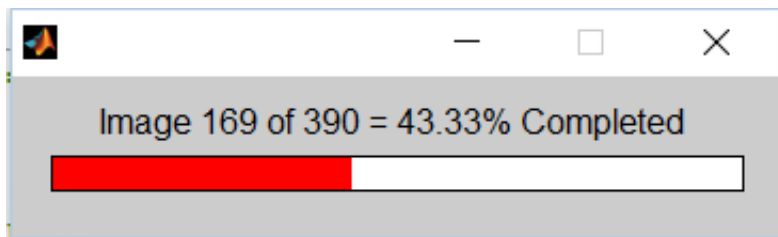


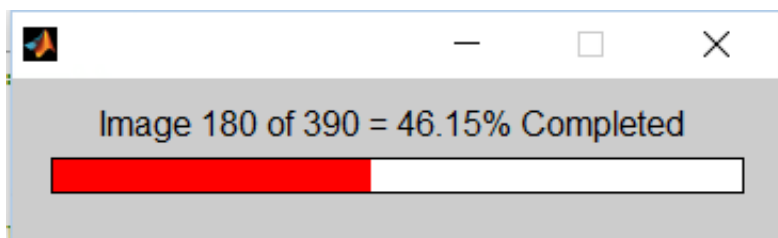
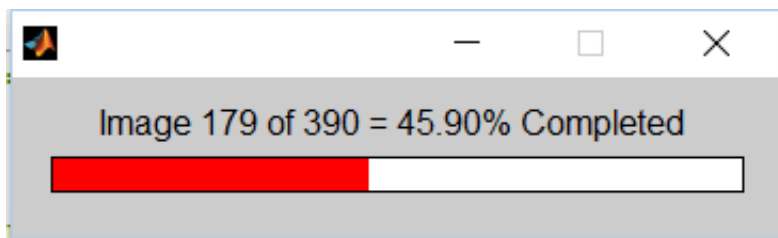
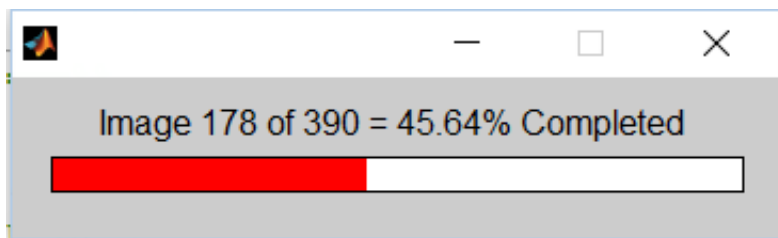
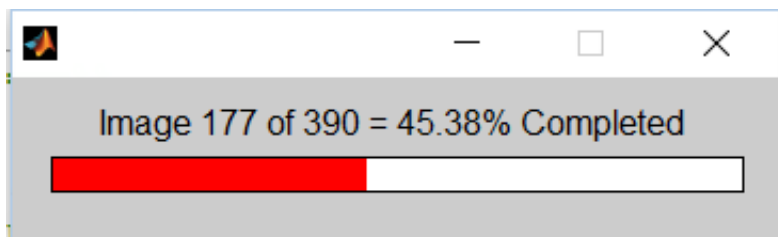
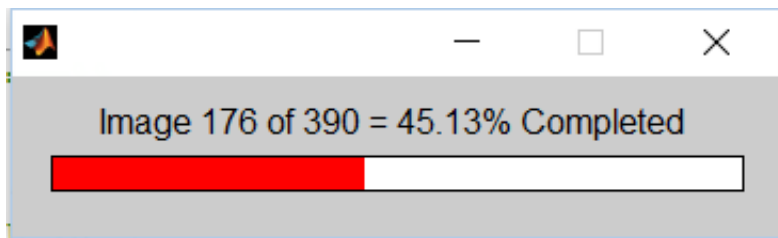
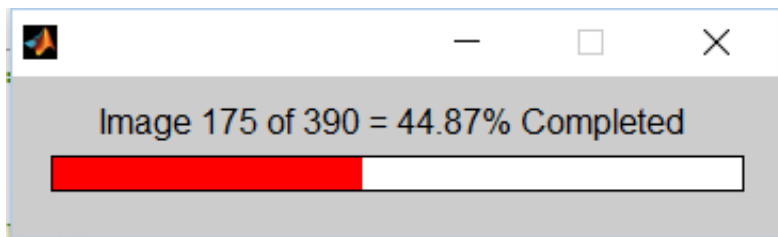


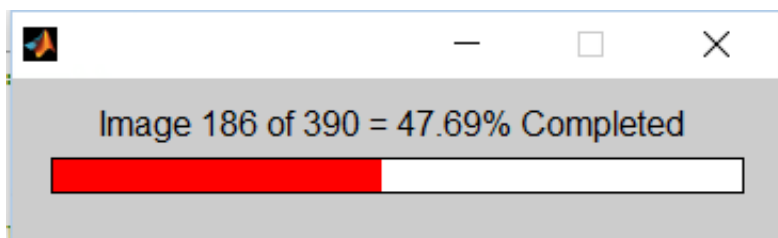
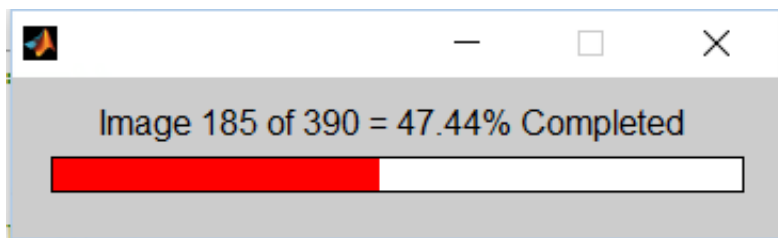
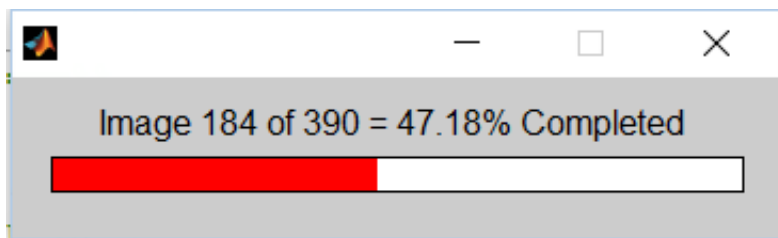
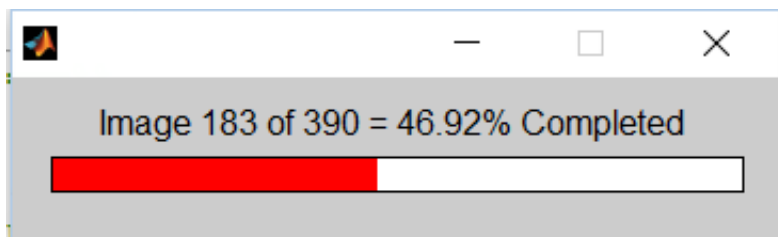
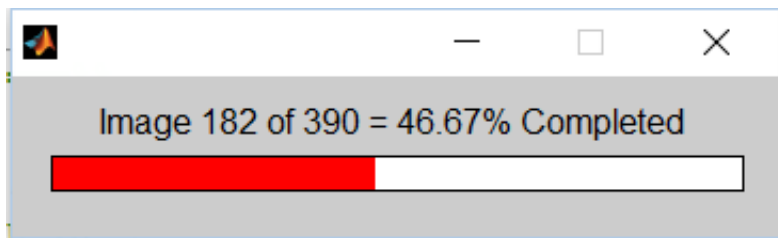
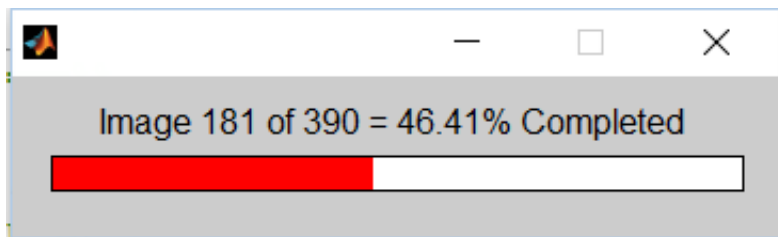


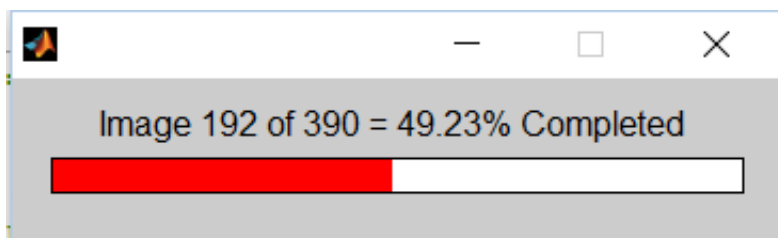
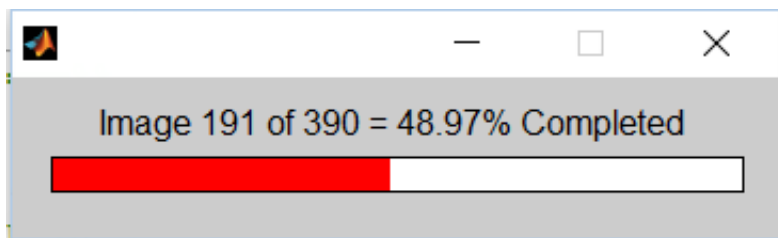
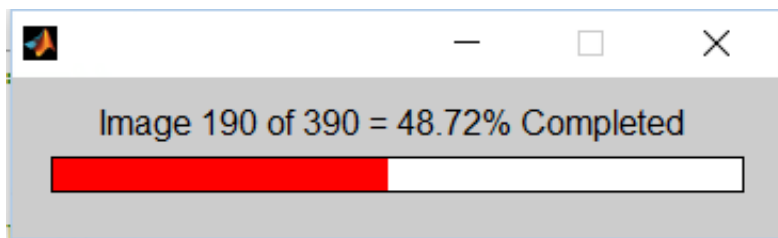
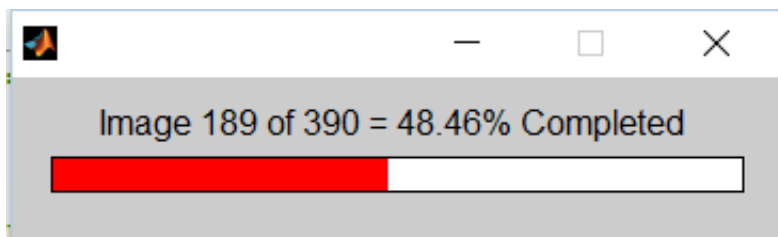
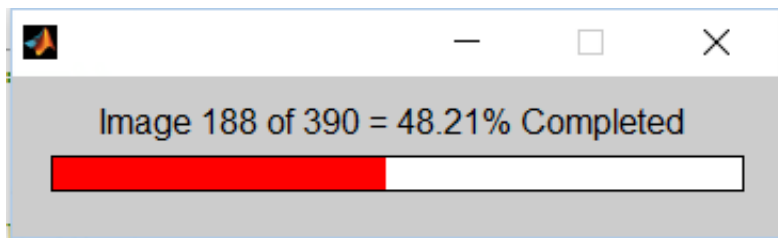
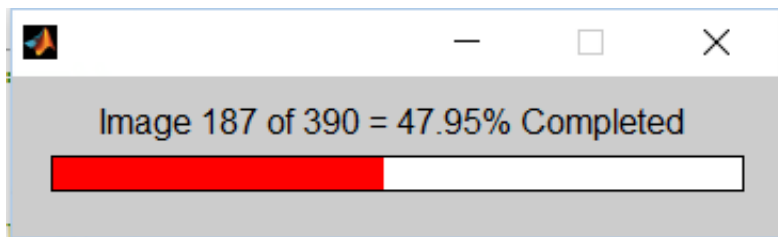


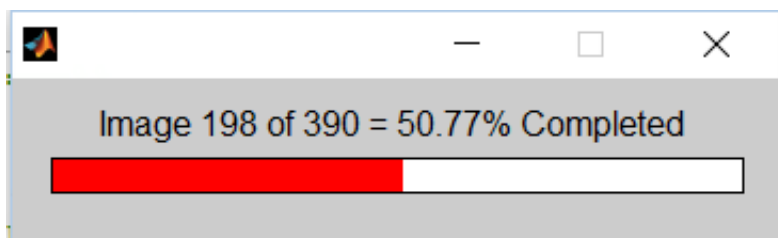
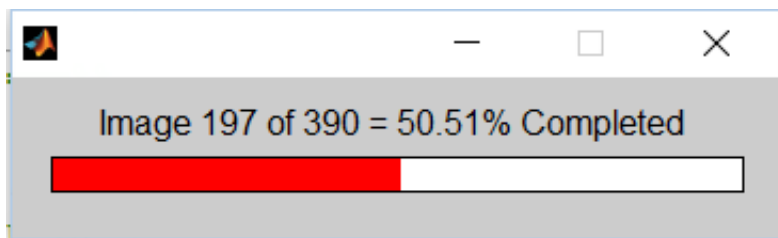
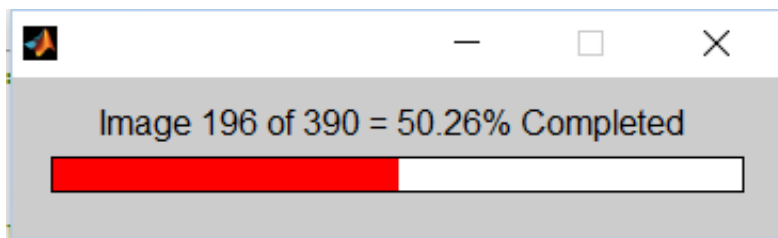
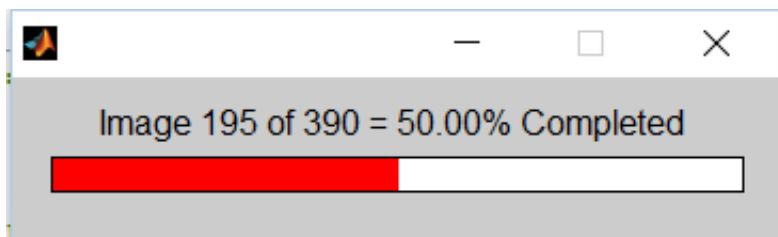
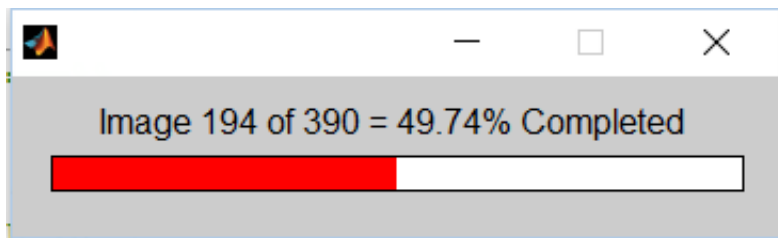
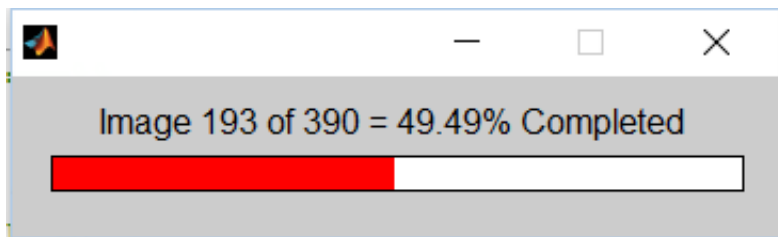


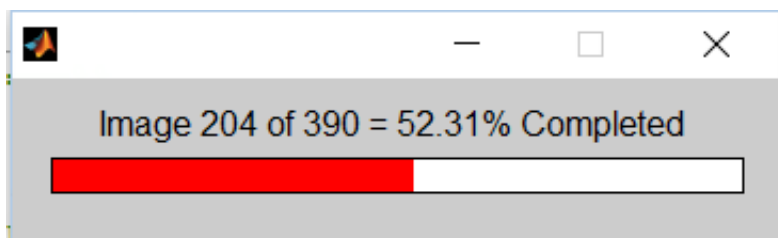
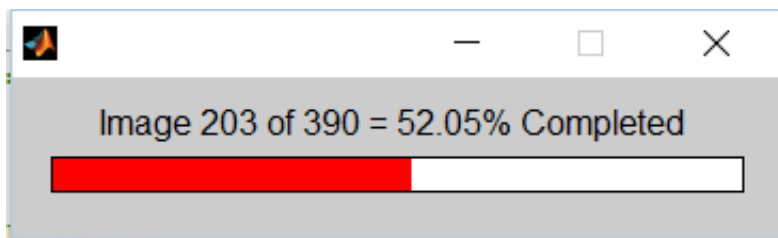
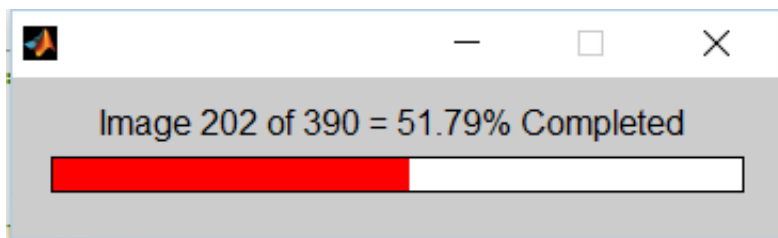
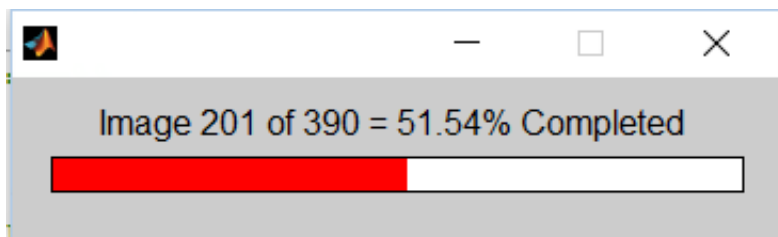
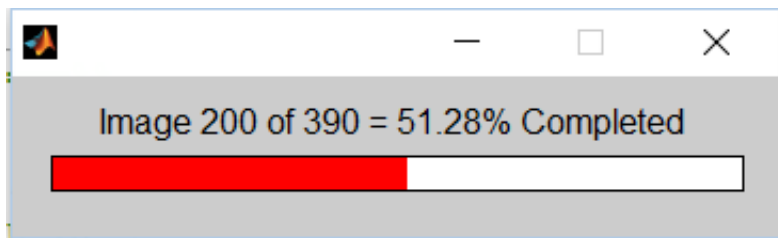
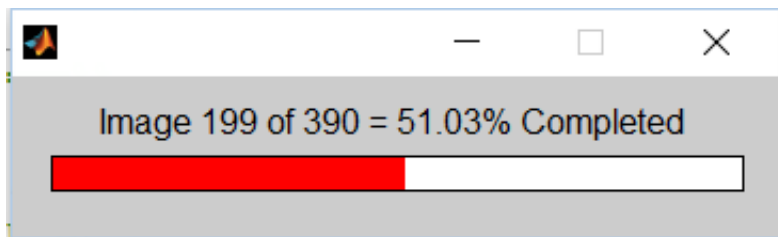


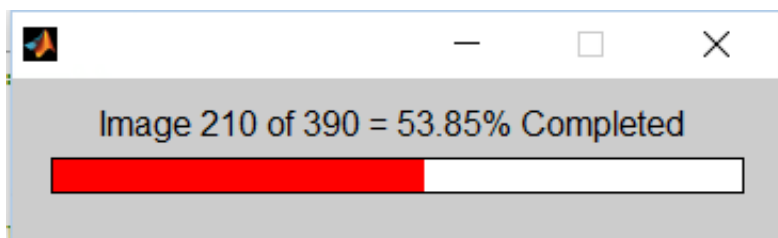
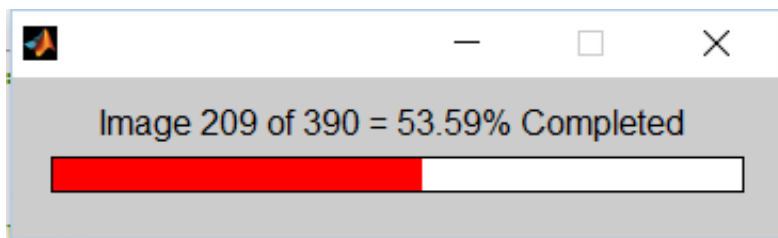
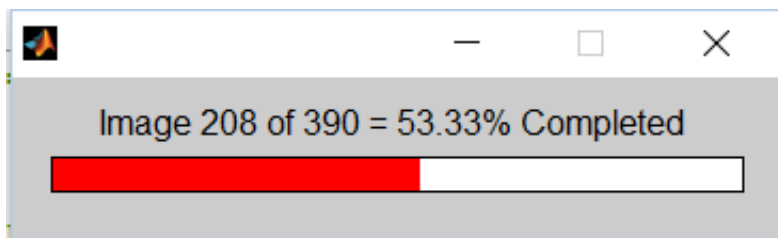
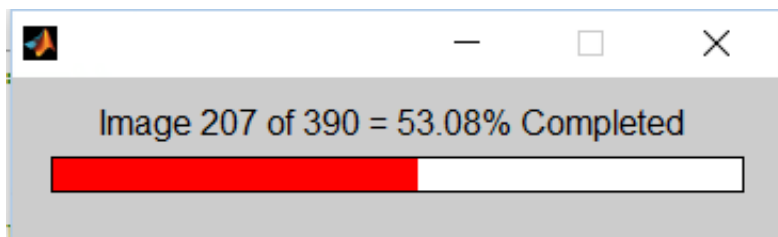
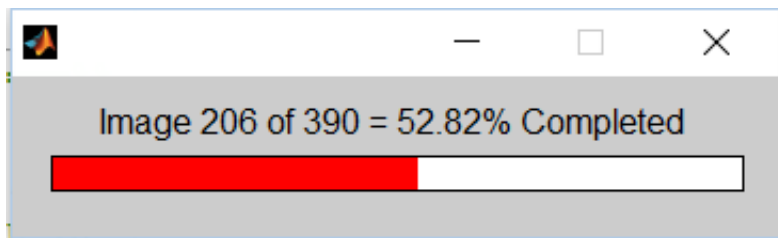
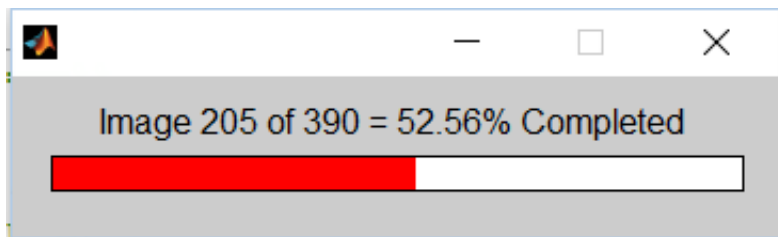


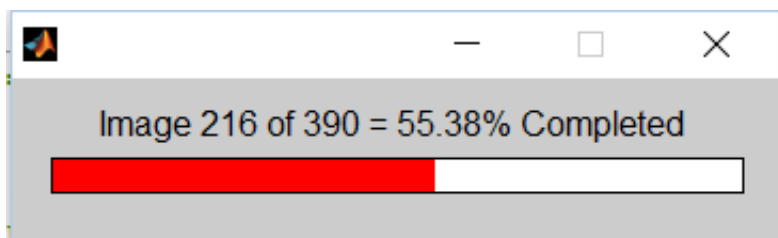
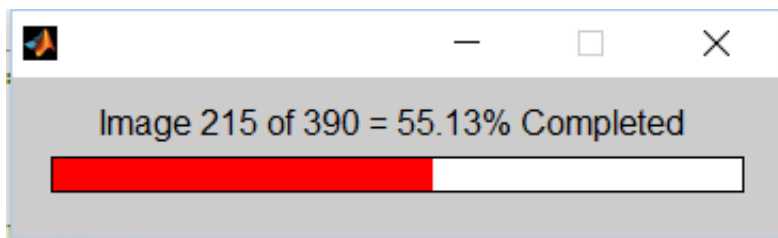
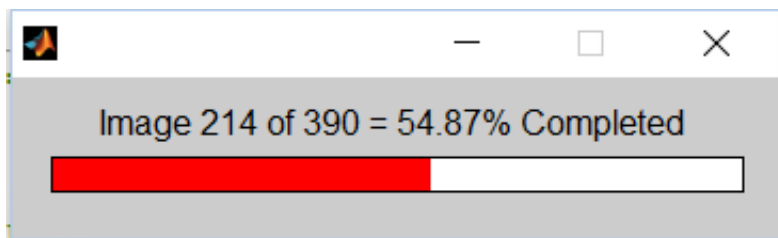
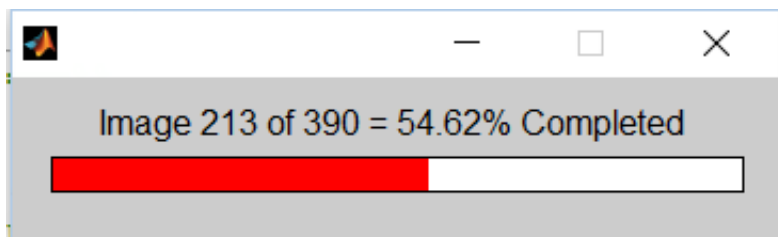
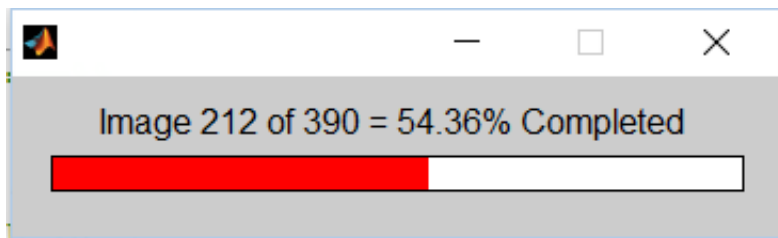
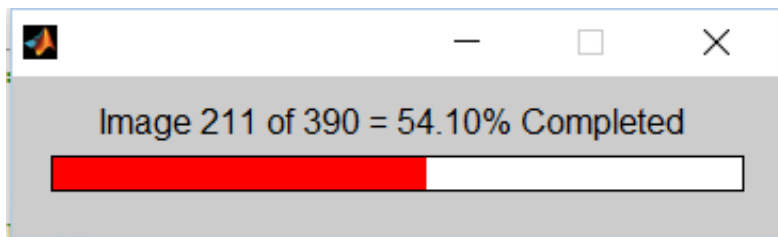




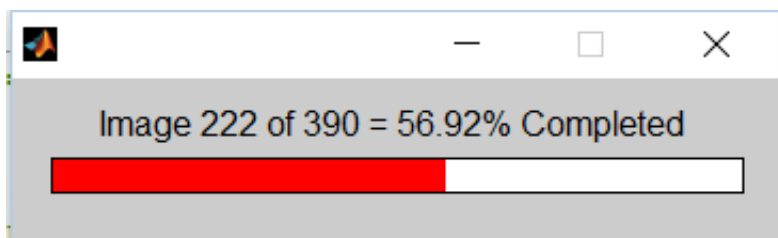
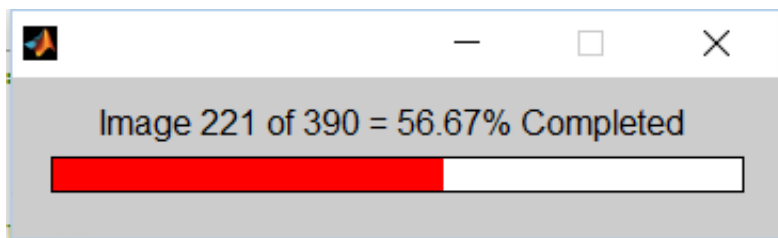
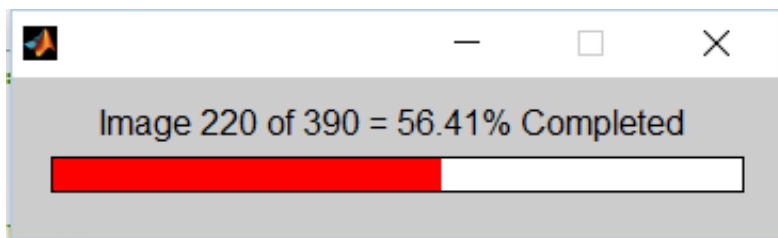
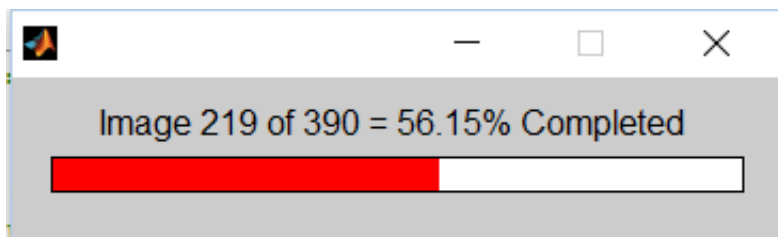
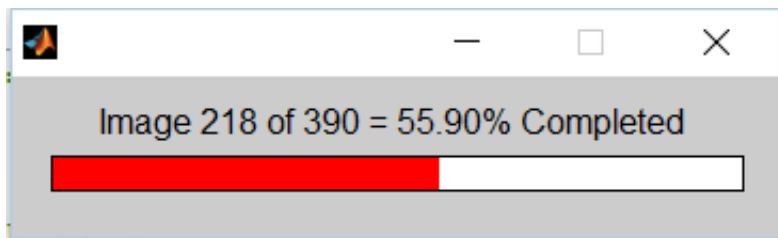
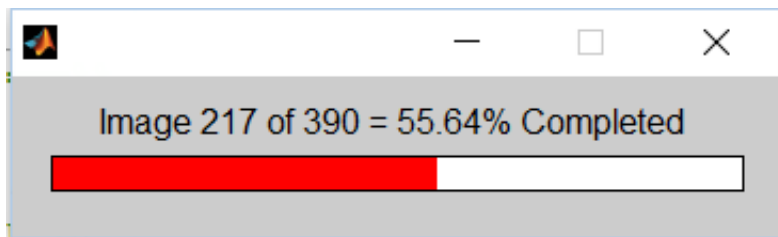


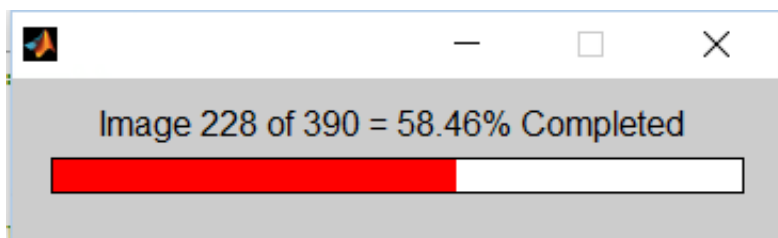
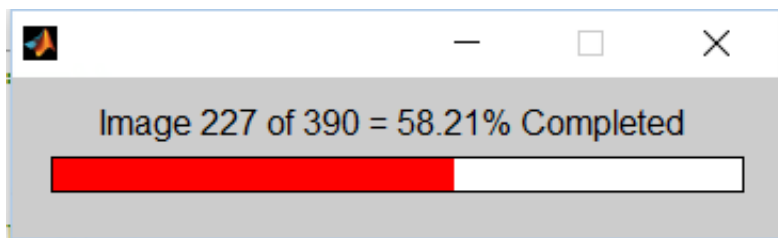
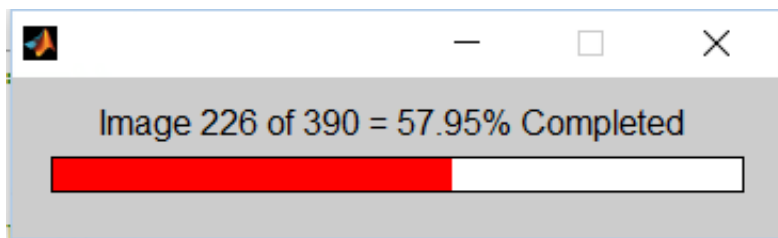
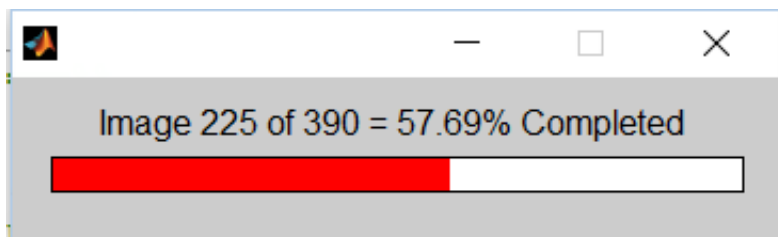
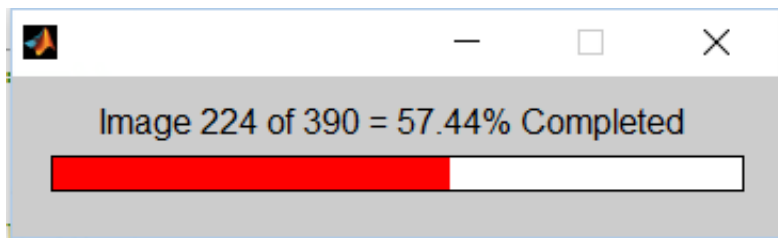
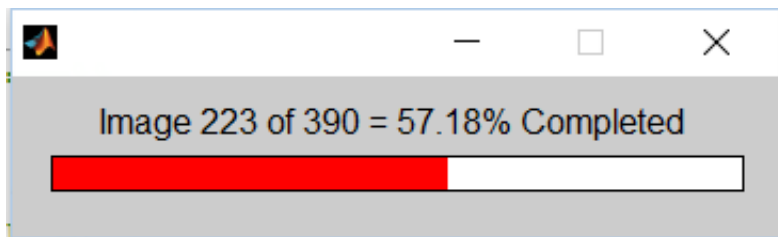


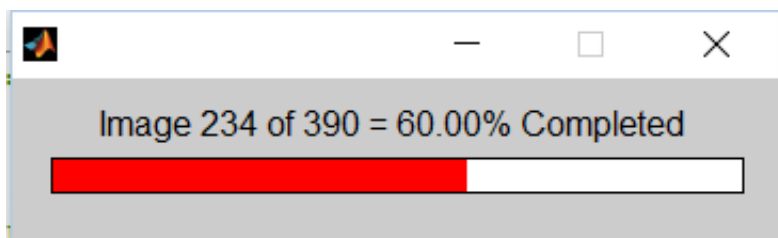
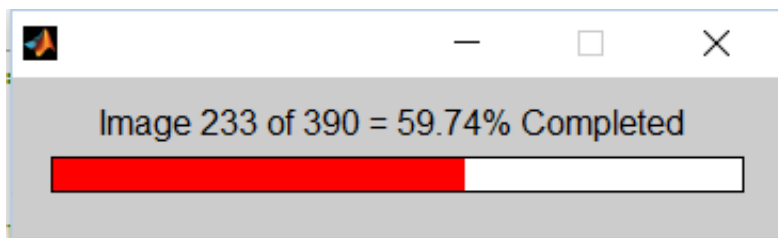
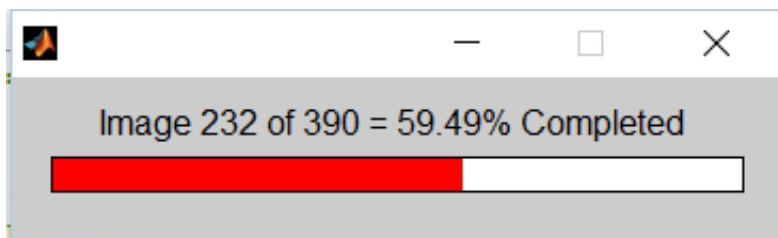
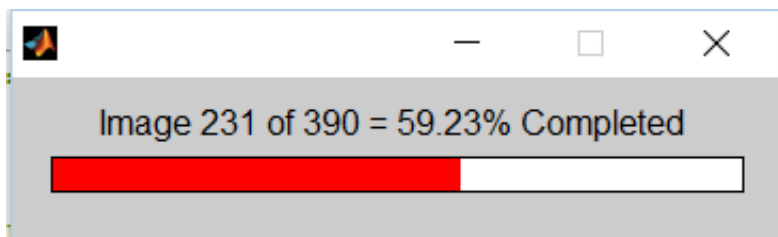
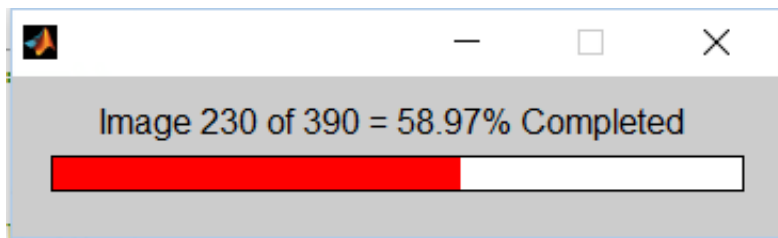
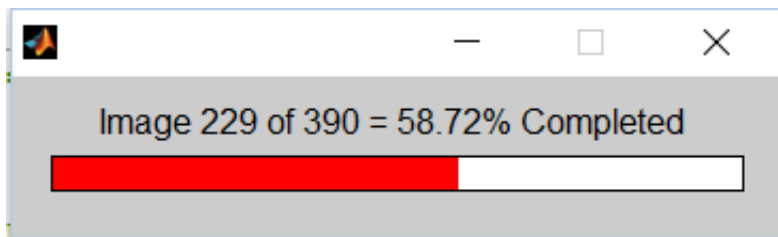


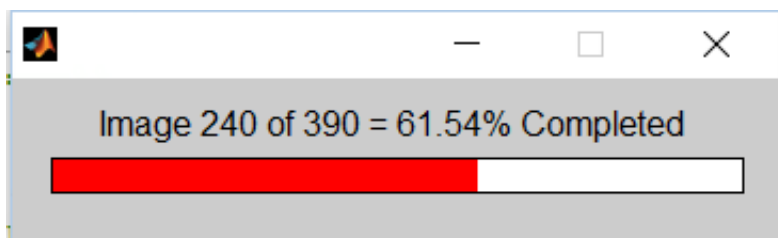
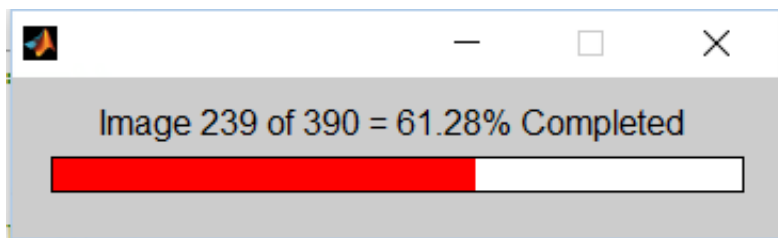
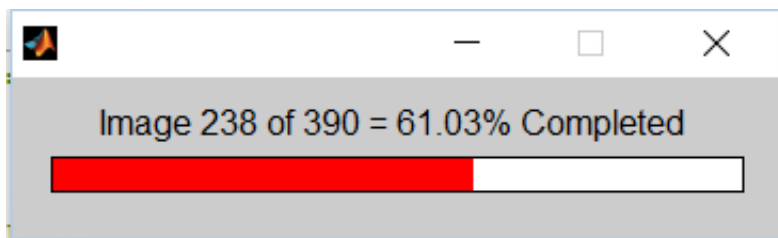
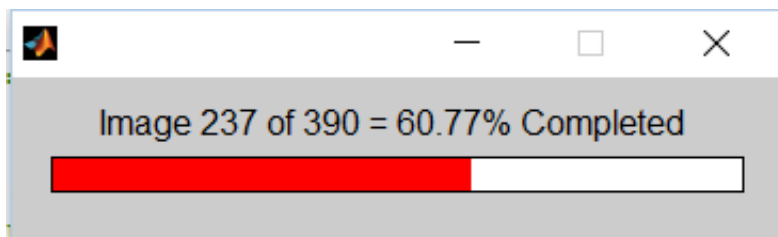
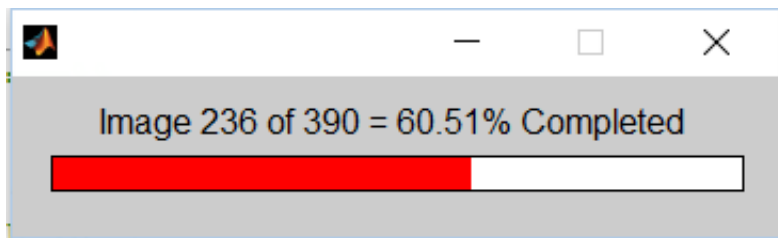
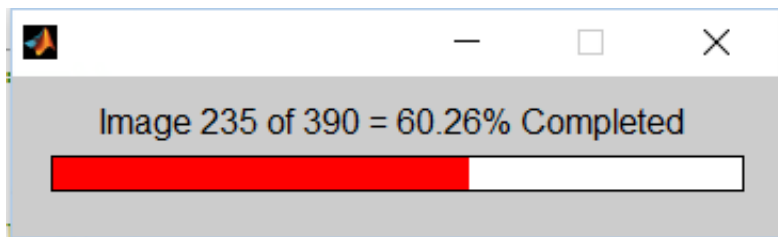


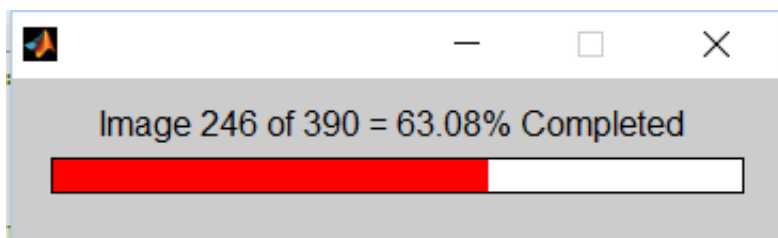
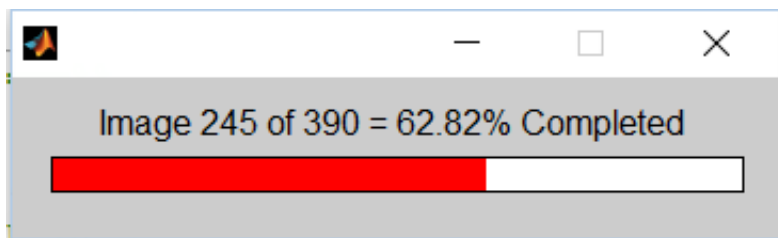
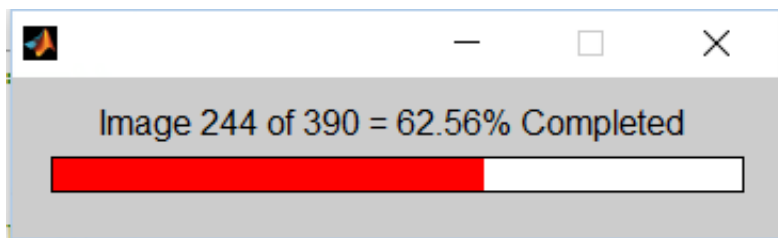
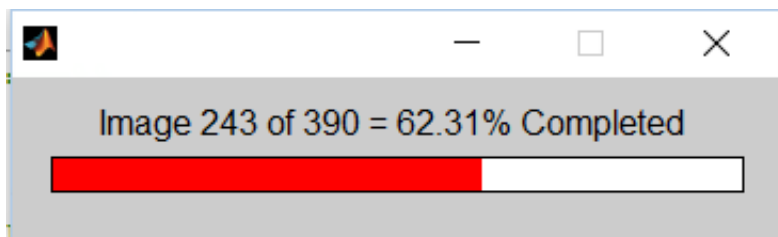
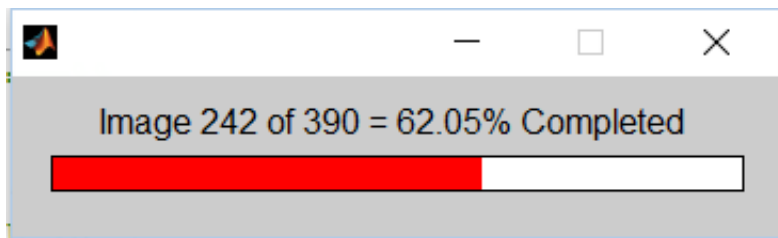
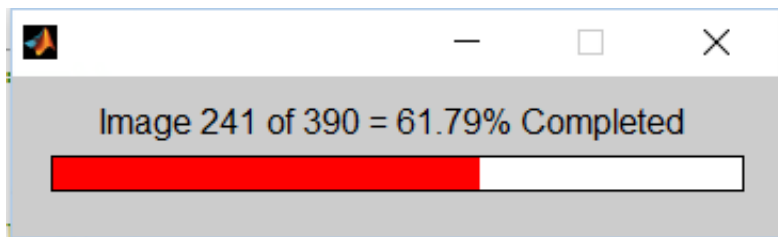


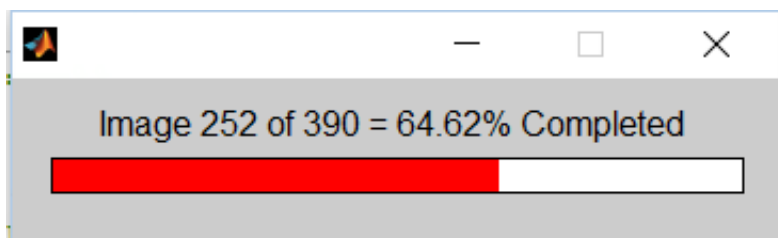
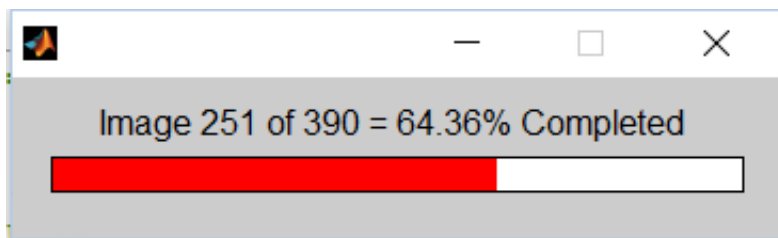
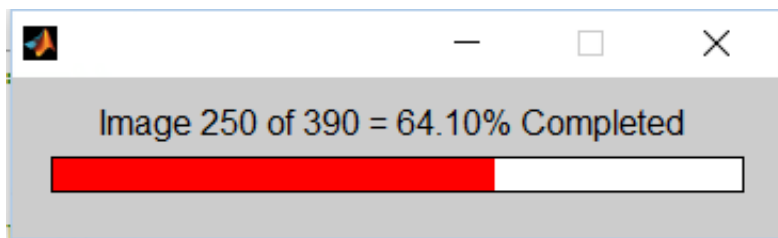
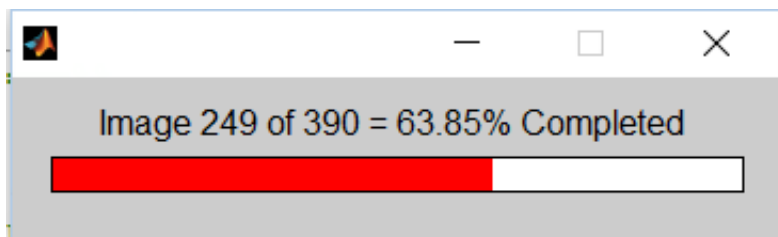
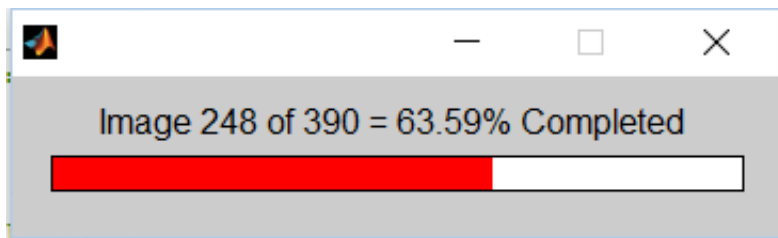
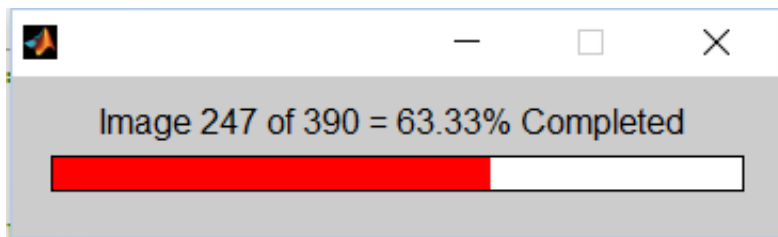


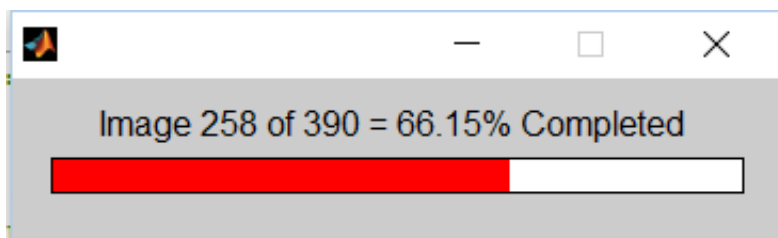
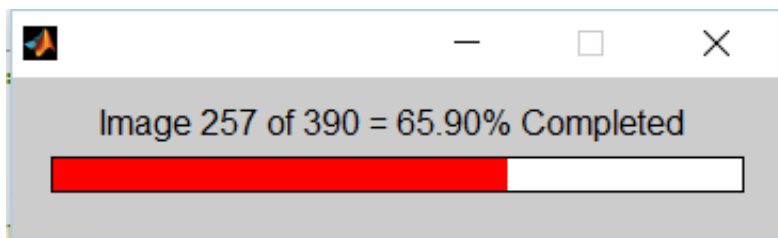
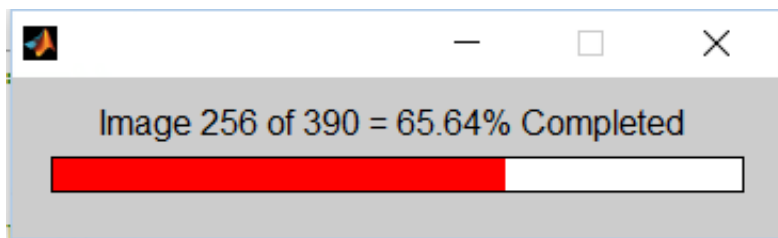
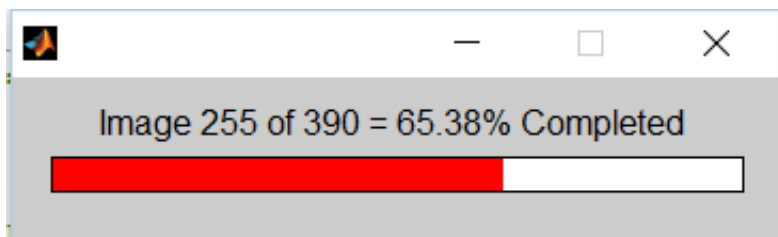
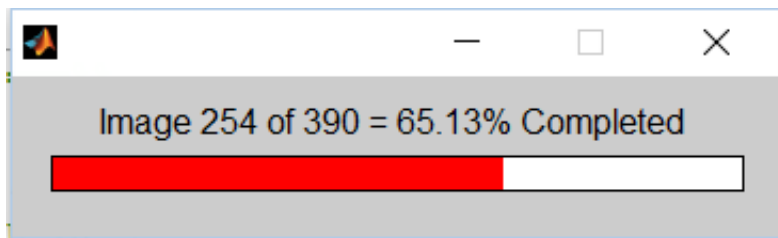
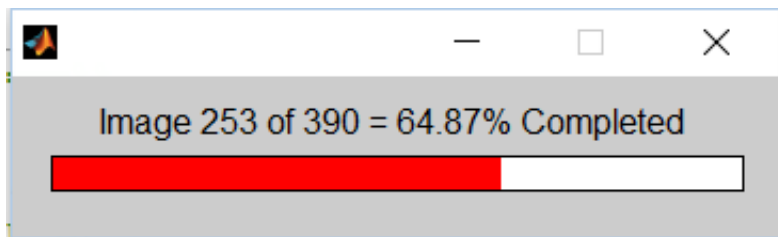


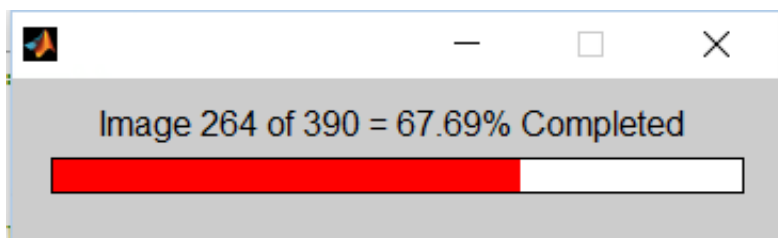
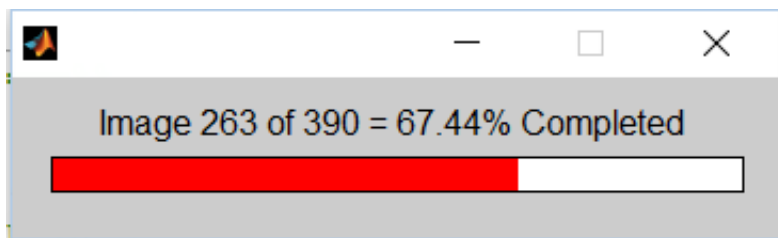
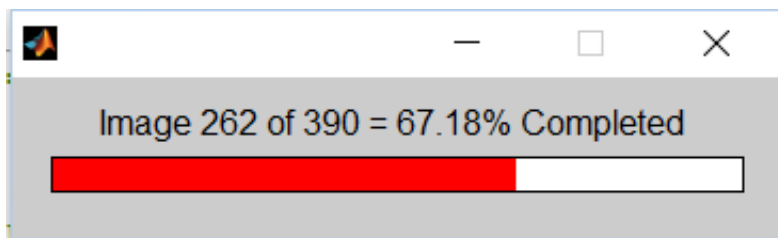
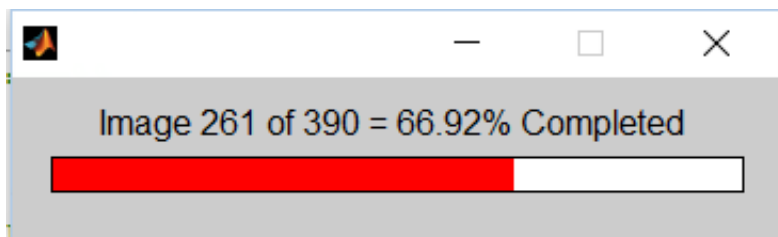
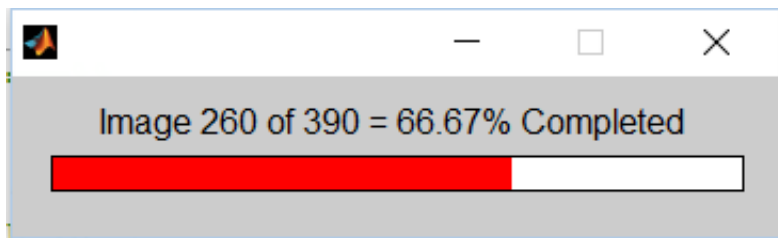
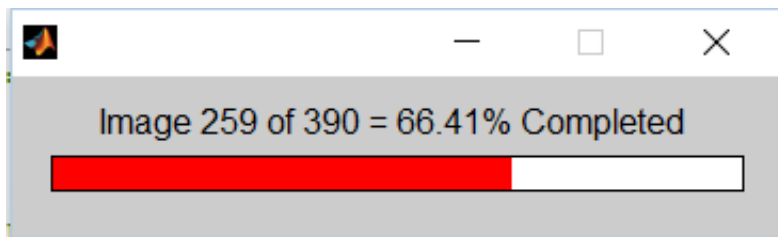




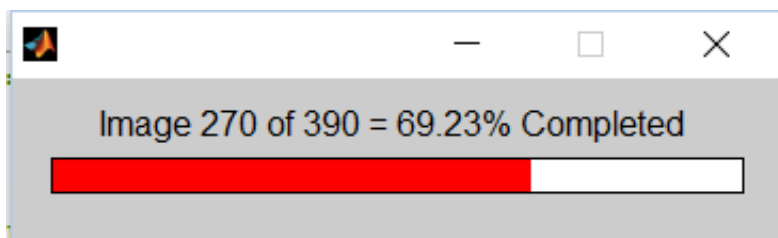
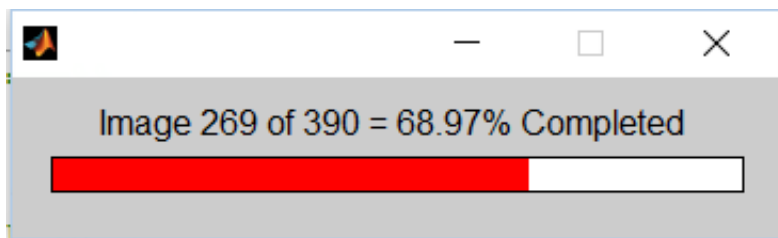
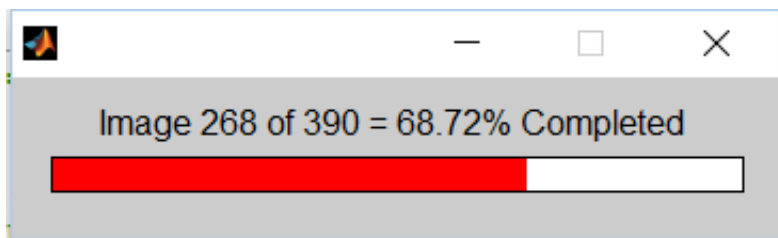
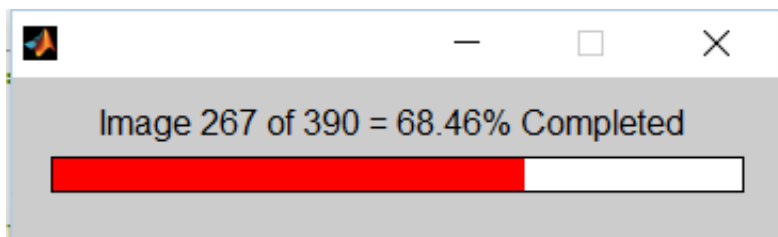
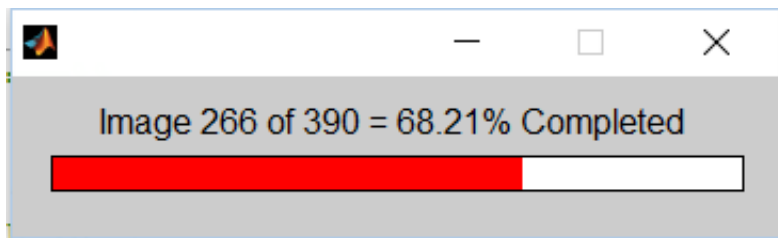
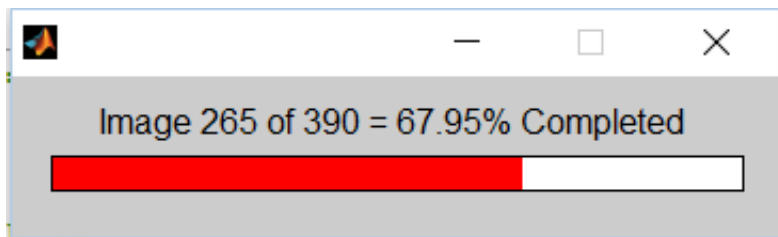


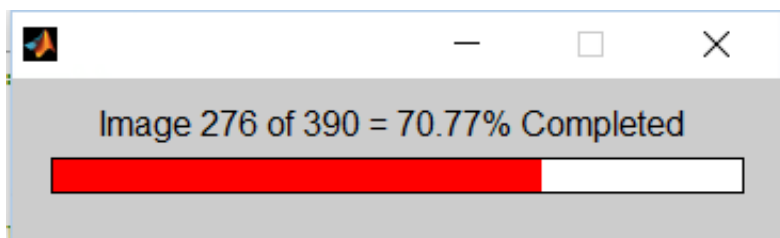
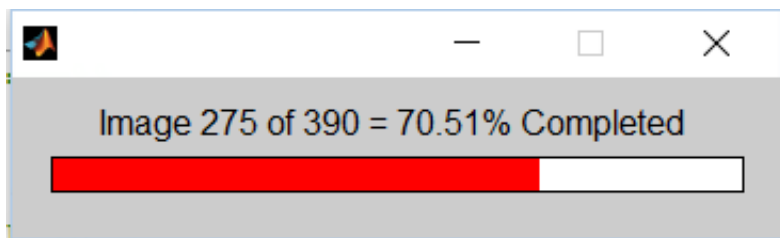
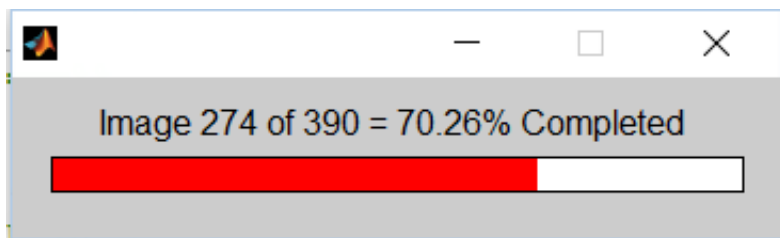
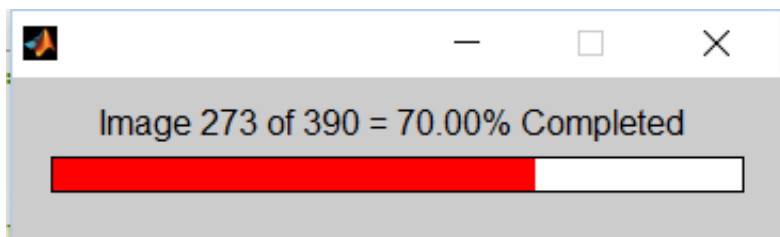
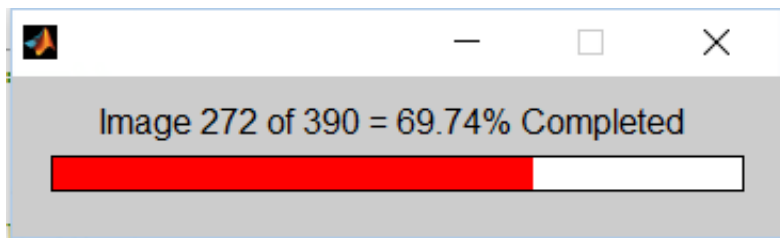
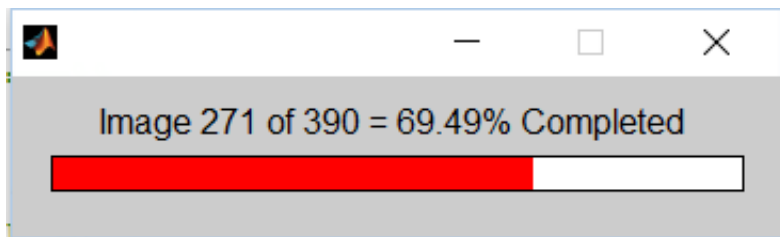


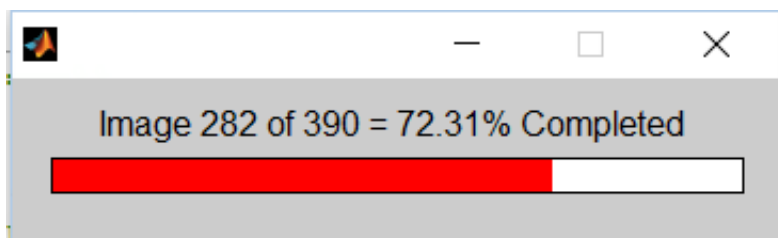
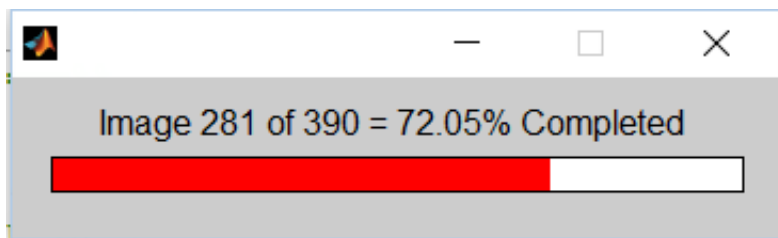
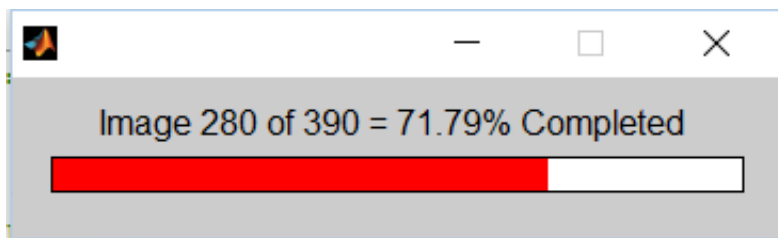
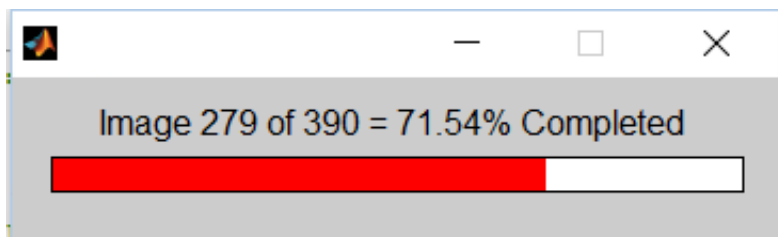
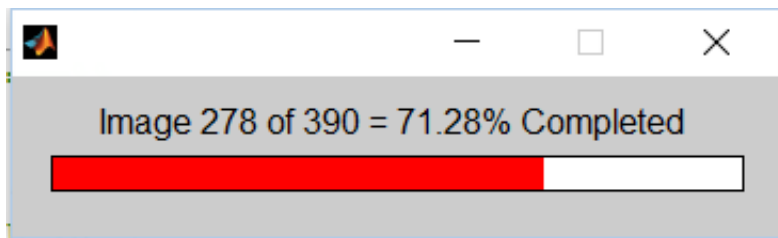
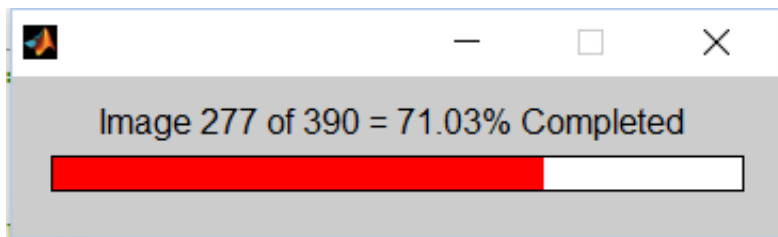


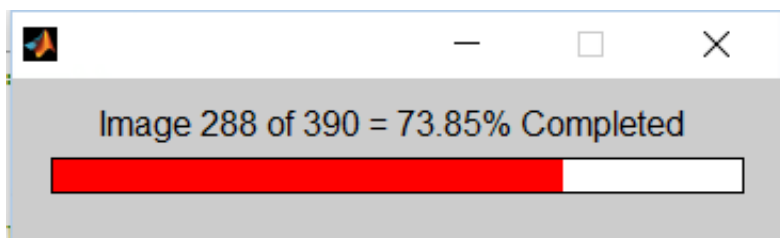
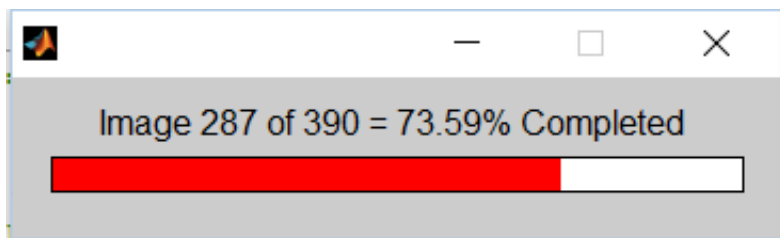
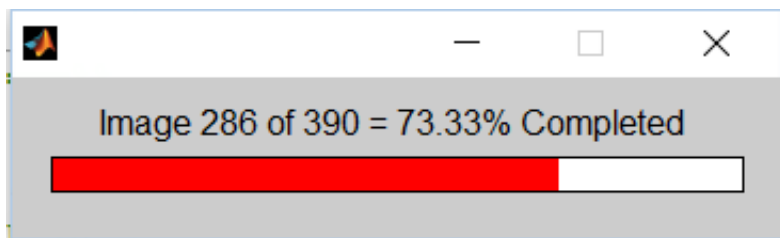
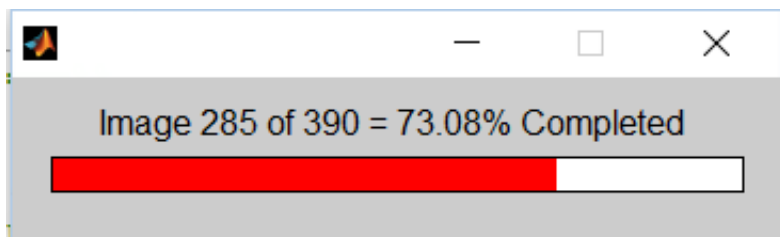
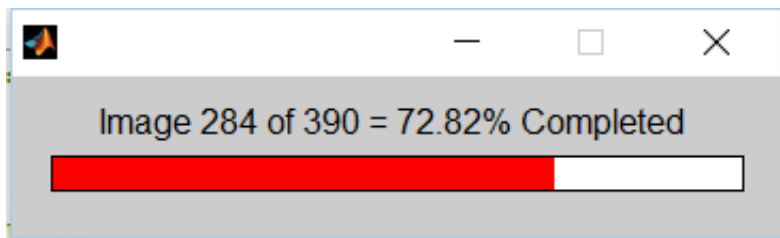
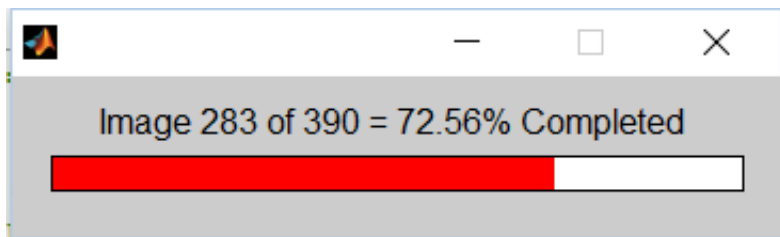


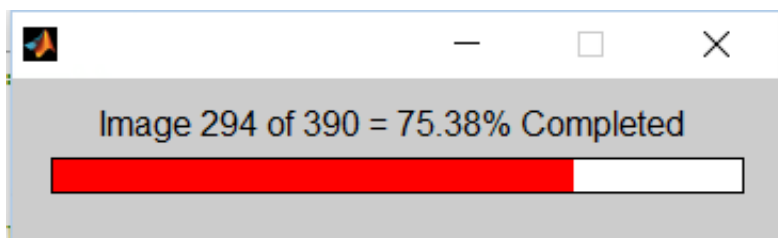
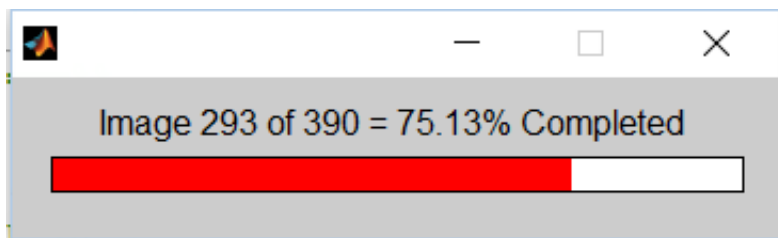
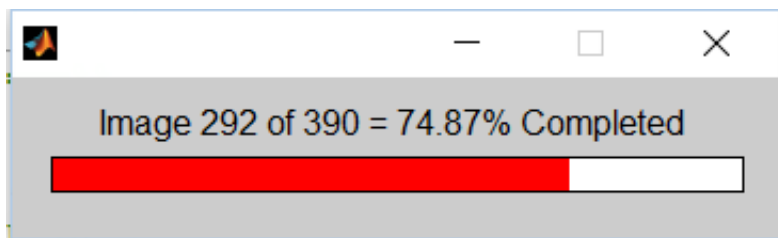
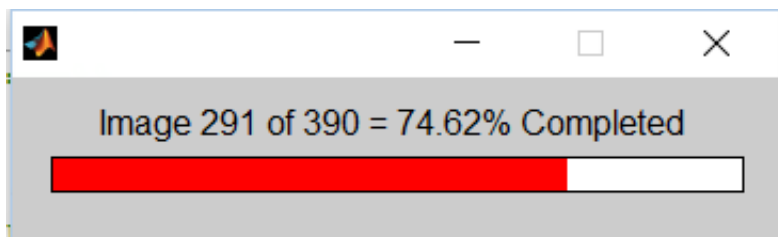
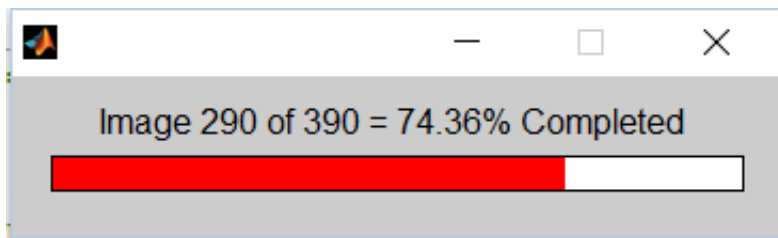
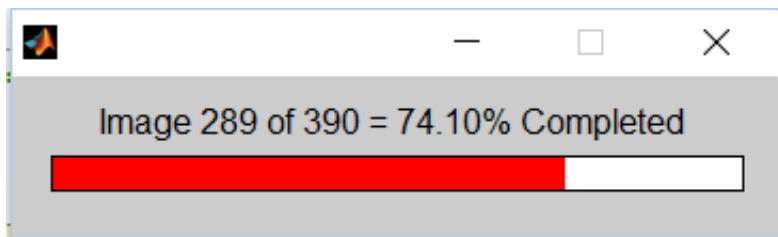


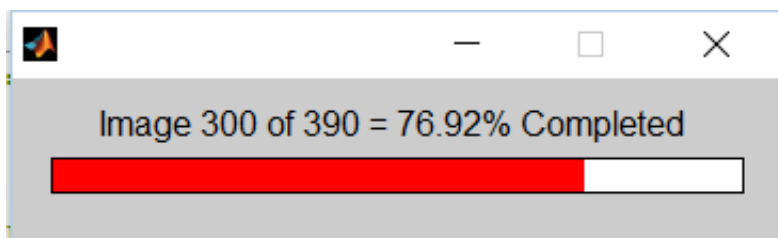
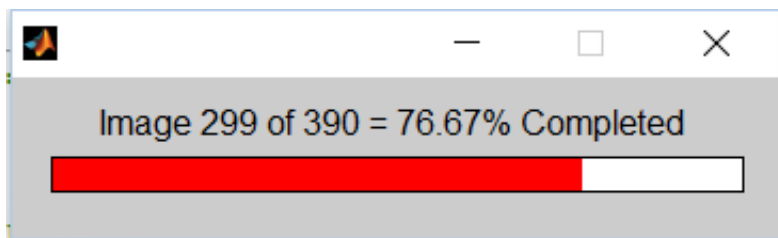
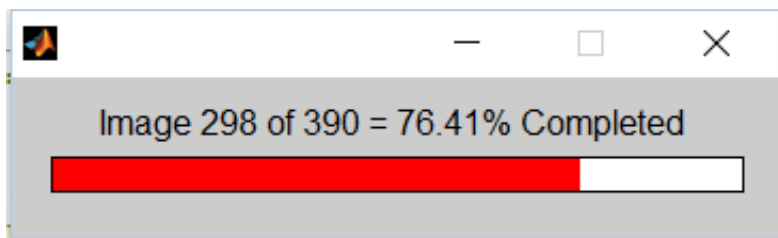
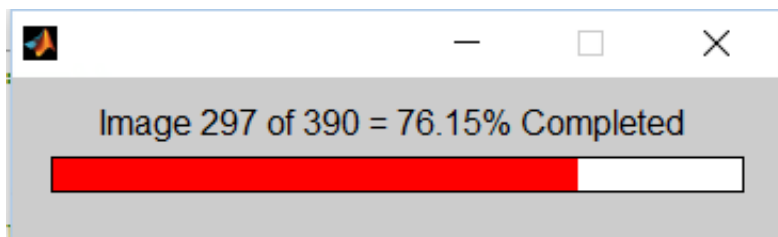
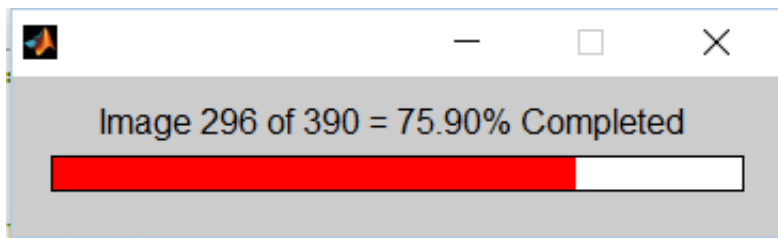
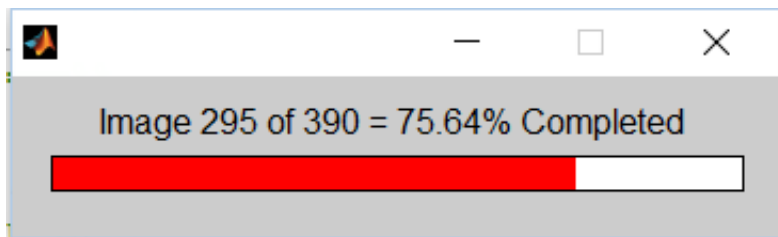


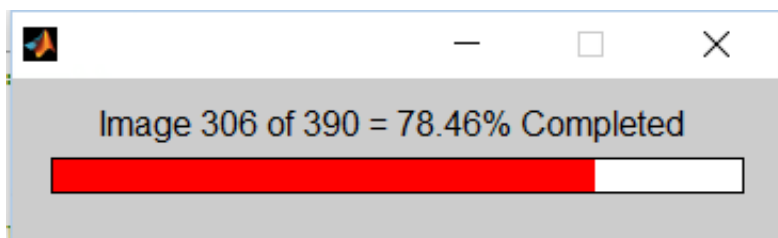
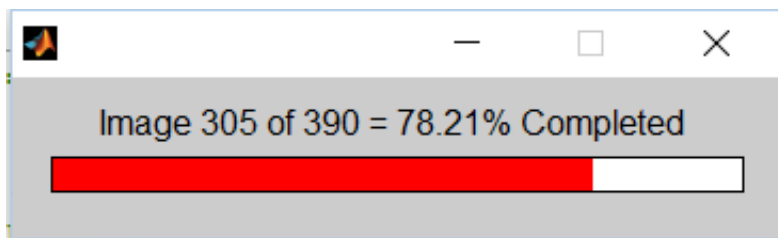
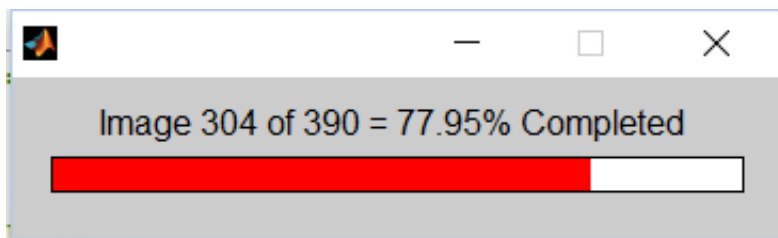
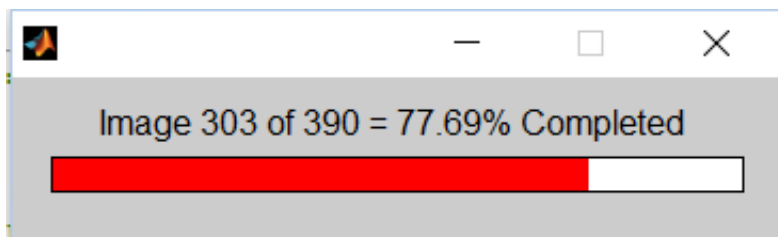
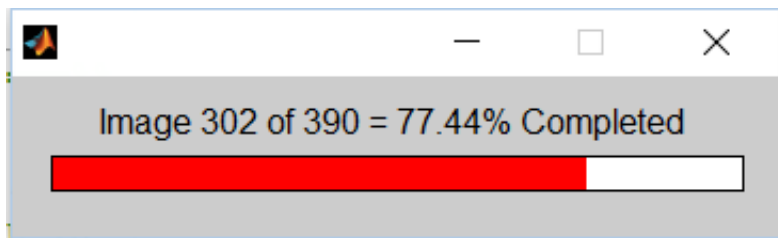
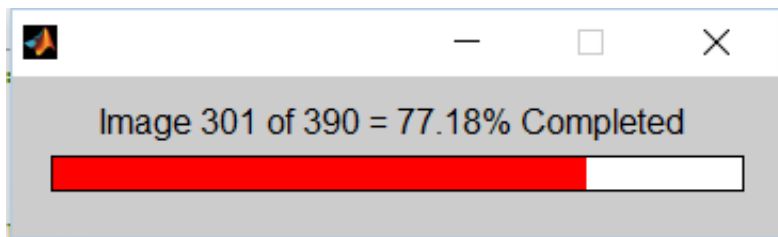


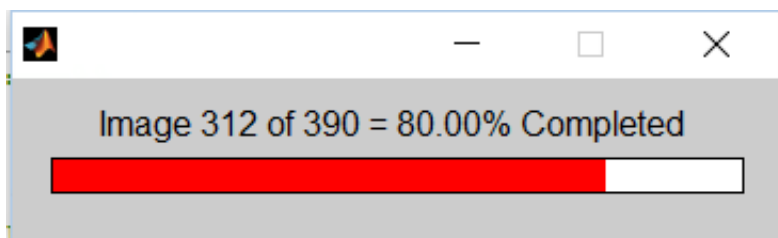
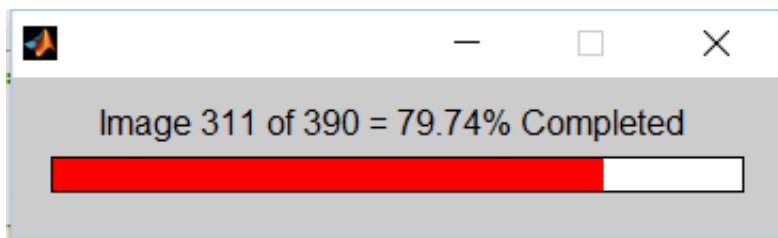
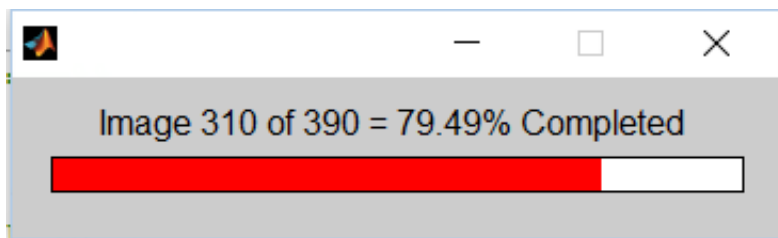
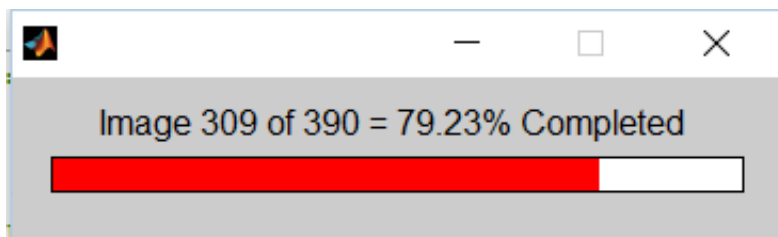
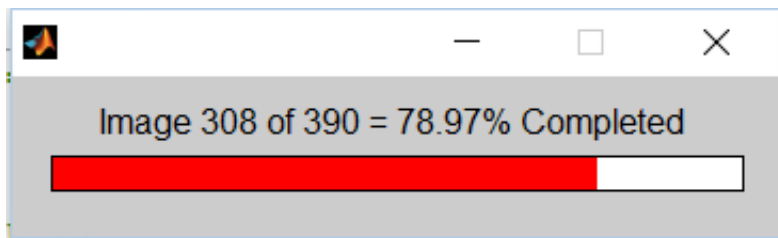
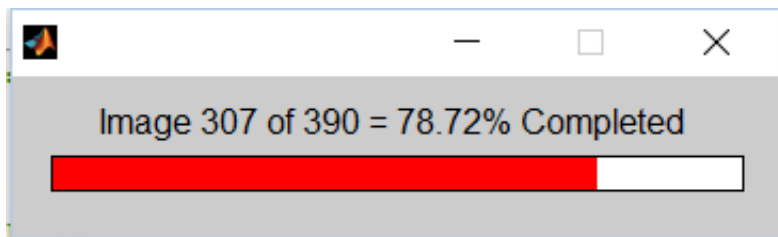




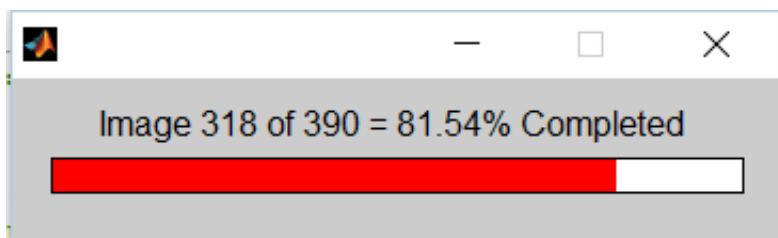
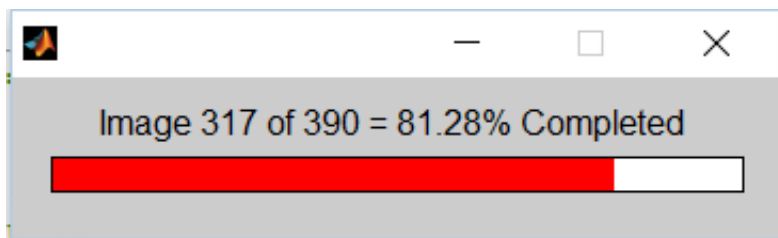
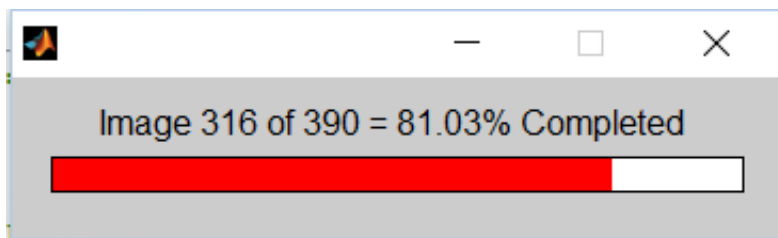
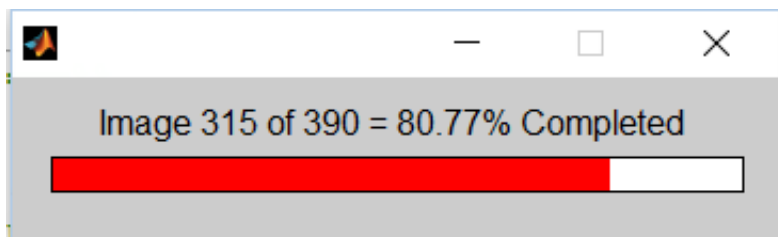
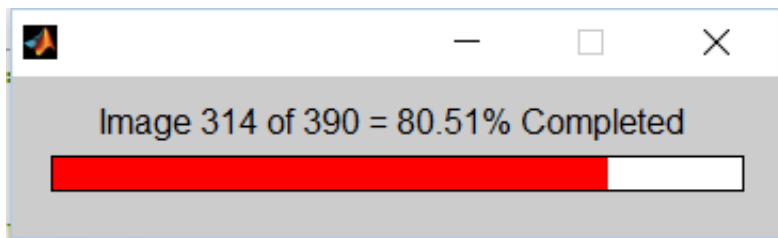
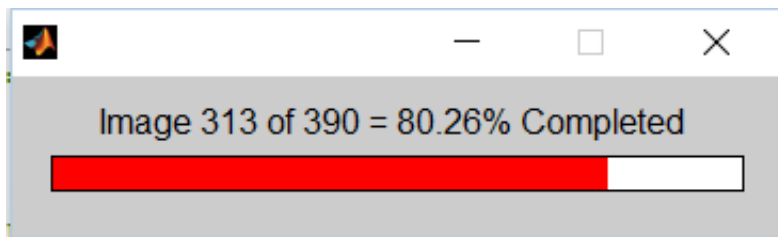


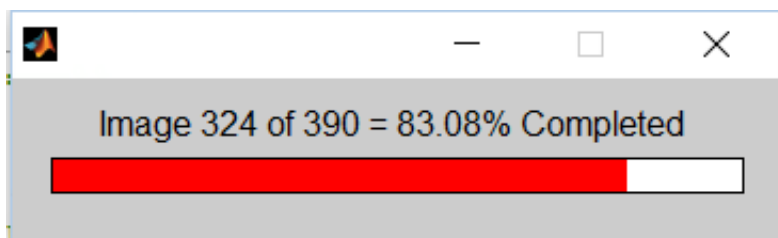
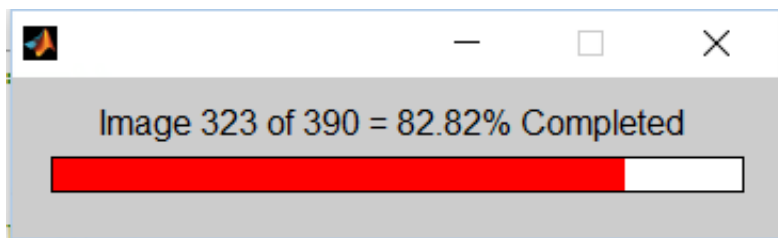
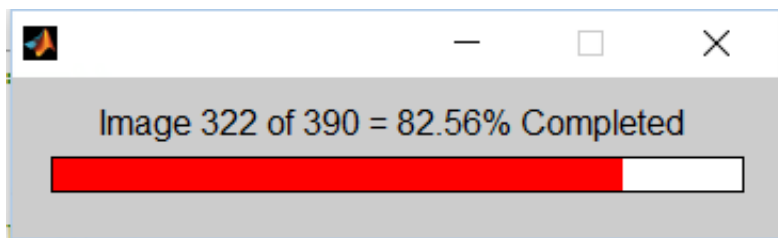
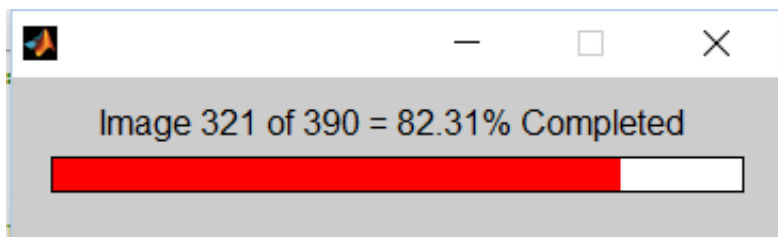
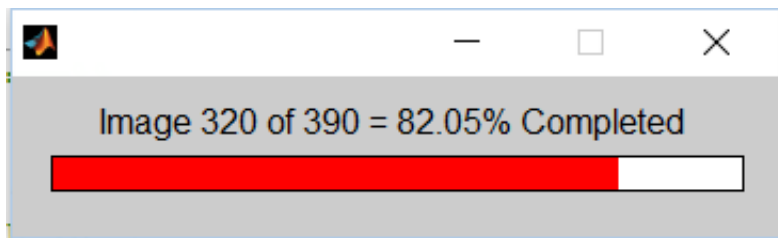
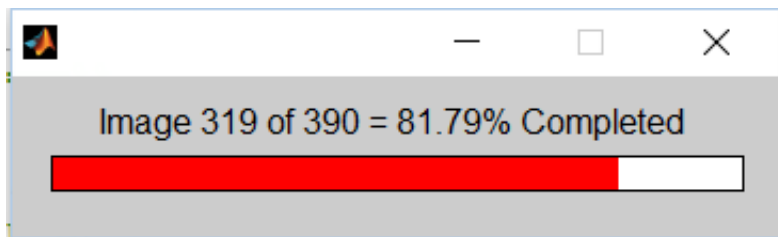


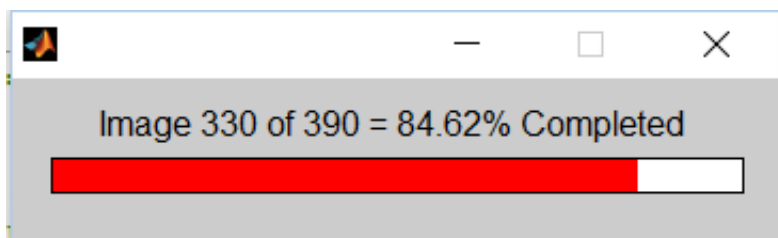
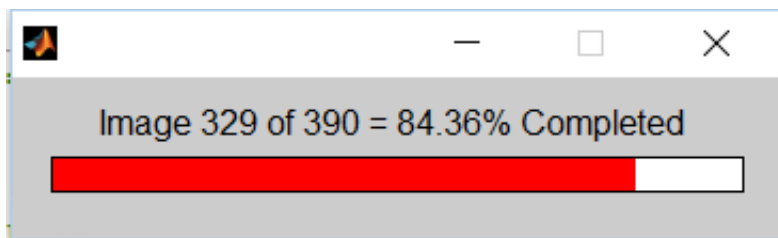
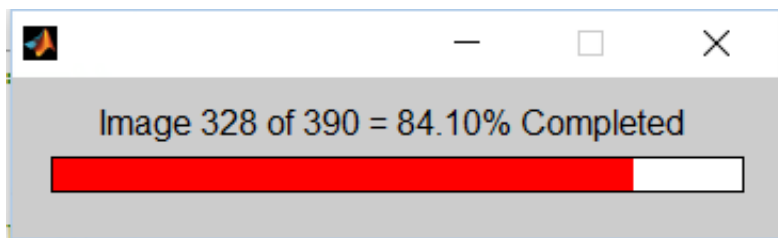
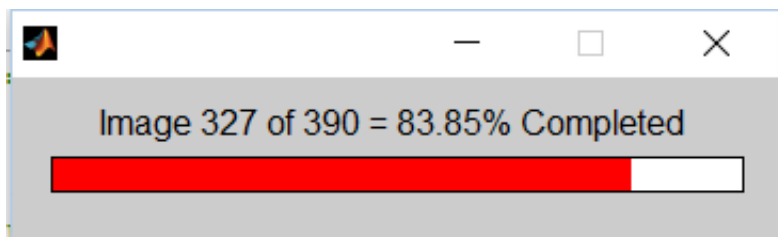
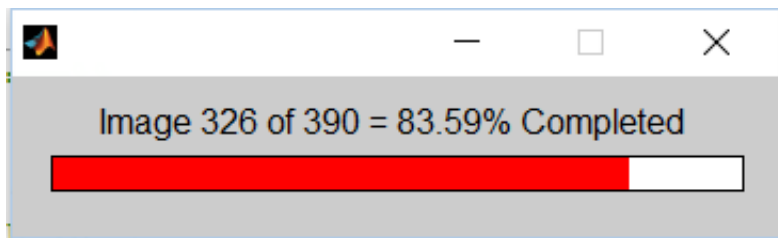
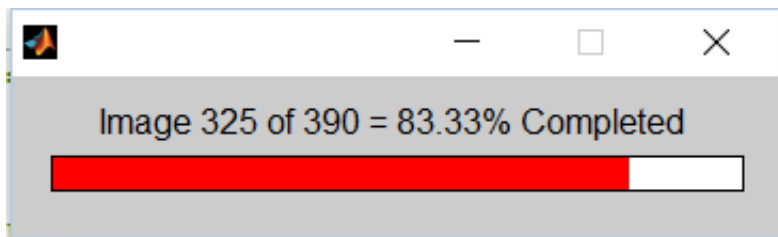


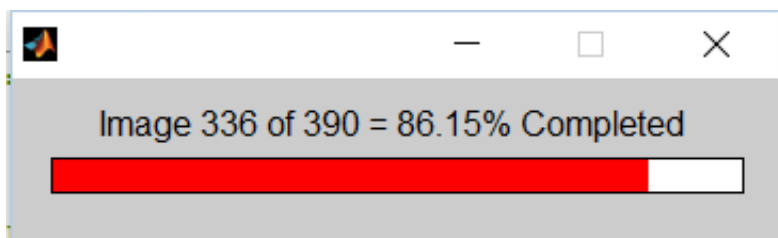
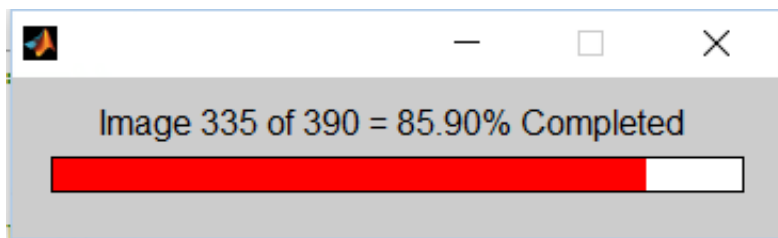
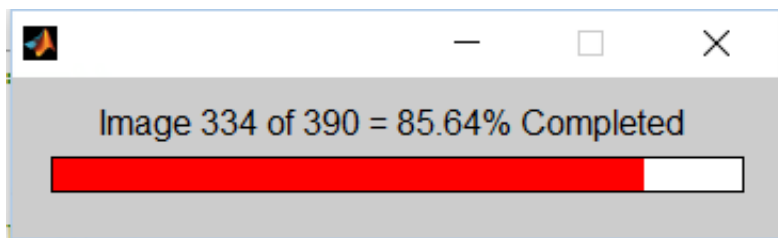
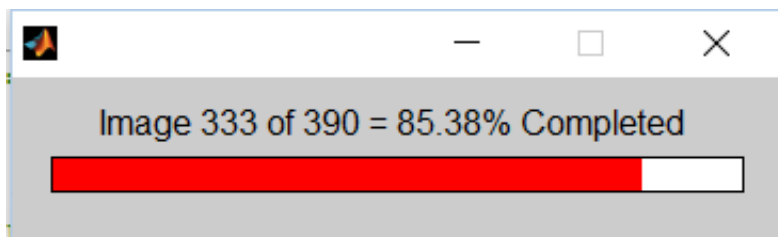
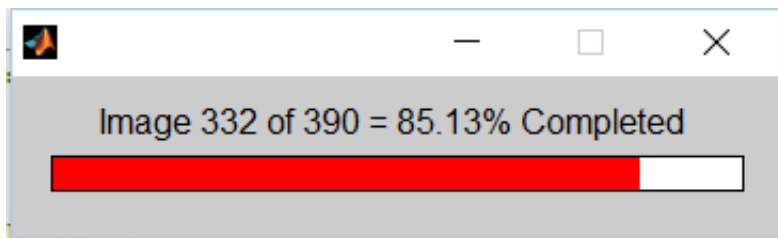
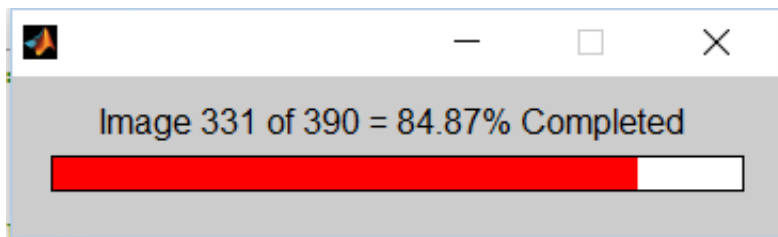


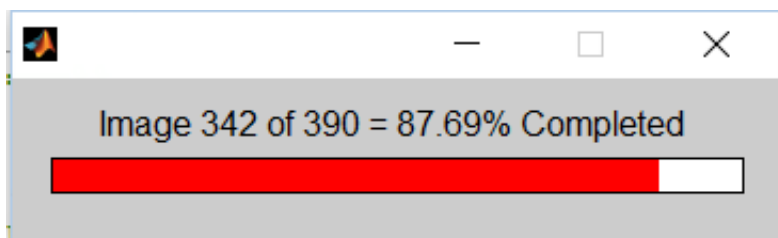
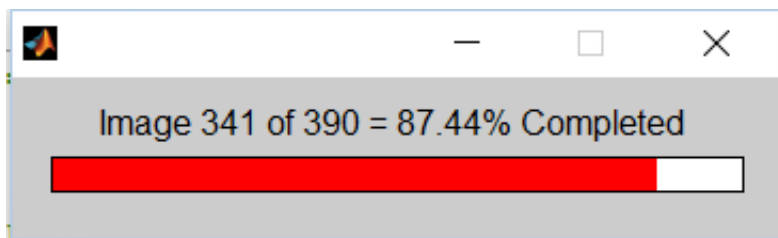
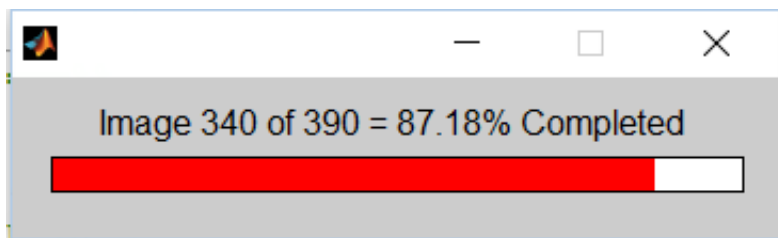
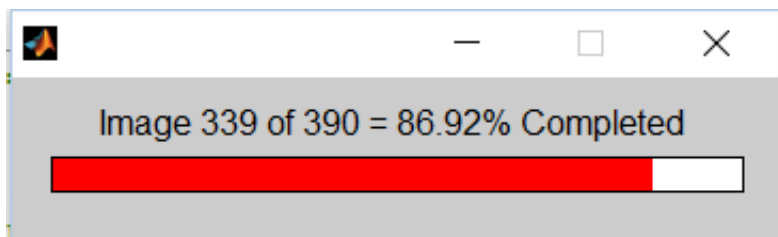
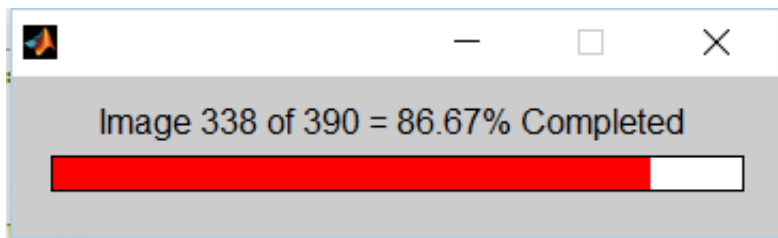
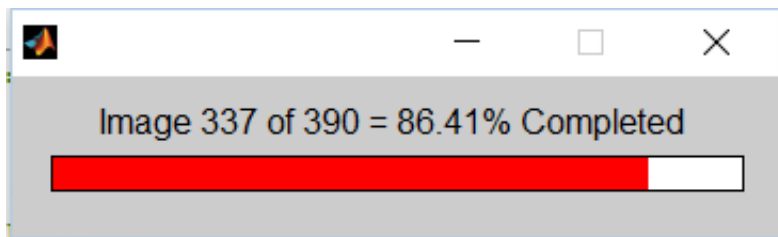


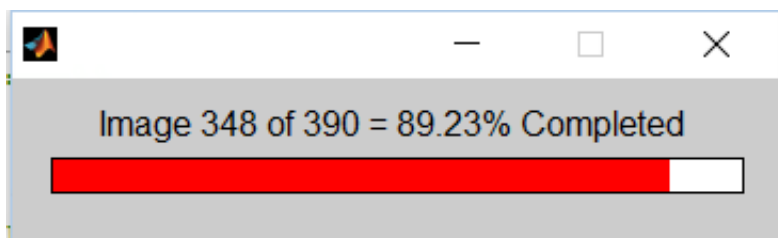
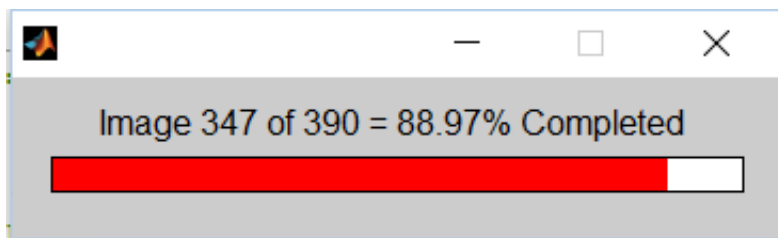
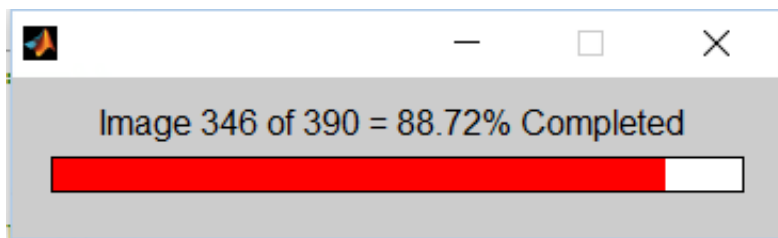
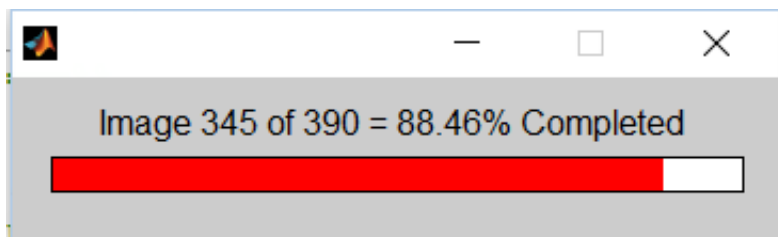
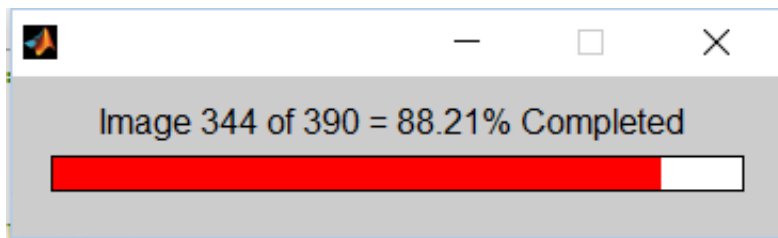
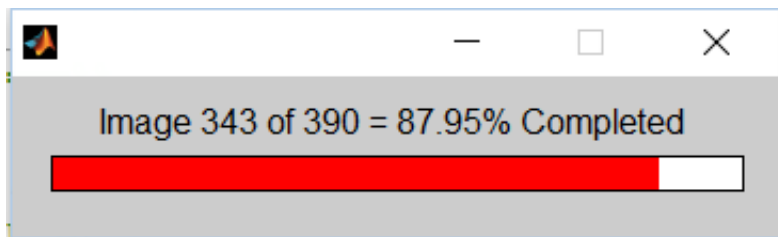


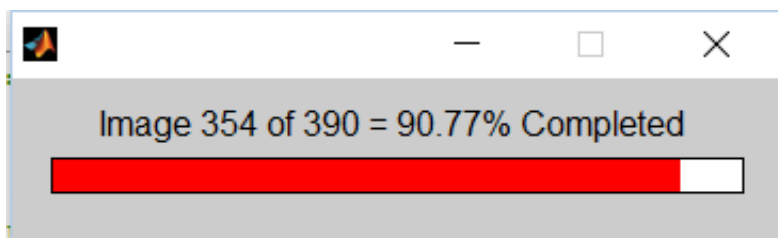
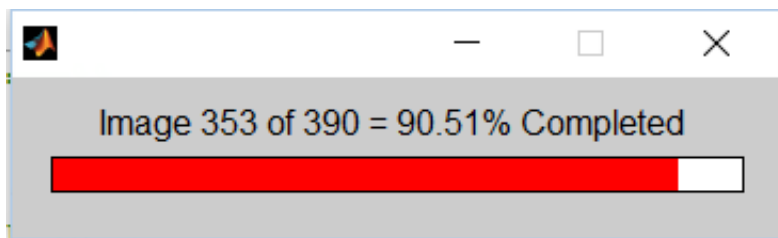
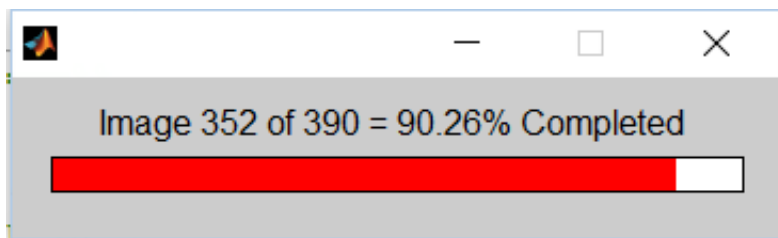
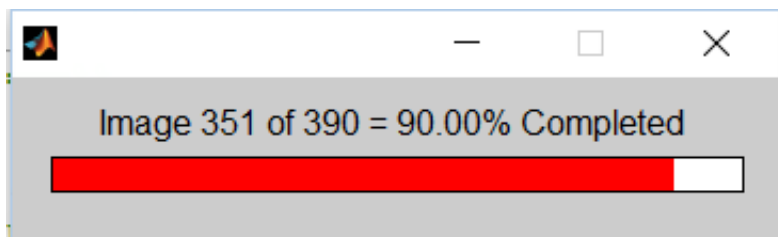
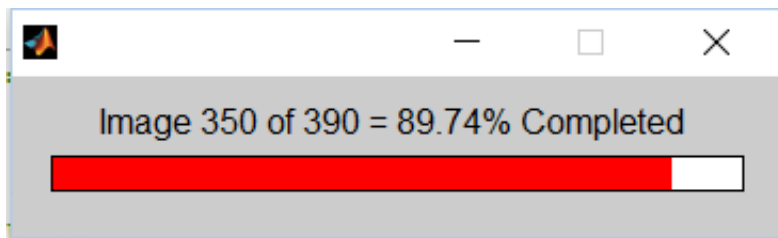
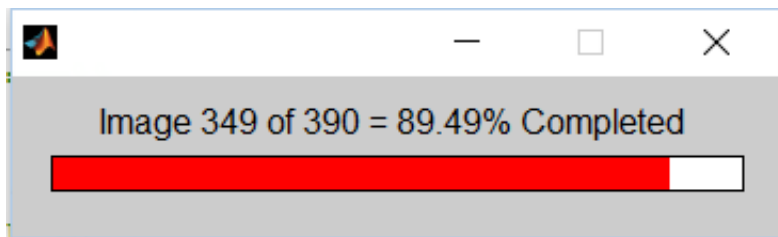


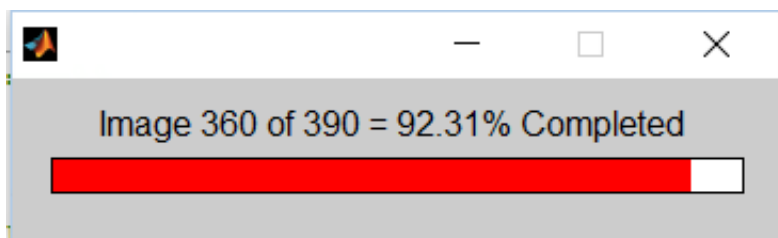
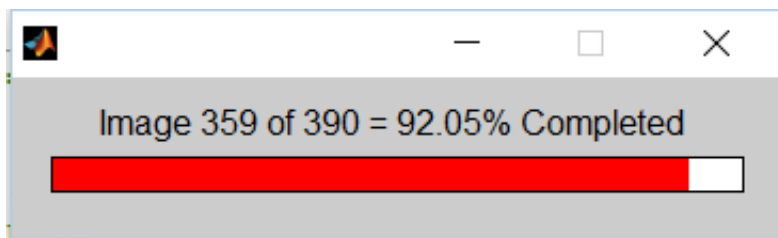
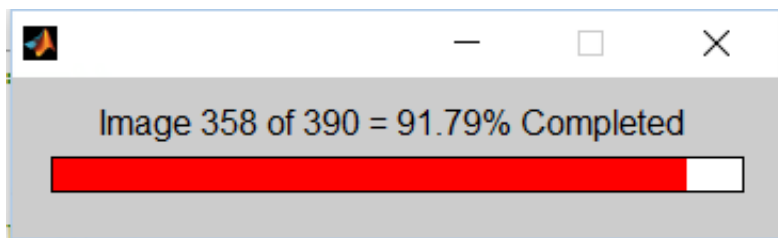
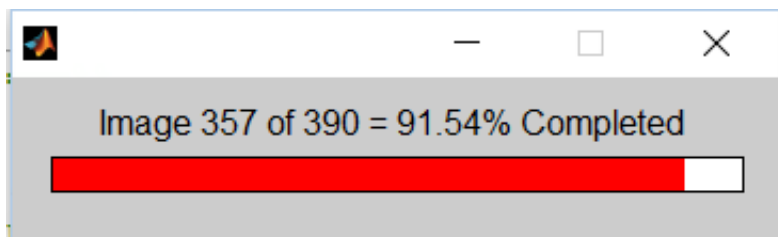
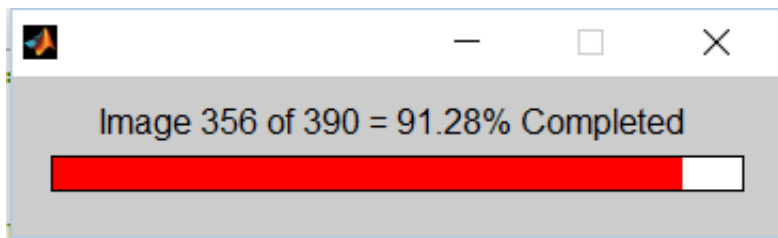
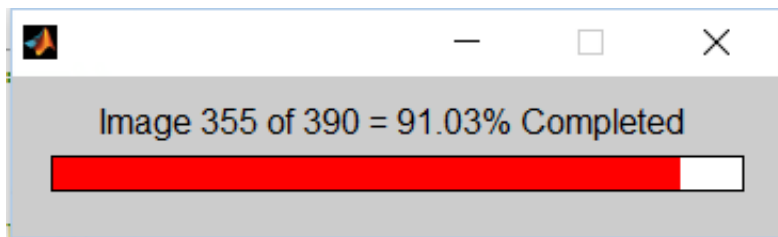




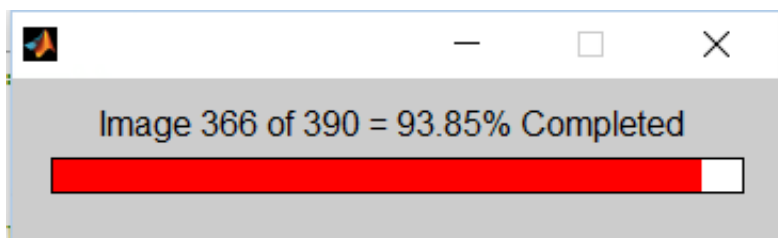
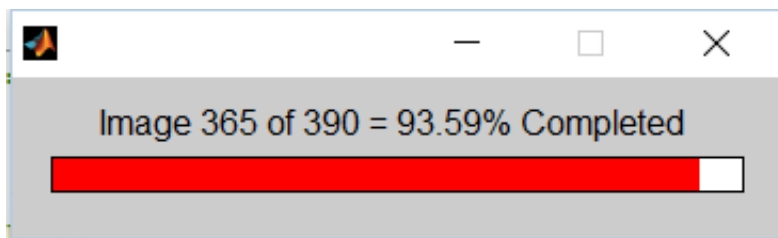
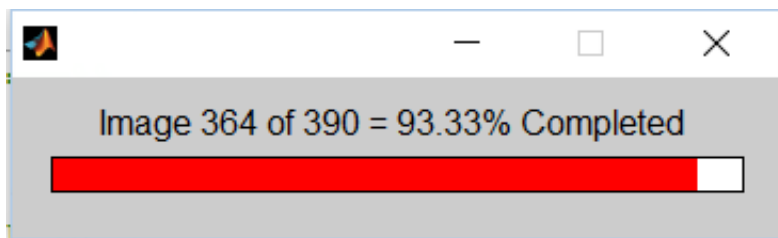
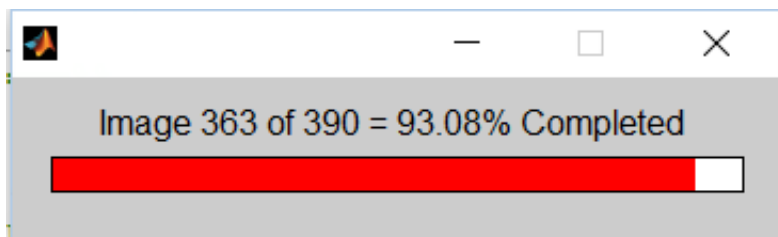
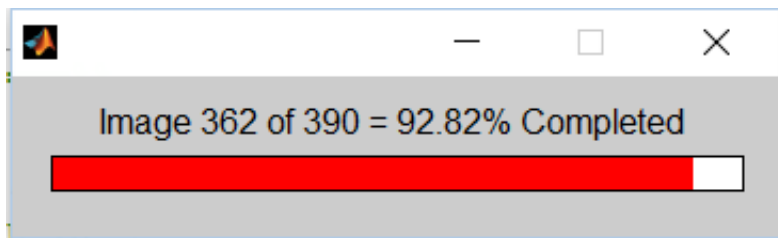
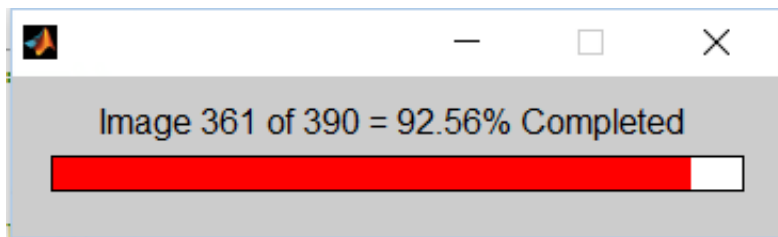


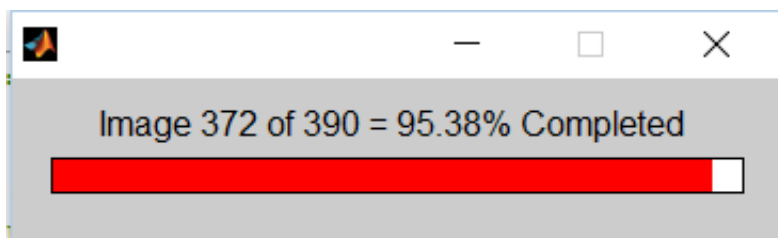
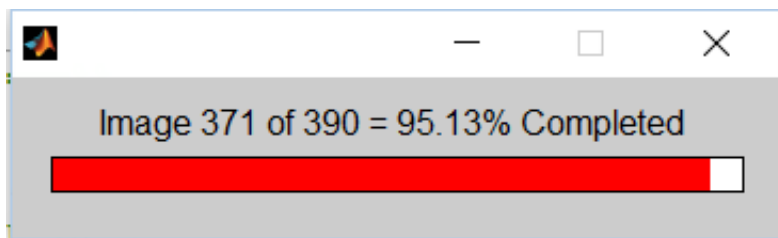
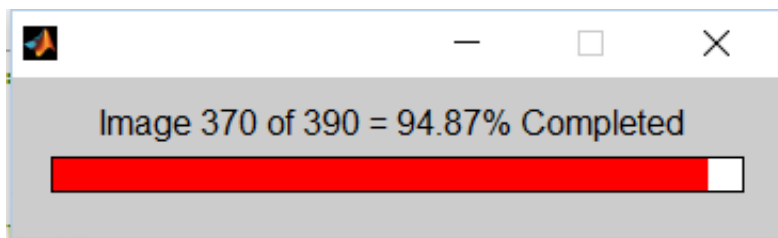
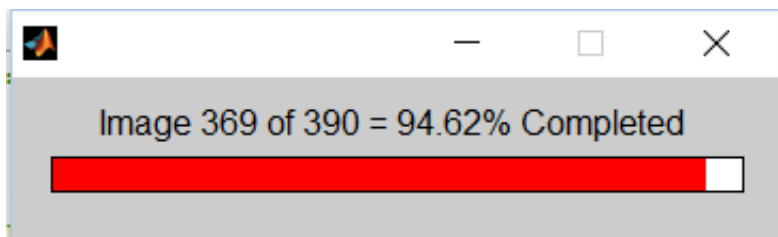
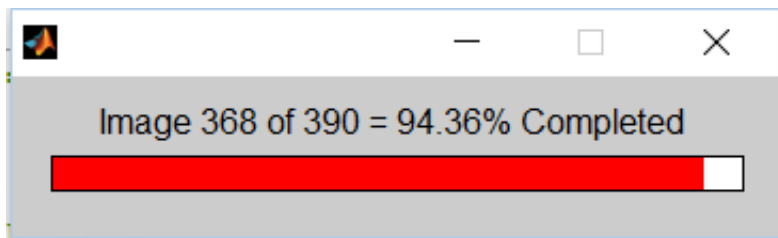
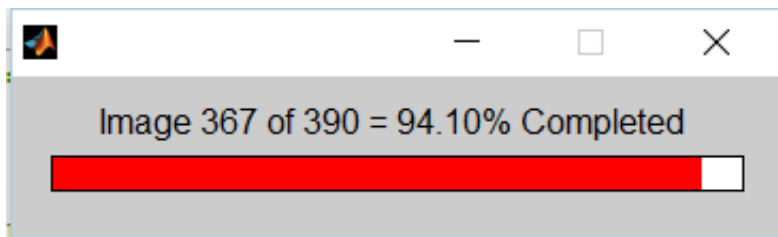


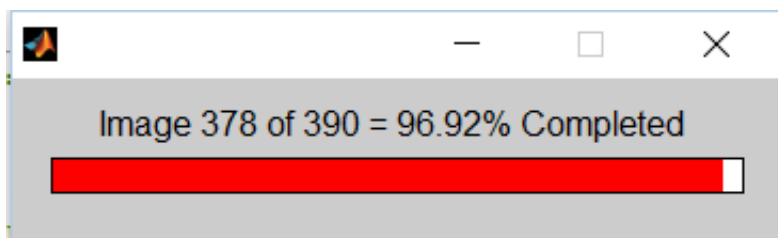
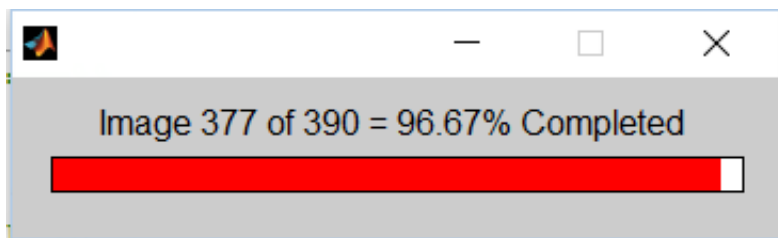
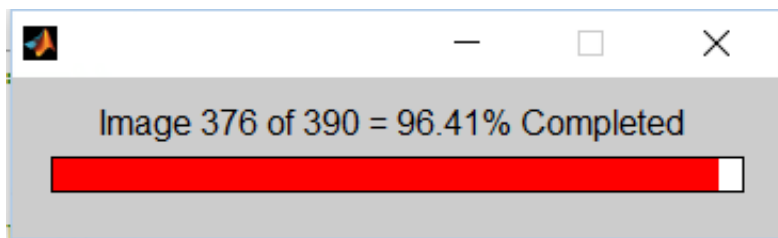
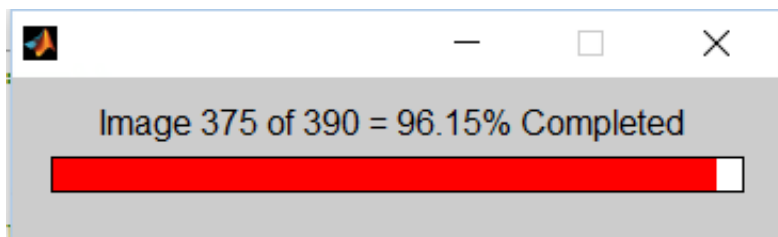
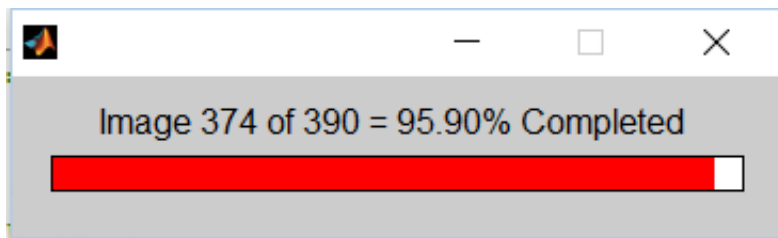
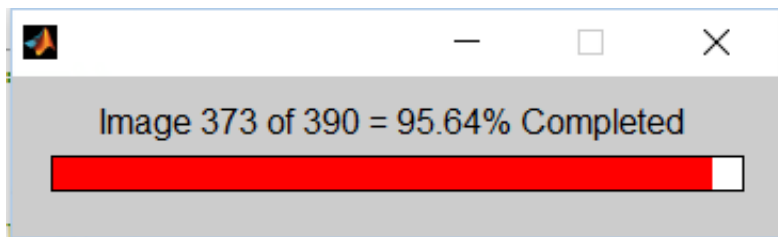


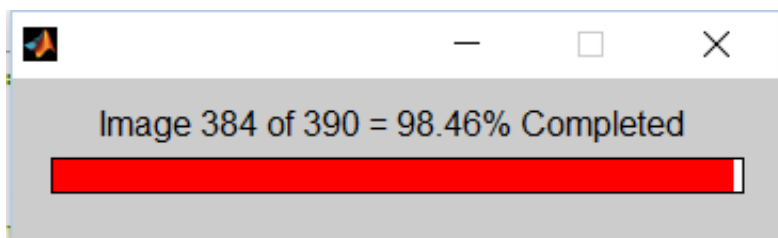
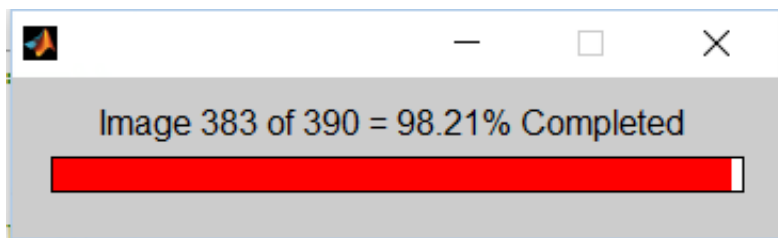
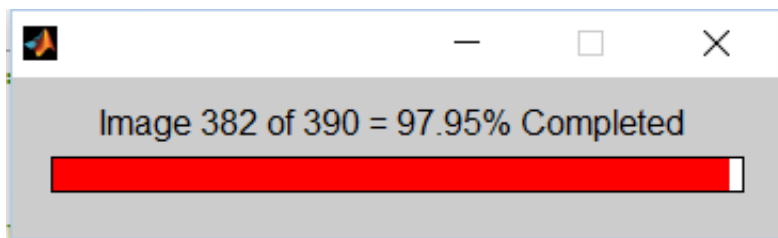
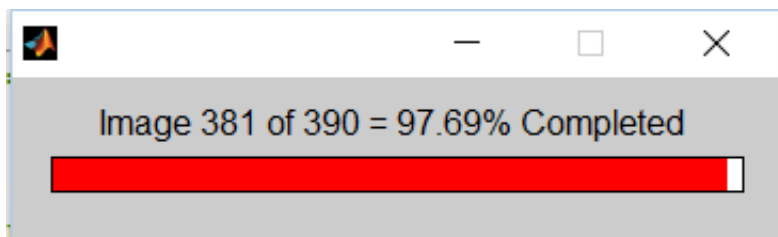
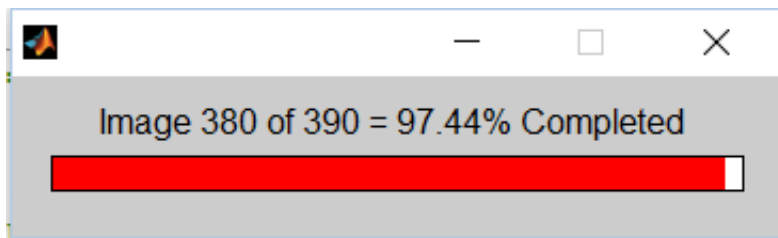
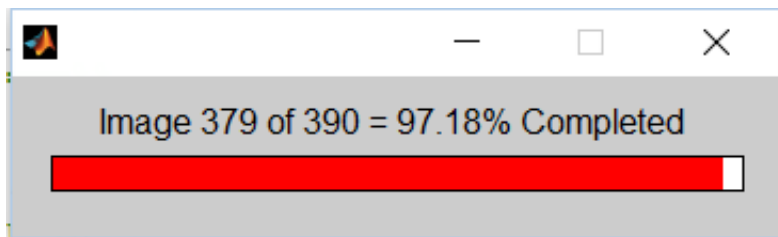


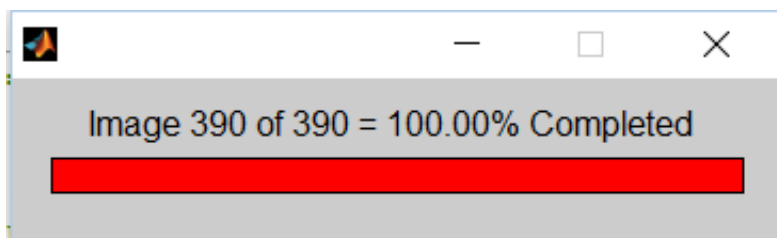
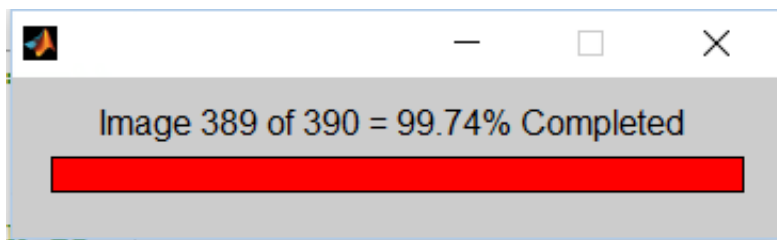
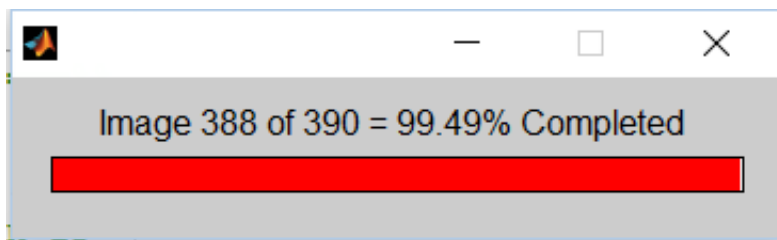
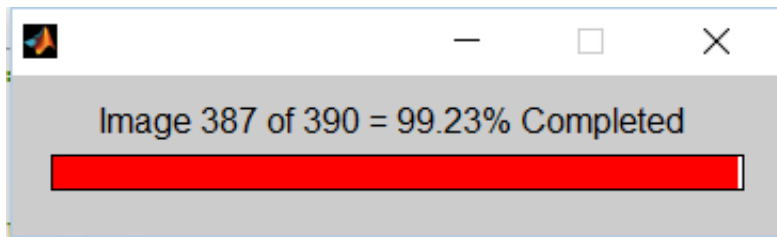
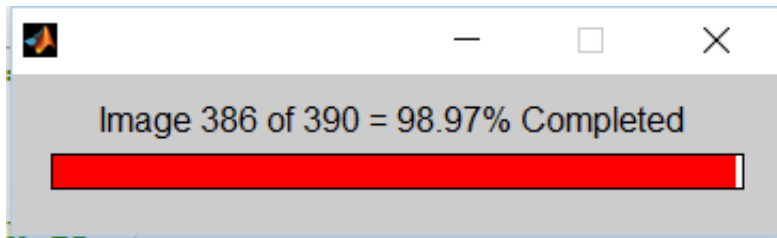
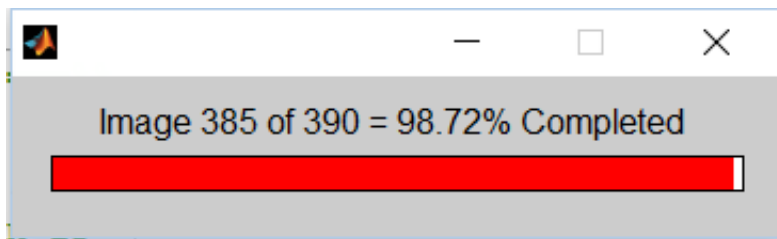












```
end
```

```
i = i+1;
```

```
j = 1;
```

end

```
rate = (hit/counter)*100;  
str = ['Detection Rate is ',num2str(rate), '%'];  
disp(str);
```

Detection Rate is 83.0769%