Loughborough University

Cover Sheet for Assessed Coursework

Name: **Sayali Avinash Chavan**

ID Number: **B812081**

Module Title: **Computer Vision & Embedded Systems**

Module Code: **18COP507**

Degree Programme: **Advanced Computer Science** Year/Part: **2018/T**

Coursework Title/Description: **Vehicle Make & Model Recognition**

Staff Member responsible: **Prof. Eran Edirisinghe**

Deadline for Submission: **11-1-2019**

If this coursework was part of a group activity, list the names of the other group members:

NA

DECLARATION:

I certify that the attached coursework is my own work and that anything taken from or based upon the work of others has its source clearly and explicitly cited.

Signature:

Coursework received by:.....(Signature)

Date received:

Index

Introduction

The proposed system is Vehicle Make and Model Recognition (VMMR) to detect a Vehicle's make and model based on a frontal view of the vehicle. The image includes the grill, headlights, bumper, the license plate are grayed out due to security reasons. The main reason to implement this system is added security along with the number plate recognition in real world as the cloning of number plate theft is increasing nowadays, so to make it more secure the VMMR gives one extra level of security. The given system can be implemented using various methods like Computer Vision, Machine Learning, Deep Learning- CNN etc.

For this project I have chosen the Computer Vision approach which works on feature extraction from given image and compare it with each image in database and show the best match output with highest matching points using **SURF** detector implemented using MATLAB.

Requirements Specification

1. Design and implement a VMMR system that can recognise a vehicle's make and model based on a frontal view of the vehicle. From the database of various testing and train images. Select one image as Input of the frontal view of a vehicle and it should give output as the vehicle-make and model image along with its name.
2. Also, design VMMR system with a simple Convolutional Neural Network (CNN) along with a schematic diagram, its explanation of each layer in the CNN, discuss three strategies to prevent overfitting and concept of Transfer Learning.

Design

Main part of Design was to select the best feature detection Algorithm. For this experiment I have selected SURF function. Essentially, it is extracting the feature and then finding the most number of matched points between potential matching pairs and is looking for the minimum error match.

- **pre-processing steps:**
  Before selecting any image for detection there are some pre-processing steps performed as below:
  The Database has 625 images of 25 different make and models of the frontal view of a vehicle which includes the grill, headlights lights, bumper and the license plate is greyed out with exact crop function to reduce any interference in the image detection and feature matching. Randomly 10 images of each module are selected out of 25 for testing in another folder. The training folder contains 375 images. Each image was renamed to its vehicle name to confirm the result.

- **System Block Diagram:**
  The process flow starts from Test image block where user can select any image from test Dataset and the arrows indicate the process flow direction.

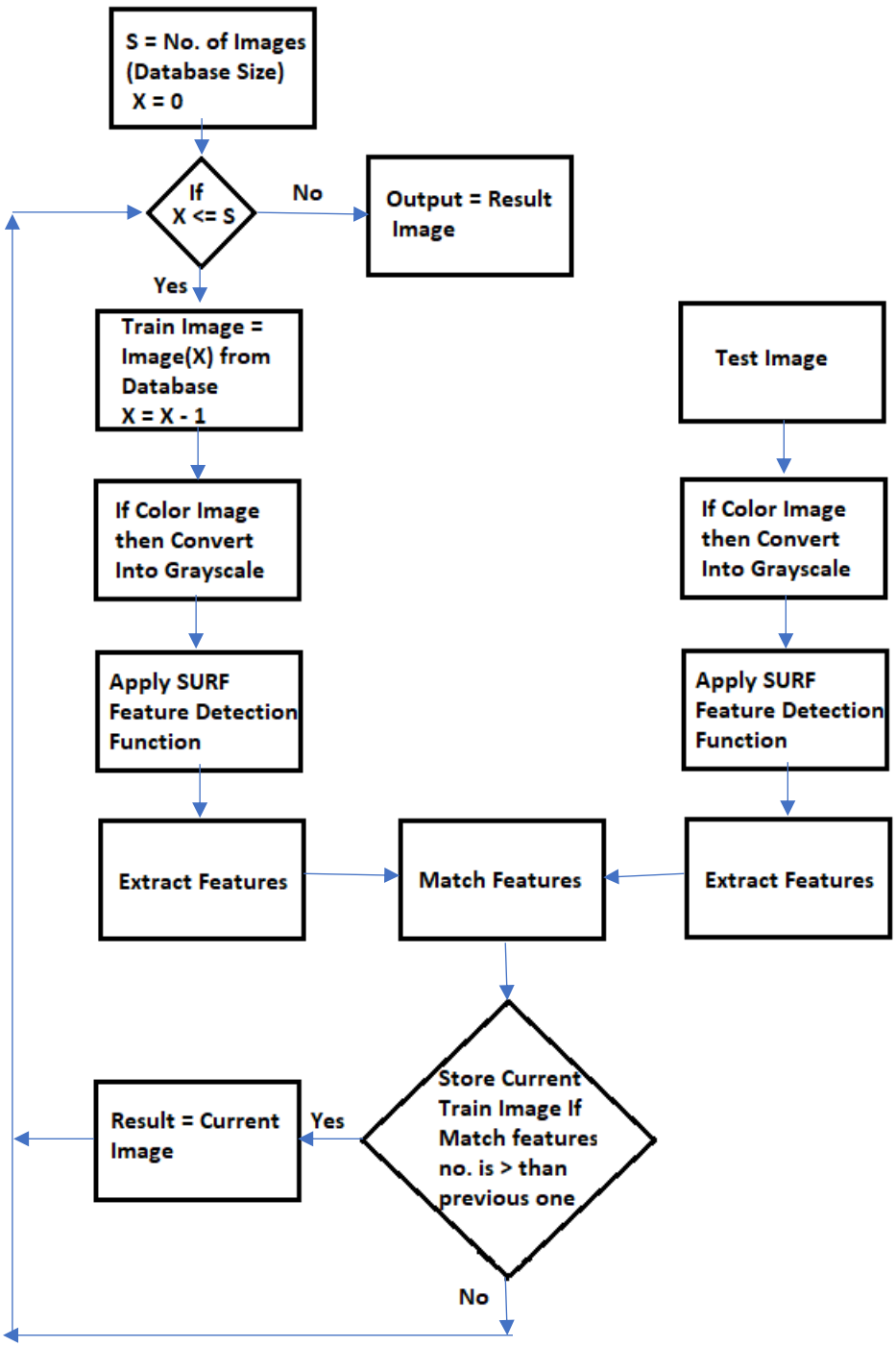


Fig: VMMR Block Diagram

- **Working of Block Diagram:**
  The process is divided into 4 parts: Input, Surf Detector, Extract Features, Match Features. One test image feature points are checked with all the images in Train Database one by one and the image with most matching point is shown as output.

1. **Input:**

- User can select any Test image from test database as input to Test Image block. That Test Image is then converted into grayscale if it's a color image. Once it is converted into 2-D grayscale image is forwarded to SURF function.
- X is variable initially loaded with Zero value to act as a counter.
- S variable stores the value of number of images in Train Dataset Folder.
- Simultaneously on the left side of Test Image block the for loop is used to select all images one by one from Train Image database and those are also forwarded to the SURF function after converting into greyscale same as Test image.

2. **SURF Function:**
   Detect SURF features and returns a SURFPoints. Surf Point consists of object, points about SURF features detected in the given image. The detectSURFFeatures function implements the Speeded-Up Robust Features (SURF) algorithm to find blob features which perform the detection and create the matching points of SURF as an output for both Test and Train image. These points are forwarded to extract feature function.

3. **Feature Extraction:**
   These returns extracted feature vectors, also known as descriptors, and their corresponding locations, from both Test original image and Test Surf Points same for Train image. The extractFeatures function finds the descriptors from pixels of image which is surrounding an interest point. The pixels represent, and match features specified by a single-point location. Each single-point directs to the center location of a neighborhood pixels of given image.

4. **Match Features:**
   Once the Extract feature block has all Matching point of test image it will then forward this to Match features that is where matchFeatures function returns indices of the matching features in the two input feature sets test and train image resp. These are given to index pairs for displaying putatively matched features.
   This process will go on until all the Train images in dataset have gone through all above procedures and after every Train image result it will check if this image had more matching points than before and that current image is stored in Result variable to Display the Output along with its name to verify the output. (Uk.mathworks.com, 2019)

SURF function working: Speeded-Up Robust Features (SURF) algorithm

SURF descriptors are used in various applications for locating, recognizing, tracking objects and extraction of points from desired image. When Above algorithm was performed by SIFT algorithm it took more amount of time for computation. Hence, applied the SURF which offered both faster and more robust performance. As name suggests, it is a speeded-up version of SIFT. These enhancement of SIFT are designed by using box filter, interest points and Haar-wavelets. Also, after comparing with other algorithms like Harris corner detection algorithm which is scale variant which is not the problem with SURF. Hence, finally selected SURF Detector.

SURF descriptor is made by making a square region around the point of interest. SURF uses wavelet output around the point of interest. On those point of interest SURF creates rectangular bounding box and then the wavelet response is calculated for each sub section of the image.

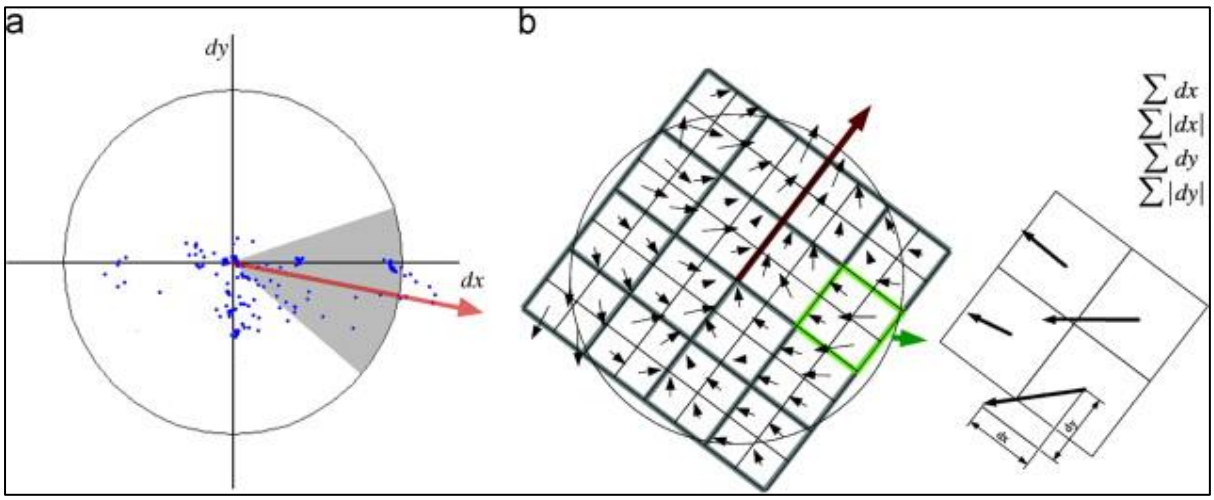


Fig: Explains the rectangular Bounding Box function working (Ars.els-cdn.com, 2019)

To detect interest points, SURF uses an integer approximation of the determinant of Hessian blob detector which is based on the sum of the Haar-wavelet response.

A Haar-like feature takes the nearest rectangular regions at a specific location in a detection window, add all the value of pixel intensities in each region shown by rectangles. Once it is done it calculates the difference between these sums. This difference is then used to categorize subsections of an image. That is the output of SURF. (En.wikipedia.org, 2019)

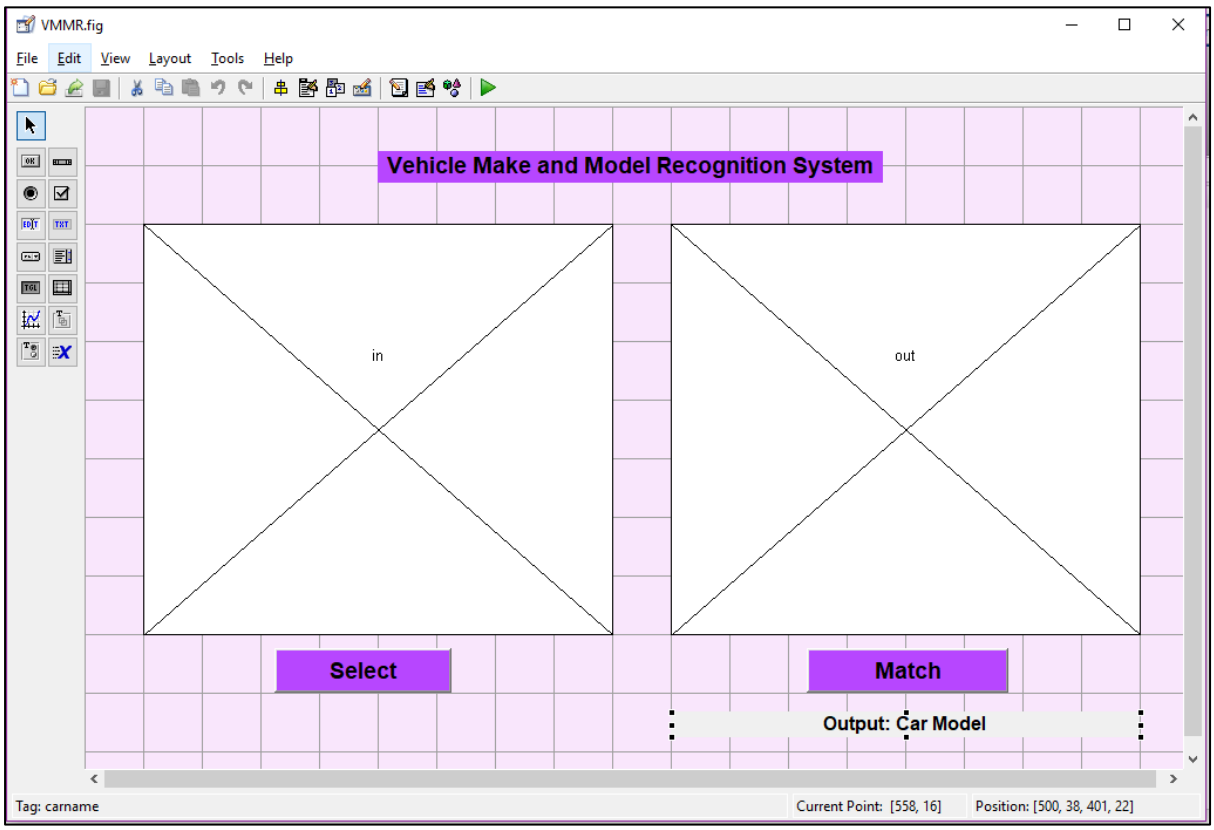
Implementation

1. Code

SURF is implemented using following main standard libraries of feature detectors in MATLAB with the help of inbuild Computer Vision Toolbox functions: (Uk.mathworks.com, 2019)

- Browse function: uigetfile
- Test Image as input: fullfile
- Display it guide Axis: handles.axisname
- Read images from database: dir
- Check if image is color image if yes then convert into grayscale: rgb2gray
- Compare image with current image from training database: For loop
- Read Training image data path: fullfile
- Get the SURF features: detectSURFFeatures
- Extract features: extractFeatures
- Match features: matchFeatures

2. GUI:



Dragged and dropped Push button, Axis, Text box, Label components

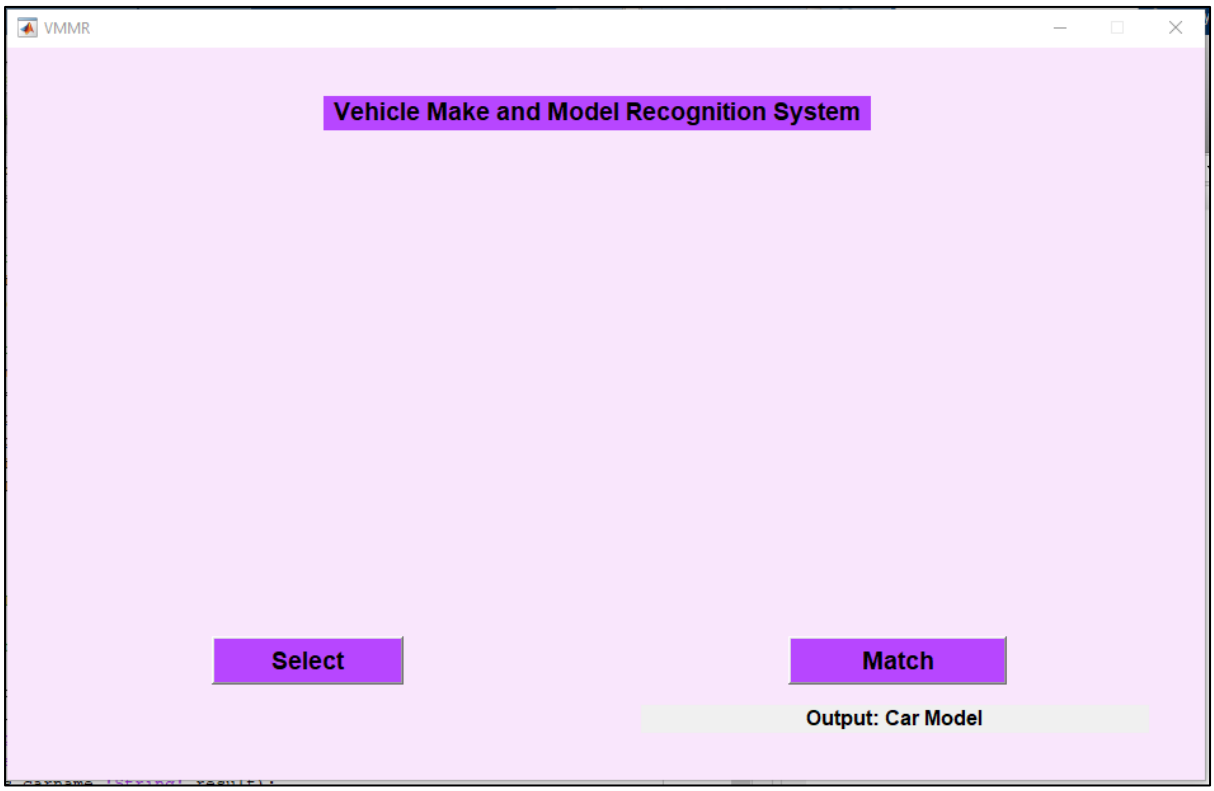
Changed below Properties:

- BackgroundColor, ForegroundColor
- String: to Display on screen
- Tag: to identify component needed for function call.

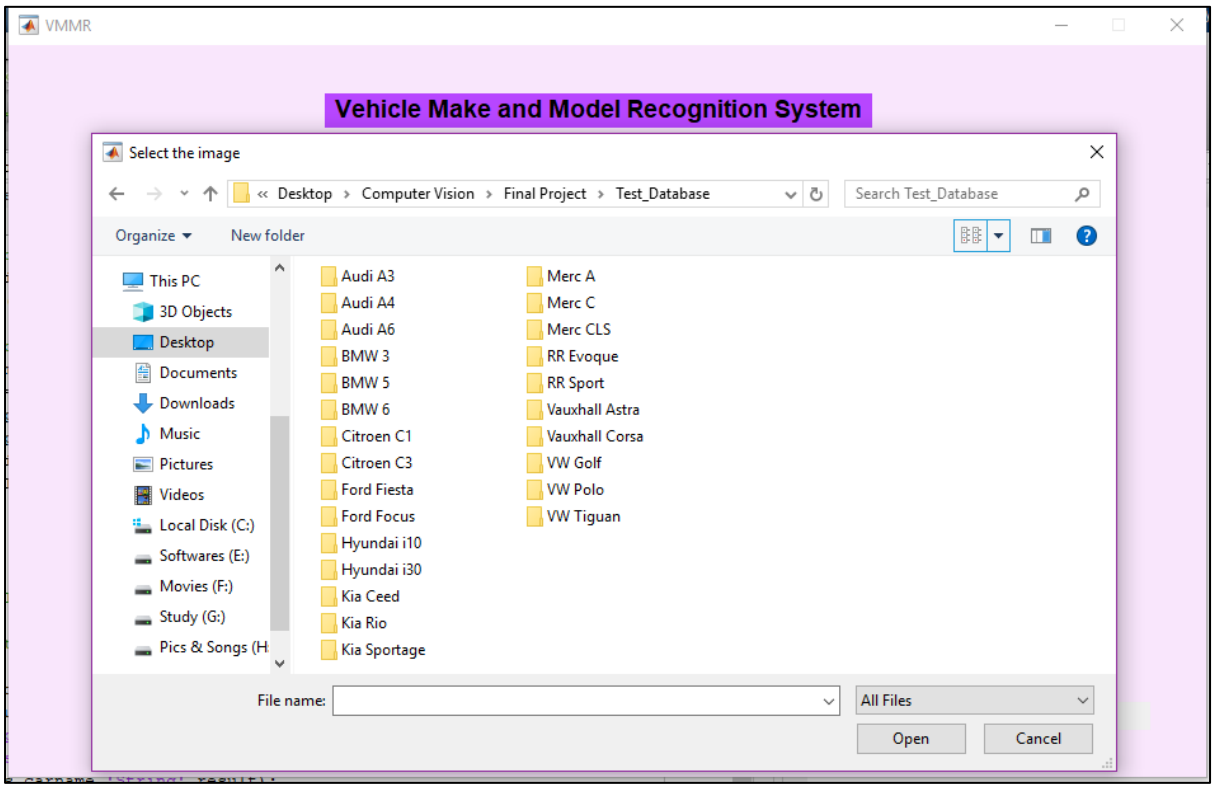
- Push Button: Once clicked on this it will call the Search_pushbtn function which is used to open new window for browsing and selecting any Test database image that image is displayed it on in axes, and Match_pushbtn will call the match function and shows the result
- Axis: visibility unchecked so no output till user selects the image.

3. Working Functionality of GUI

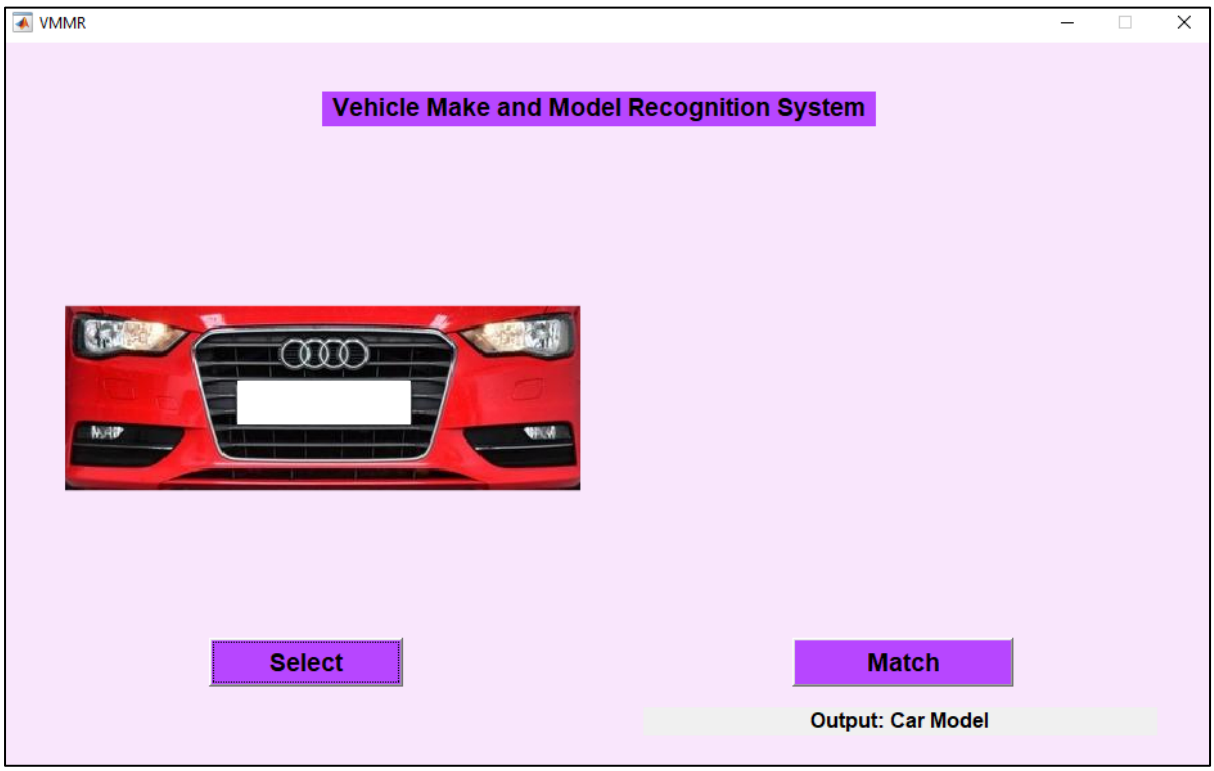
1. When clicked on **Run** Program gets the below pop-up



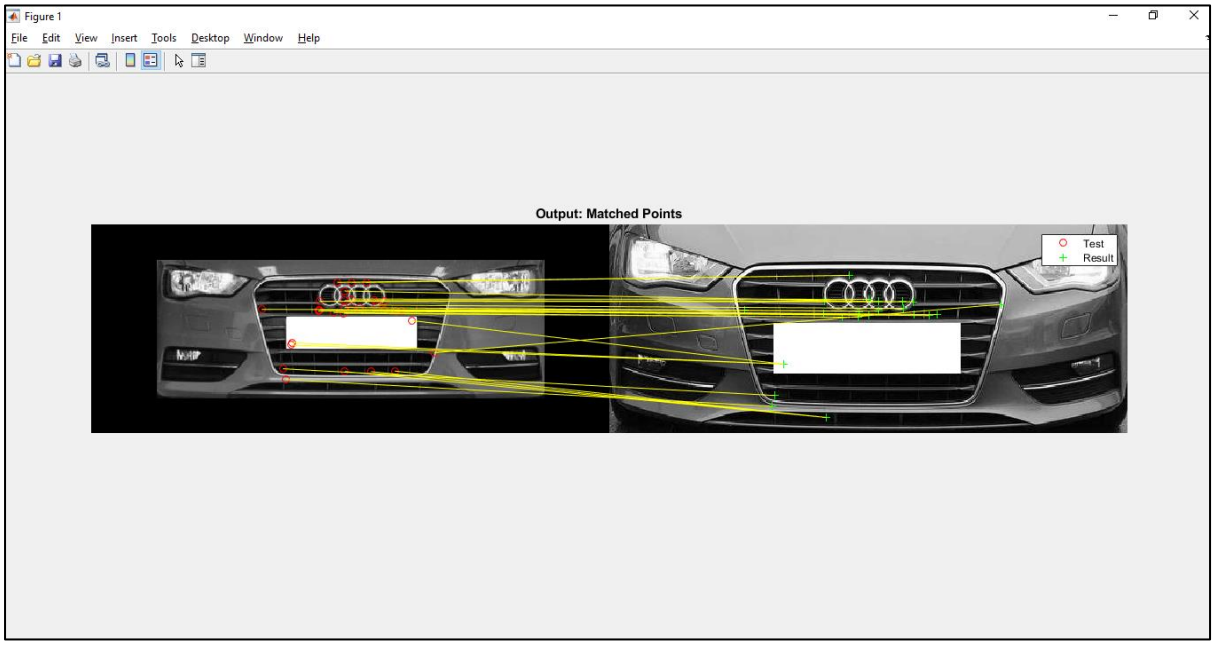
2. Clicked on Select option it gives user to browse ad select image from Test_Database where each folder has 10 test images of each car model.



3. After Selecting any image, it will be shown as below



4. Two Windows Pop-up one with Matched Point due to Montage functionality used in MATLAB code and another window which shows the name of Car Model along with the color version of an image next to selected test image.
   For example: User has selected Audi A3:



Testing & Evaluation

The Database contains 625 images in total.
First Folder: Training_Database which includes 25 vehicle make-models each 15, in total 375 images.
Second Folder: Testing is done by randomly selecting 10 images of each 25 vehicle make-models which are 250 images stored in folder named Test_Database.
Hence Each image is different in both folders.
So, one Test image will check each of 375 images for best matched feature every time.
Below are the Final testing results with accuracy and precision for each 25 vehicle make-models its accuracy is calculated on basis of 10 images each.

Vehicle Make and Model Name	Matched Number	Accuracy
1. Audi A3	9/10	90%
2. Audi A4	6/10	60%
3. Audi A6	5/10	50%
4. BMW 3	9/10	90%
5. BMW 5	10/10	100%
6. BMW 6	9/10	90%
7. Citroen C1	8/10	80%
8. Citroen C3	8/10	80%
9. Ford Fiesta	9/10	90%
10. Ford Focus	8/10	80%
11. Hyundai i10	7/10	70%
12. Hyundai i30	6/10	60%
13. Kia Ceed	8/10	80%
14. Kia Rio	9/10	90%
15. Kia Sportage	10/10	100%
16. Mercedes A	9/10	90%

17. Mercedes C	8/10	80%
18. Mercedes CLS	7/10	70%
19. Range Rover Evoque	9/10	90%
20. Range Rover Sport	10/10	100%
21. Vauxhall Astra	6/10	60%
22. Vauxhall Corsa	8/10	80%
23. Volkswagen Golf	9/10	90%
24. Volkswagen Polo	9/10	90%
25. Volkswagen Tiguan	9/10	90%

Above result are Proved by Testing performing for 25 vehicles make and model with each 10 test images =250 Images are available at the Appendix.

Accuracy Calculation:
After rigorous testing it is observed as **205** correctly detected out of **250** hence total accuracy = 205/250 *100 = **82%**
For Audi it is observed that mismatched result pointed to Audi A6 in Fig. 1.6 might be because of external light source on image. Audi 4 had the same mismatched result pointed to Audi A3 in Fig. 2.2. Also, for Audi 6 All wrong matches pointed to Audi A3 model.
Hence it is observed that Audi has very minor change in model which SURF is getting failed to detect.
Where as in the result of Range Rover Sport, Kia Sportage and BMW 3 it had shown 100% accuracy.

Convolutional Neural Network (CNN)

Convolutional Neural Network widely referred as a CNN or comp net is an artificial neural network that has been most popularly used for various purpose like analysing images, data analysis and classification problems. As an artificial neural network that has variety of specialization to detect patterns and make sense of them, this pattern detection is what makes CNN so useful for image analysis.

- **Design:**

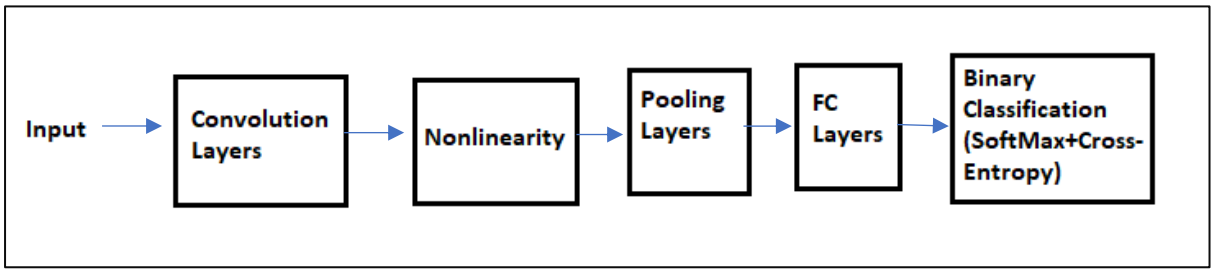
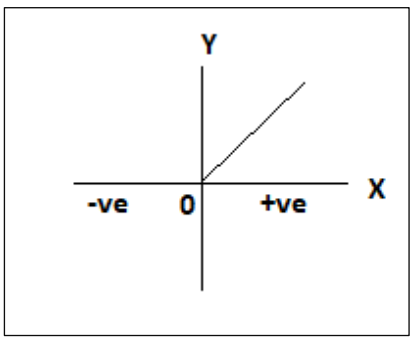


Fig: Block Diagram of basic CNN

The above block diagram of CNN is designed through by stacking basic units. Input is the Dataset with thousands of images of which 70 % of dataset is used for Training the model and 30% for Testing purpose. The % may vary depending upon application but the Training Dataset is always bigger. The main part of a CNN architecture is the convolutional layers. **Convolutional layer** receives input then transforms the convoluted input through neurons and then outputs the transform input to the next layer. With each convolutional layer we need to specify the number of filters the layers should have. filters are actually what detect the patterns in any single image multiple edges shapes textures objects etc.

As image recognition is process based on **nonlinearity**. To introduce the convoluted image which is linear process into non-linearity we use Activation function like ReLU (Rectified Linear Unit), sigmoid, tanh etc. ReLU function gives the higher slope value for higher input values this is why ReLU function can give high optimization which gives faster converge. For negative values of input, it gives negative slope as shown in figure beside. Hence, most preferred choice of activation function is ReLU.



Pooling Layer is also called as Down Sampling. Once the feature is activated by ReLU its location is not important. But we need a faster computation as CNN deals with thousands of images. This layer is used to cut down those points from image which are not necessary. This layer also controls the overfitting by reducing the complexity.

FC layer deals with neurons. One more layer is introduced which is **Dropout Layer** to prevent overfitting and maintain regulation. One way to make training deep models easier is to add skip connections that connect non-consecutive layers. Using such connections gives the following layers a reference point so that adding more layers won't worsen the performance. The skip connections also create an additional path for the gradient to flow back more easily. This makes it easier to optimize the earlier layers.

SoftMax is used for Output visual representation.

- **Architecture of AlexNet :**

Designing the architecture of a model also involves choosing the types of layers and the way they are arranged and connected to each other usually involves a lot of trial and error.

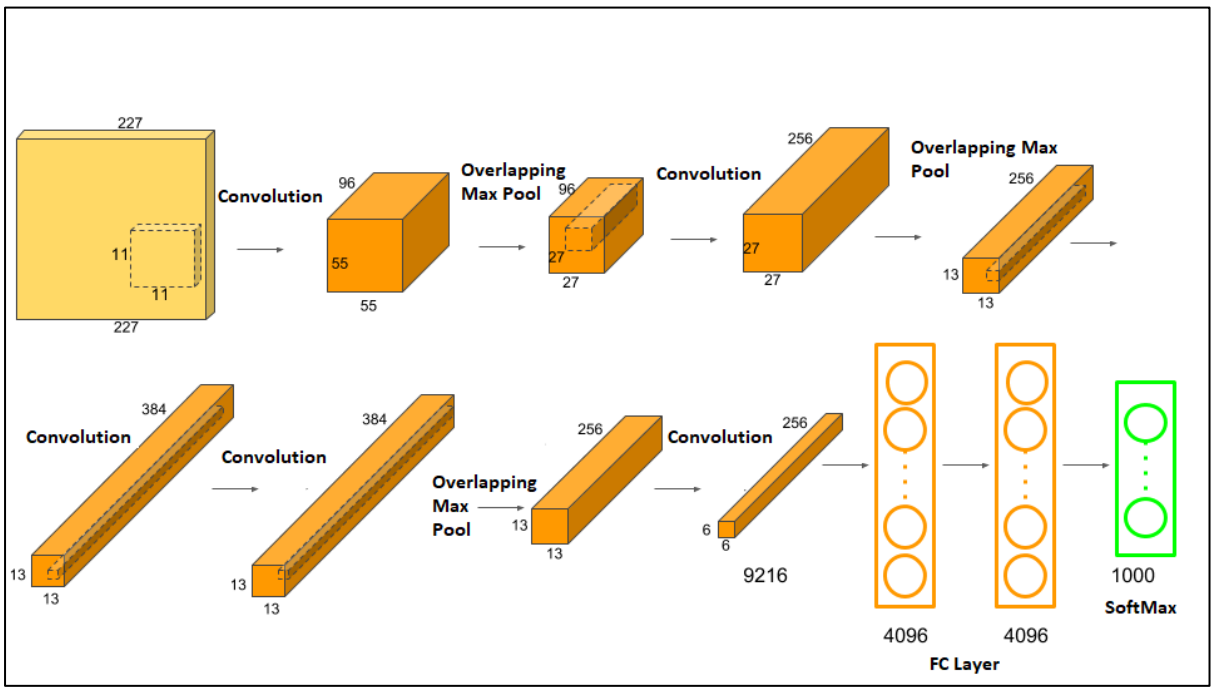


Fig: Architecture of **AlexNet** consists of 5 Convolutional Layers and 3 Fully Connected Layers(Nayak, 2019)

There are various parameters needed to design such as stride, kernels, batch size etc. Sliding window step size, also known as the stride. If input to AlexNet is an colour image of size 256×256 pixel. Hence, all images in the training set and all test images has to be 256×256 size.

Taking these considerations below calculations are performed as shown in the architecture above.

Convolution 11*11: stride = 4, 96 kernels ,(227-11)/4+1 = 55
Overlapping Max Pool 3*3: stride = 2 , (55 -3)/2+1 =27
Convolution 5*5: Pad = 2 , 256 kernels, (27+2*2-5)/1+1= 27
Overlapping Max Pool 3*3: stride = 2 , (27-3)/2+1= 13
Convolution 3*3: Pad = 1 , 384 kernels ,(13+2*1-3)/1+1= 13
Convolution 3*3: Pad = 1 , 384 kernels , (13+2*1-3)/1=1 =13
Convolution 3*3: Pad = 1 , 256 kernels , (13+2*1-3)/1=1 =13
Overlapping Max Pool 3*3: stride = 2 , (13 -3)/2+1 =6

By Using MATLAB Deep Learning Toolbox below AlexNet is designed as follows:
The Deep Learning Toolbox has Drag and drop feature to select the layers as mentioned in basic diagram and Architecture below is the image of multilayer AlexNet.

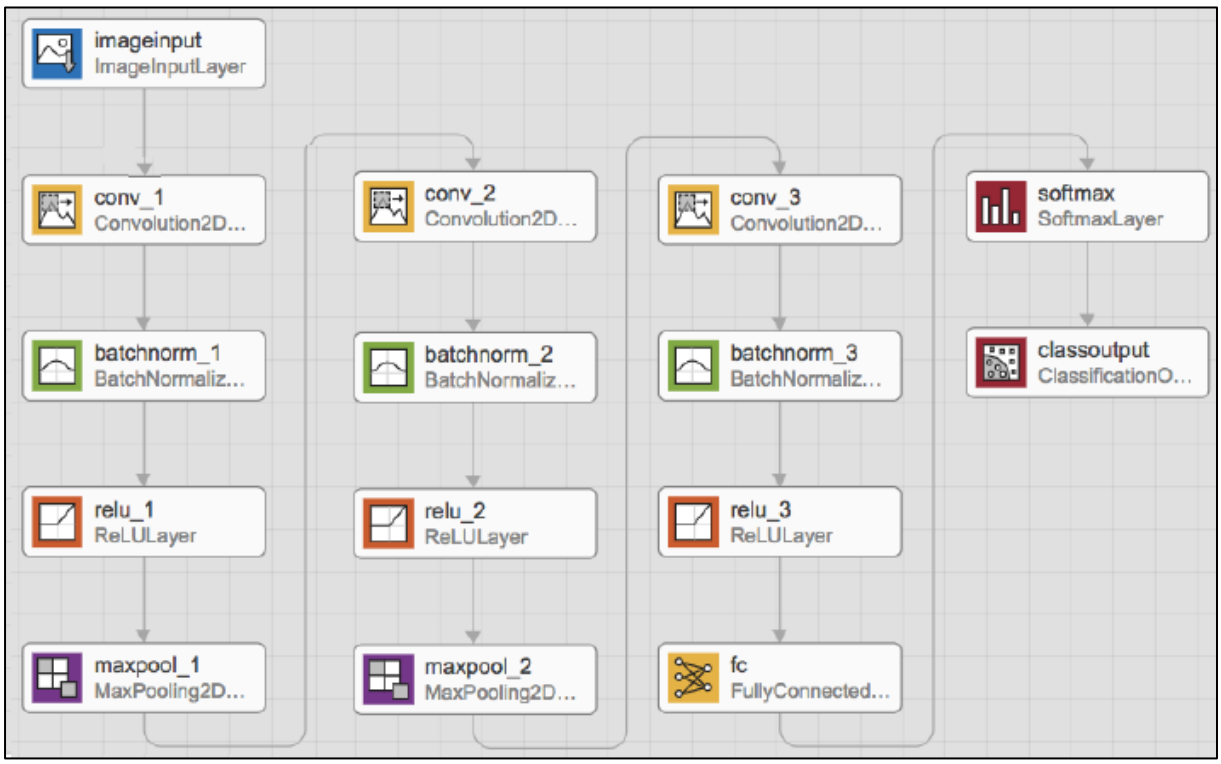


Fig: Design of AlexNet using MATLAB

Size of Dataset:

Whether the size of the dataset provided is sufficient is depends on factors that determines the sample size are: the classifier in question, the statistical characteristics of the data , on the nature of the problem, the number of features, and the complexity of your network architecture etc.

if time and sources are limited for collecting new data best use the Transfer learning method. Models trained on one task capture relations in the data type and can easily be reused for different problems in the same domain.

When we take a model that has already been trained on some large dataset and fine tune the weights to adapt it to our specification set problem. This is called **transfer learning**. This approach works in many practical case. It usually has more benefits to start smaller and increase the model capacity until the validation error stops improving. The same number of trainable parameters whether it's better to have more layers or more units per layer.
So, in nutshell, Transfer Learning is nothing but the reusing the pre-trained model on a new problem. As it enables us to train Deep Neural Networks with comparatively little data is widely used in Deep learning.

Overfitting:

Overfitting occurs when our model becomes really good at being able to classify or predict on data that was included in the training set but is not so good at classifying data. For example, when asked question in exam conceptually but which was not taught in the class we are unable to answer it because without understanding concept we cannot think outside of box. Similarly, If model don't understand the concept it fails to detect test image this is called as Overfitting.

we can tell that the models overfitting based on the validation accuracy and loss as well as the training accuracy and loss if the validation metrics are considerably worse than the training metrics then that's indication that our model is overfitting.

Methods to overcome model Overfitting:

1. **Data Augmentation**
   Data augmentation this is the process of creating additional augmented data by reasonably modifying the data in our training set for image data for example we can do these modifications by cropping rotating flipping or zooming. Augmentation allows us to add more data to our training set that's similar to the data that we already have but it's just reasonably modified to some degree so that it's not the exact same as original.

2. **Dropout**
   The general idea behind dropout is that if you add it to a model it will randomly ignore some subset of nodes in a given layer during training.It drops out the nodes from the layer hence the name dropout this will prevent these dropped out nodes from participating in producing a prediction on the data this technique may also help our model to generalize better to data it hasn't seen before

3. **Regularization**
   Regularization helps us to select the model complexity to fit the data. Regularization penalizes all used features, not a selected subset.

Conclusion and Further work:

To conclude from above testing it is observed that SURF computations are faster than SIFT and more accurate. However, when working with image with lot of background details other than Frontal view of vehicle it was showing the less accuracy. From Test results it is observed that Audi had 3 models and the Audi A6 model mostly pointed out the other Audi models than itself. Whereas Kia Sportage, Range Rover Sport, BMW3 has the best Dataset or very different features hence it showed 100% accuracy and the Audi matched with the Audi Make but wrong model hence it has less accuracy as compare to other models. Hence, it is observed at the more clear features shown in image while capturing and clutter free the data with cropped and perfect aspect ration the more efficient output we get.
The size of Dataset doesn't matter if the data has noise. And for the CNN design dataset size depends on features which we want to extract, application. However, best way is to use transfer learning to reduce time and cost.
Designed system gives one more security level to the vehicles.Due to time constraint I was not able to show all test results and CNN AlexNet implementation, but this is definitely in the future scope.

Appendix

- **Code**

```
function varargout = VMMR(varargin)
% VMMR MATLAB code for VMMR.fig
%      VMMR, by itself, creates a new VMMR or raises the existing
%      singleton*.
%
%      H = VMMR returns the handle to a new VMMR or the handle to
%      the existing singleton*.
%
%      VMMR('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in VMMR.M with the given in arguments.
%
%      VMMR('Property','Value',...) creates a new VMMR or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before VMMR_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to VMMR_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help VMMR
```

```
% Last Modified by GUIDE v2.5 06-Jan-2019 15:22:21

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @VMMR_OpeningFcn, ...
                   'gui_OutputFcn',  @VMMR_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before VMMR is made visible.
function VMMR_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no out args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to VMMR (see VARARGIN)

% Choose default command line out for VMMR
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = VMMR_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning out args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line out from handles structure
varargout{1} = handles.output;




% --- Executes on button press in select_pushbtn.
function select_pushbtn_Callback(hObject, eventdata, handles)
% hObject    handle to select_pushbtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global path;
[FileName,FilePath ]= uigetfile('*','Select the image'); %Browse function
path = fullfile(FilePath, FileName);               % Test Image as input
imshow(path,'Parent',handles.in);                  % Display it guide Axis
```

```
% --- Executes on button press in match_pushbtn.
function match_pushbtn_Callback(hObject, eventdata, handles)
% hObject    handle to match_pushbtn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global path;
test_img=imread(path);
temp=0;
% Read images from database
training_img=dir(strcat('Training_Database','/*.jpg'));

% Check in image is color image if yes then convert into grayscale
if(size(test_img,3)==3)
    test_img=rgb2gray(test_img);
end

% compare image with current image from training database
for i=1:size(training_img)

   % read Training image data path
   matchpath=fullfile('Training_Database',training_img(i).name);
   match_img=imread(matchpath);

   % Check if image is color image if yes then convert into grayscale
    if(size(match_img,3)==3)
       match_img=rgb2gray(match_img);
    end

   % Get the SURF features
   pts_1 = detectSURFFeatures(test_img);
   pts_2 = detectSURFFeatures(match_img);

   % Extract features
   [fe_1, pts_1] = extractFeatures(test_img, pts_1);
   [fe_2, pts_2] = extractFeatures(match_img, pts_2);

   % Match features
   index_pairs = matchFeatures(fe_1, fe_2);
   matched_points1 = pts_1(index_pairs(:, 1), :);
   matched_points2 = pts_2(index_pairs(:, 2), :);

   % count of matched features
   store=size(index_pairs);
   s=store(1:1);

   % Store current image if it has higher number of matching feature
   % points than last image
   if(s>temp)
      temp=s;
      mat_pt1 = matched_points1;
      mat_pt2 = matched_points2;
      out_img = match_img;
      result  = matchpath;
    end
end



% Display the Final Result
figure;
% Place test and result image next to each other in the same image
% and show arrows to Coordinates of points in image
showMatchedFeatures(test_img, out_img, mat_pt1,mat_pt2,'montage');
```

```
imshow(result,'Parent',handles.out);
title('Output: Matched Points');
legend('Test','Result');
set(handles.carname,'String',result);
```

Test Result of All Vehicle Make and Model:

1. Audi A3

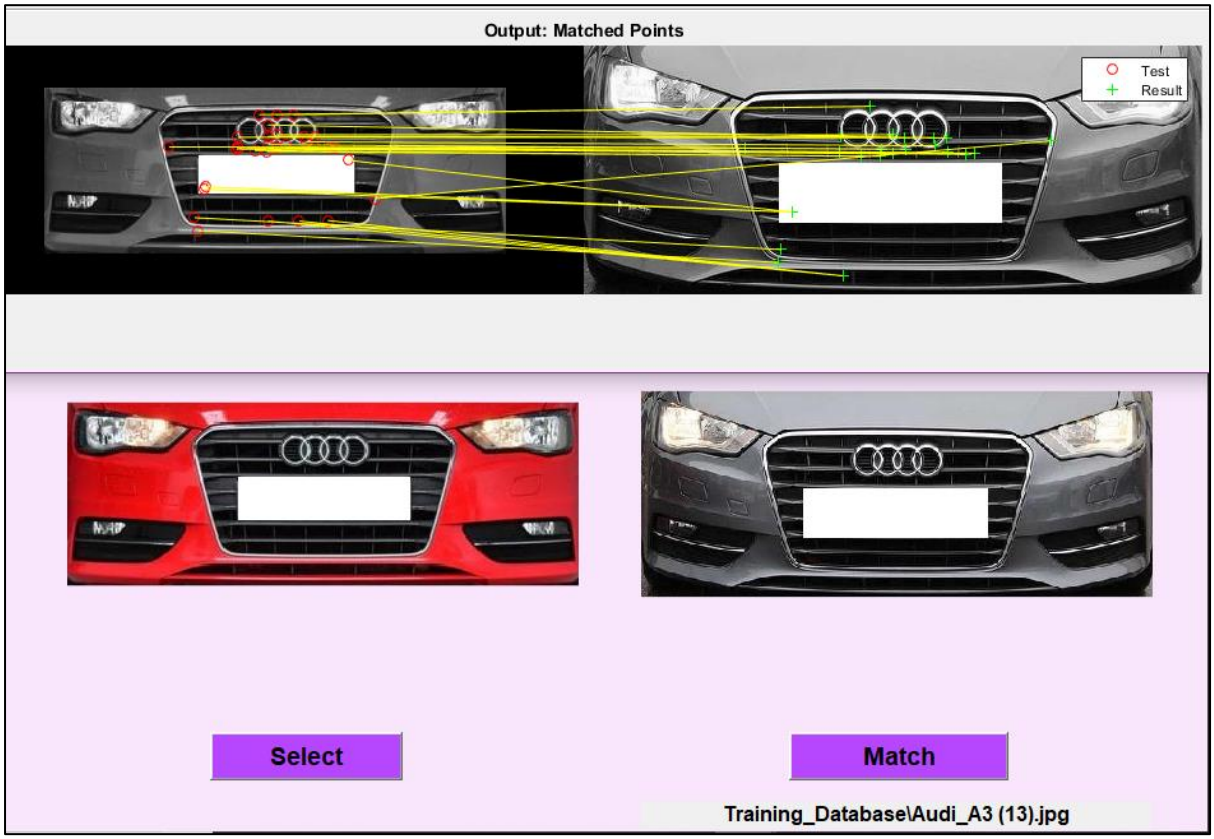


Fig 1.1 Output: Audi A3

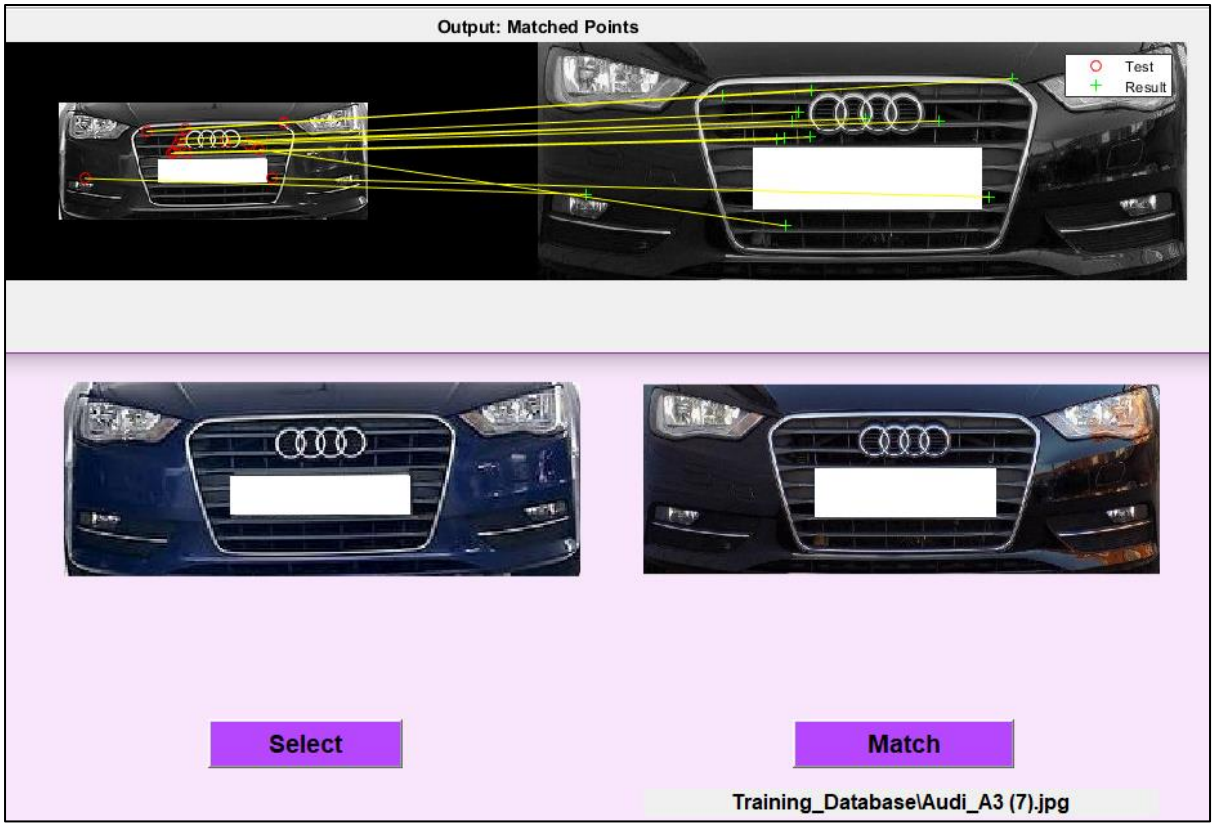


Fig 1.2 Output: Audi A3

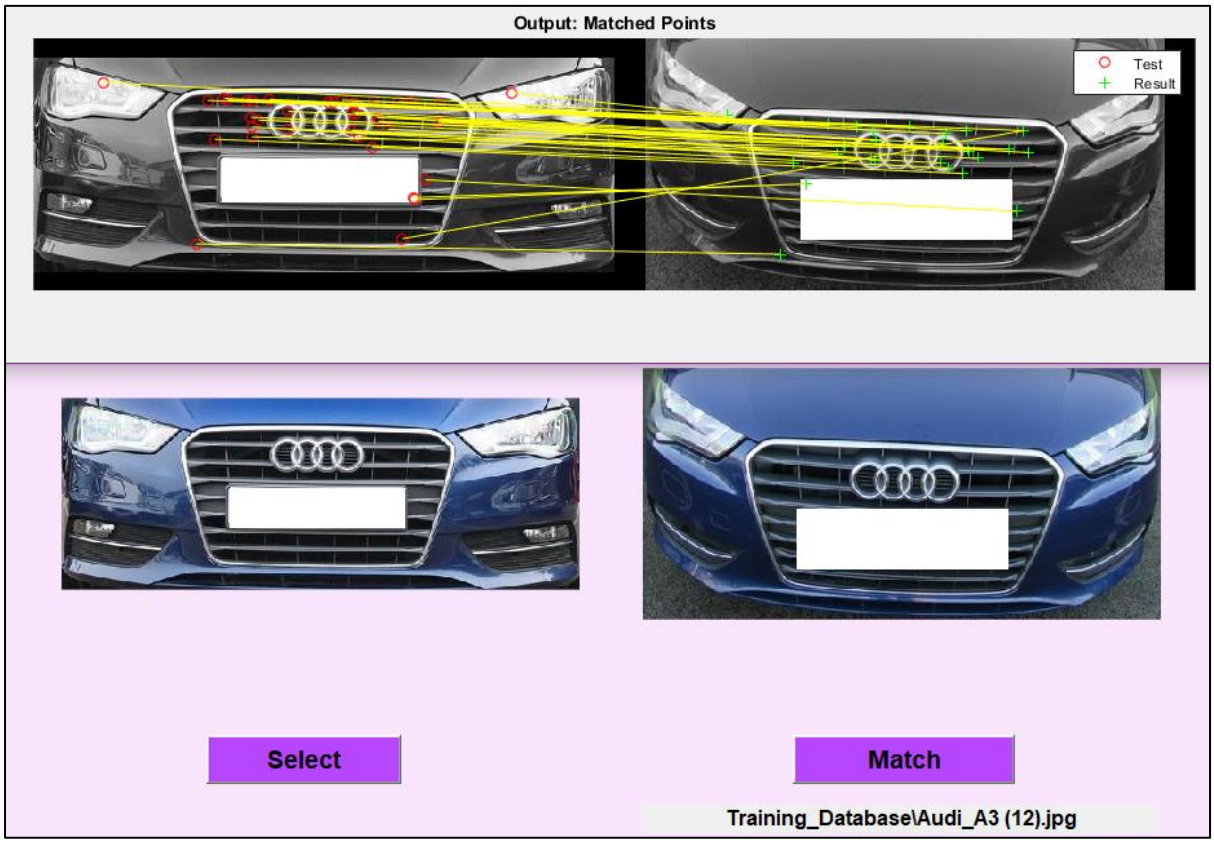


Fig 1.3 Output: Audi A3 : Higher number of matching points found

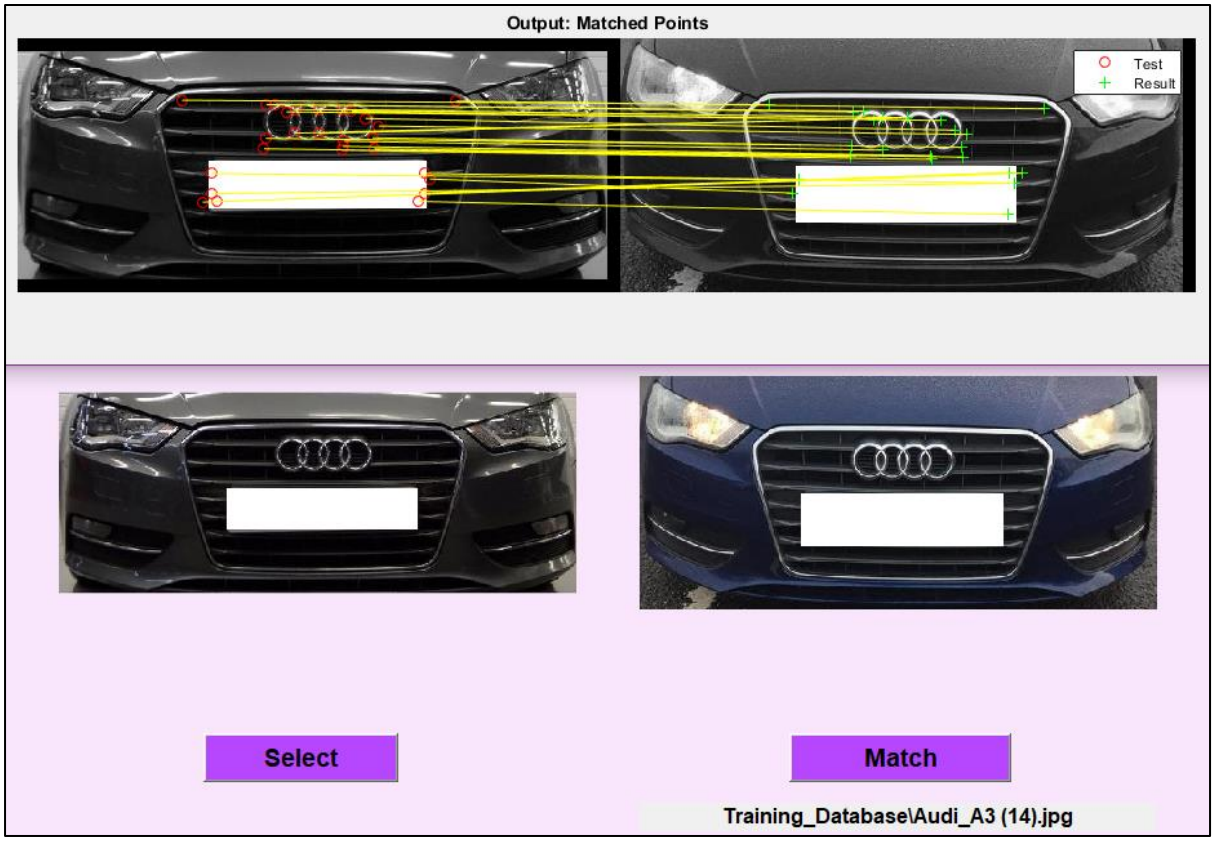


Fig 1.4 Output: Audi A3

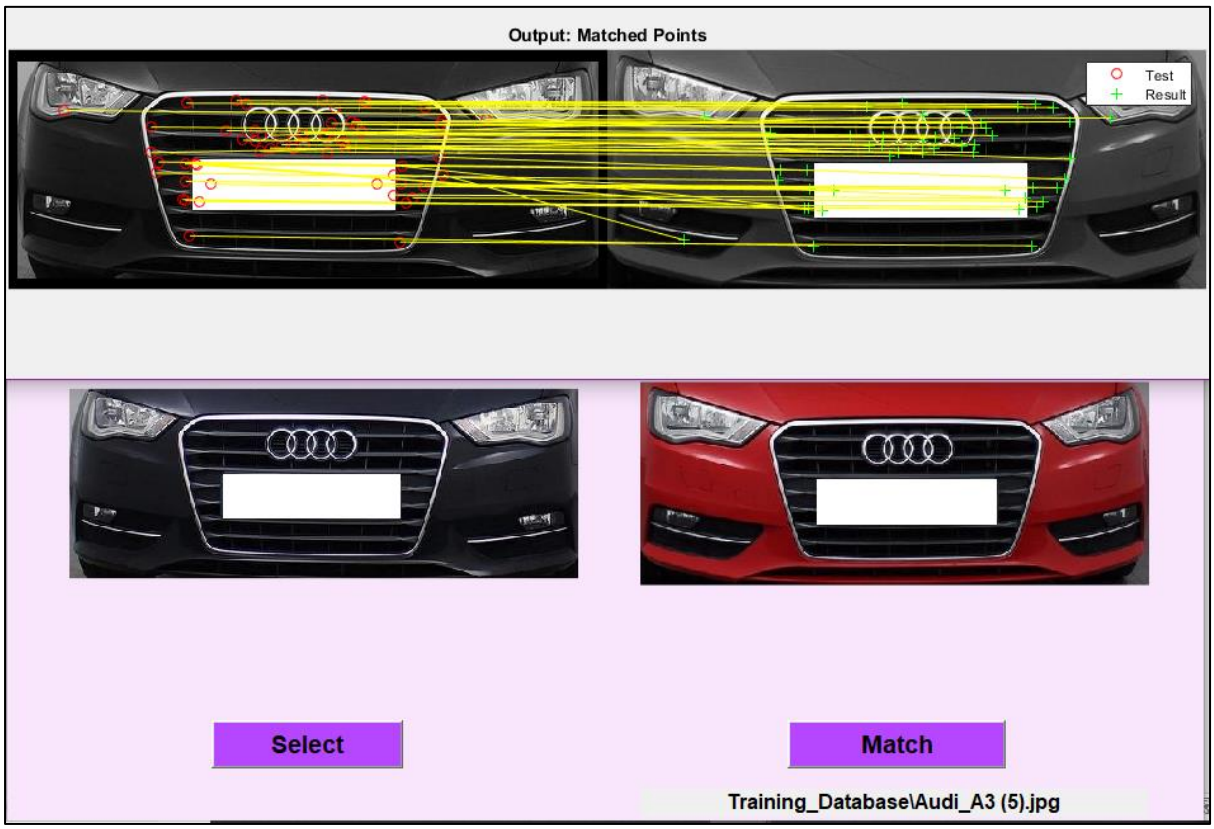


Fig 1.5 Output: Audi A3

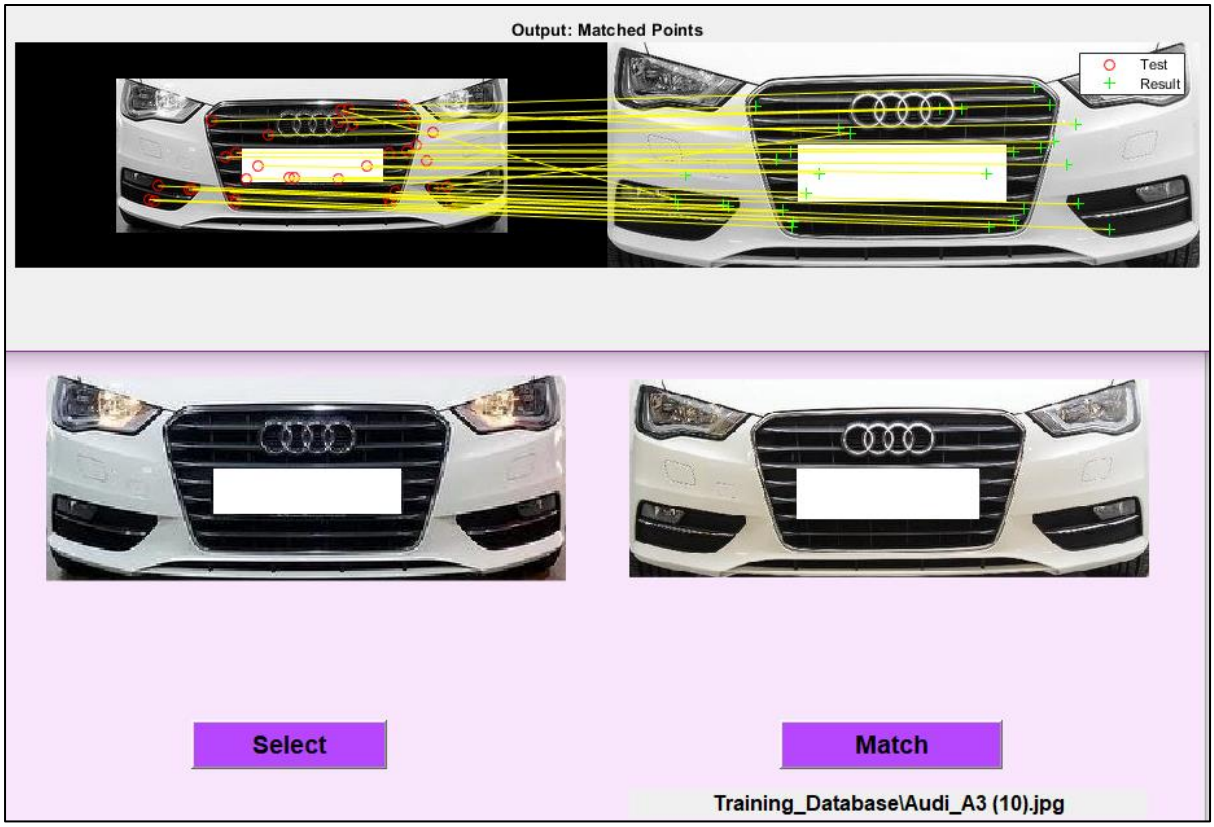


Fig 1.6 Output: Audi A3

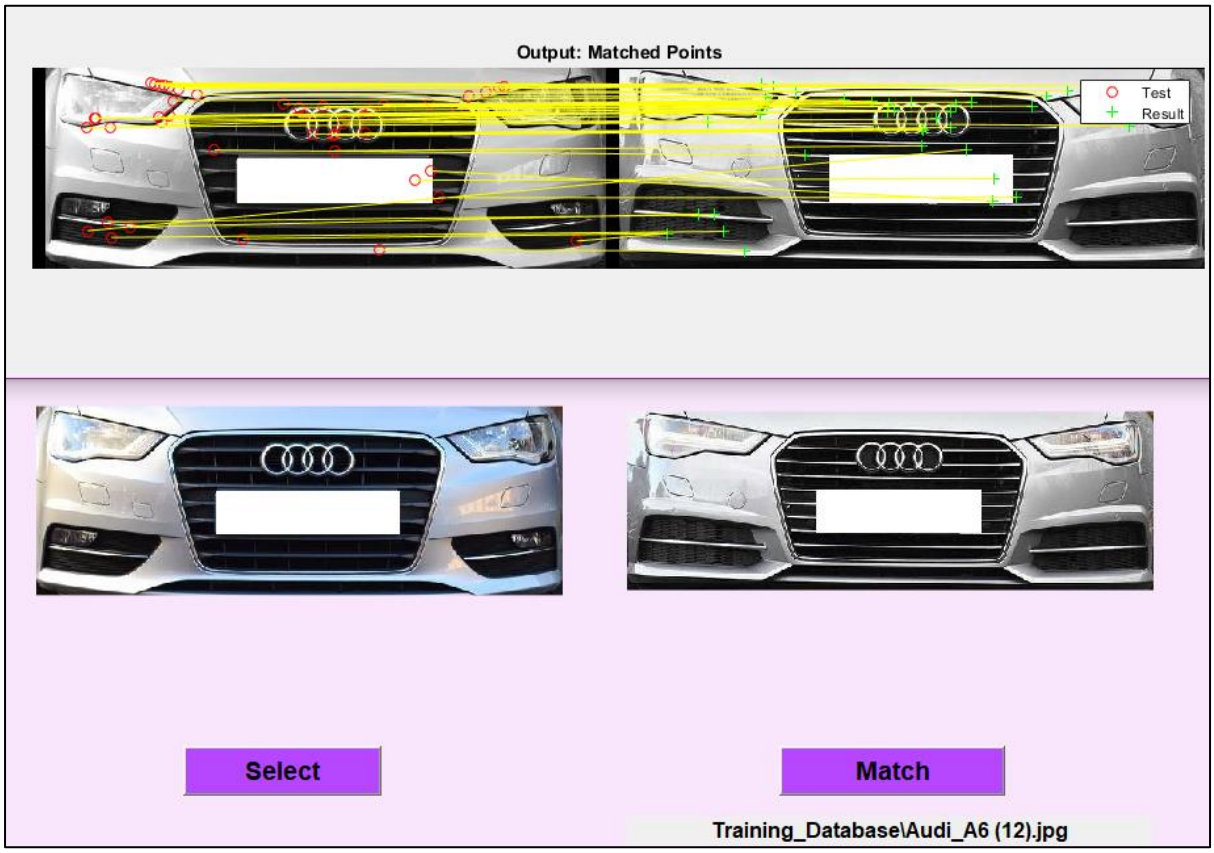


Fig 1.7 Output: **Audi A6**: Wrong Result

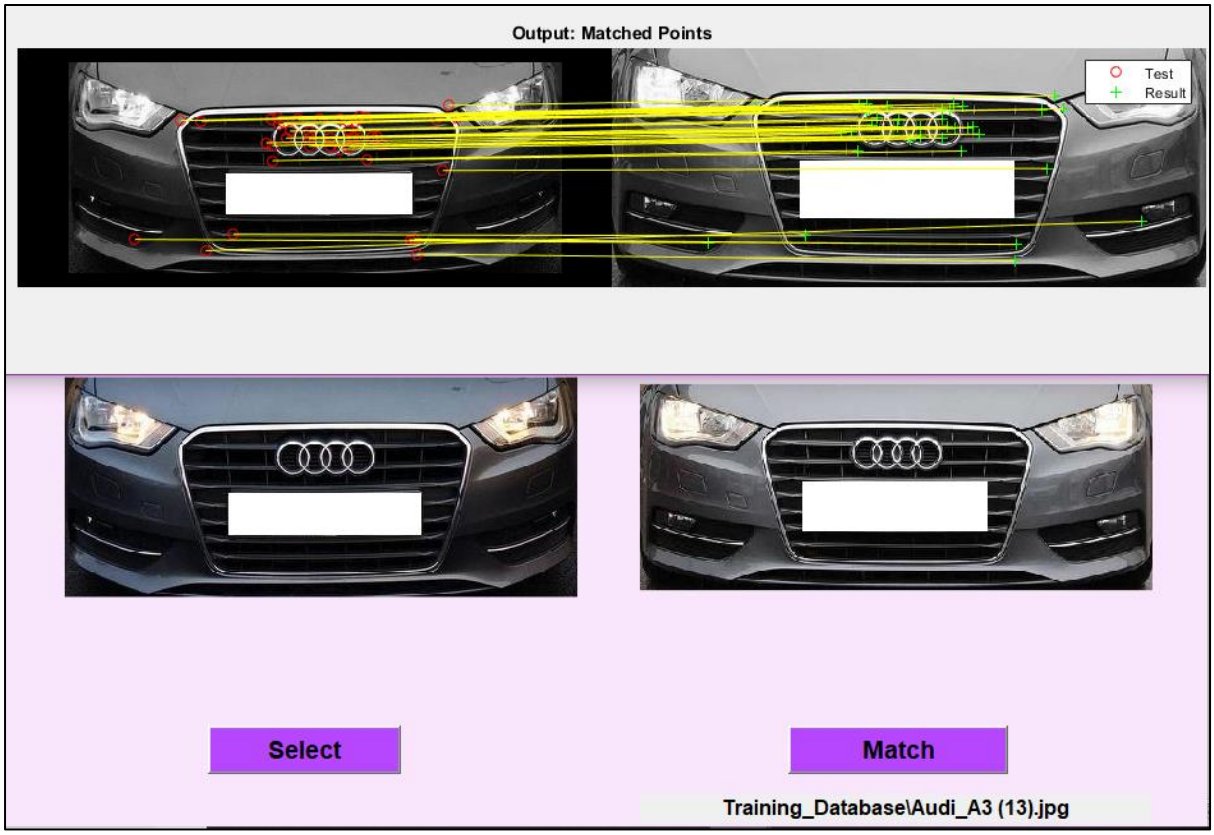


Fig 1.8 Output: Audi A3

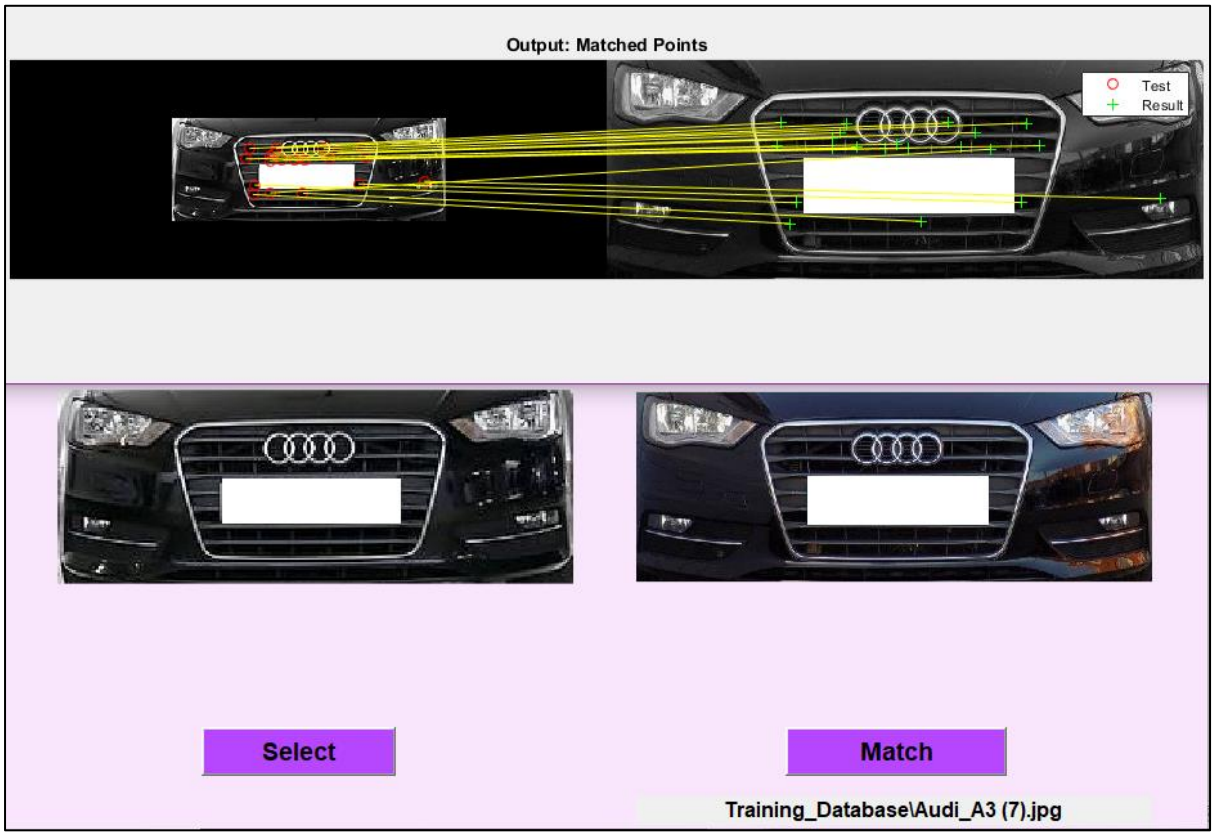


Fig 1.9 Output: Audi A3

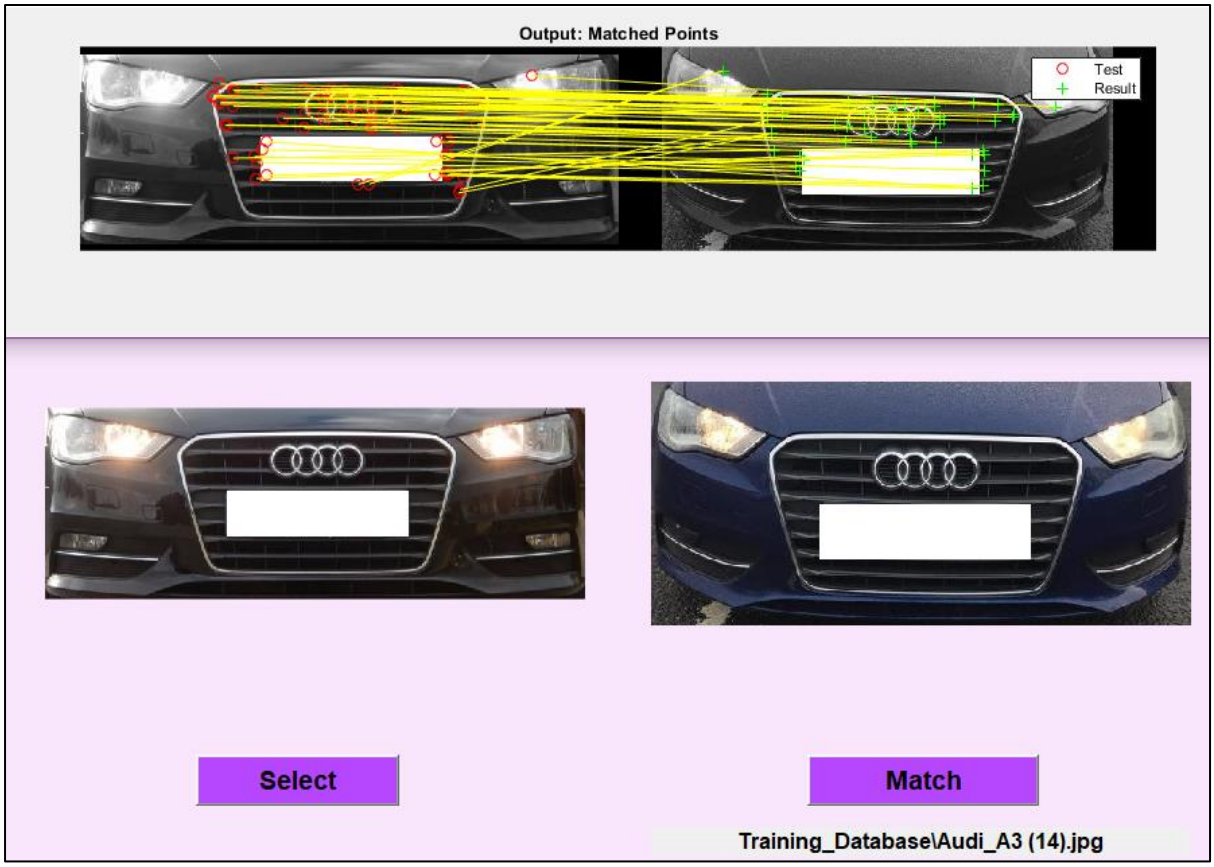


Fig 1.10 Output: Audi A3

2. Audi A4

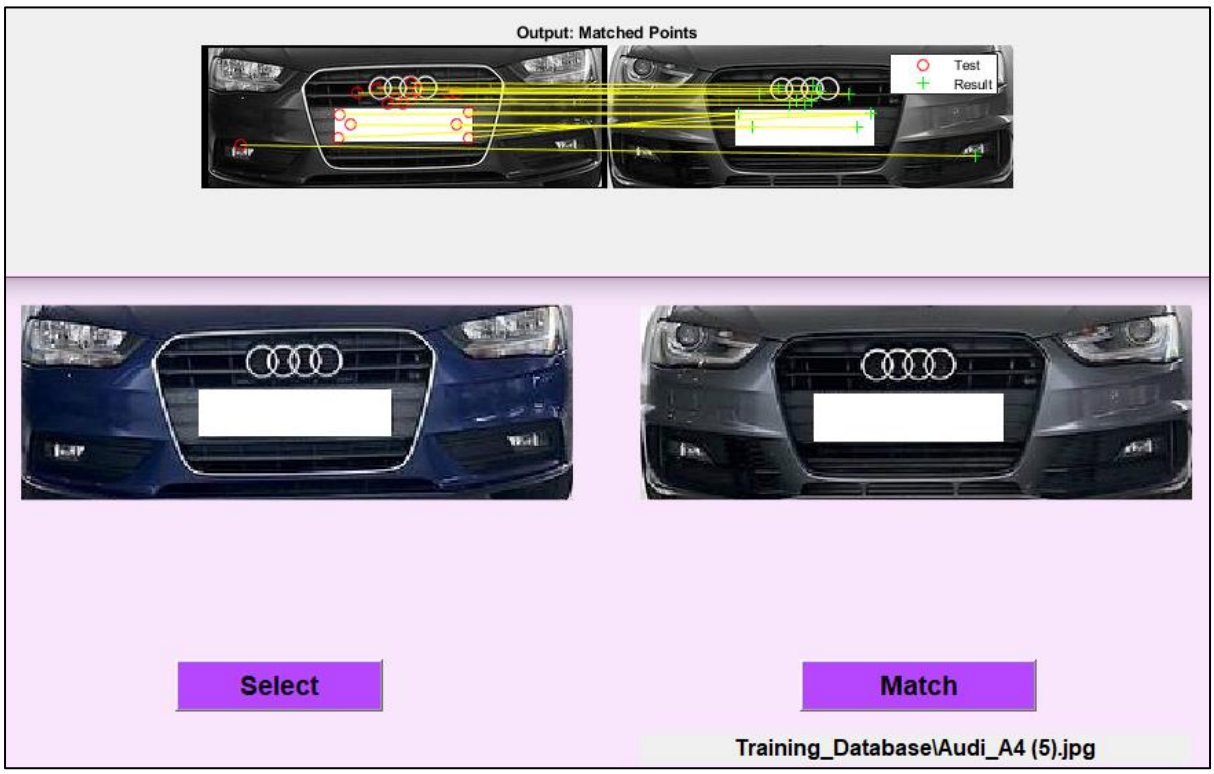


Fig 2.1 Output: Audi A4

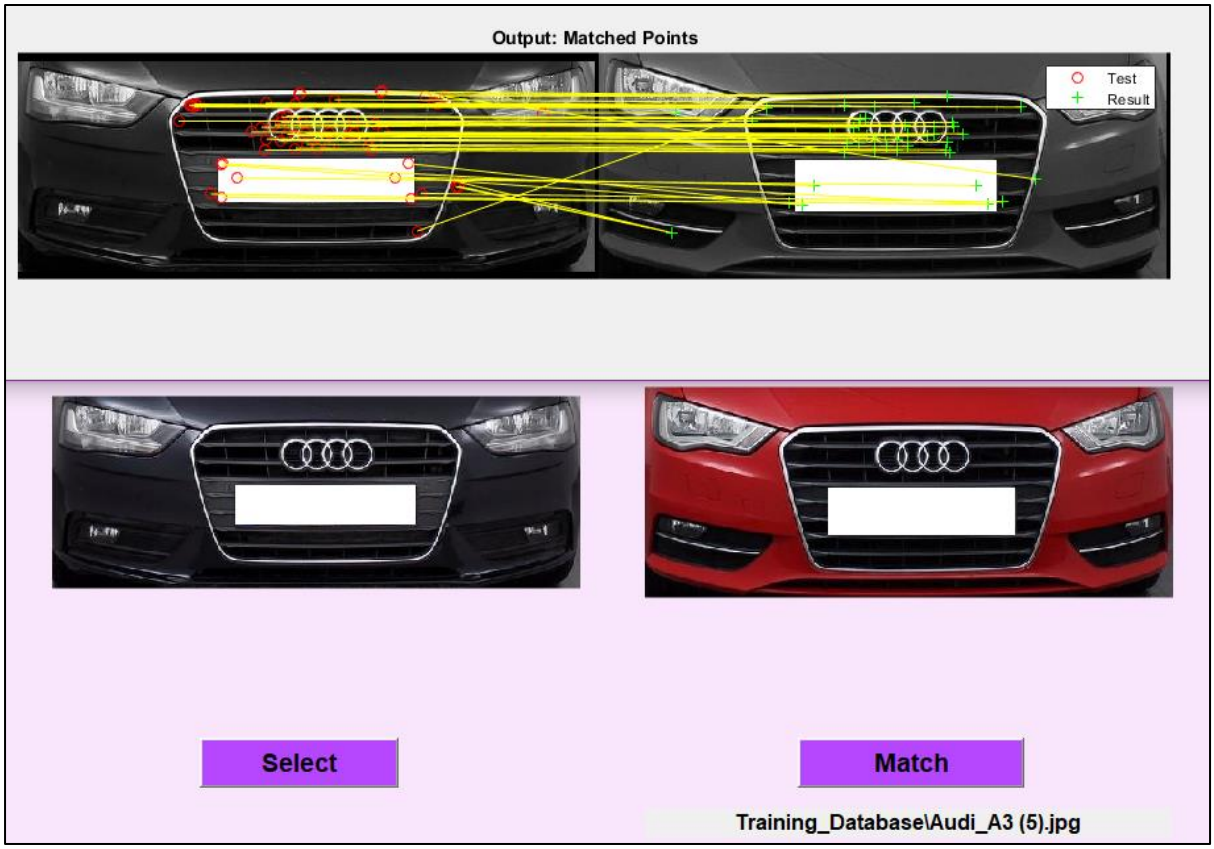


Fig 2.2 Output: **Audi A3:** Wrong Result

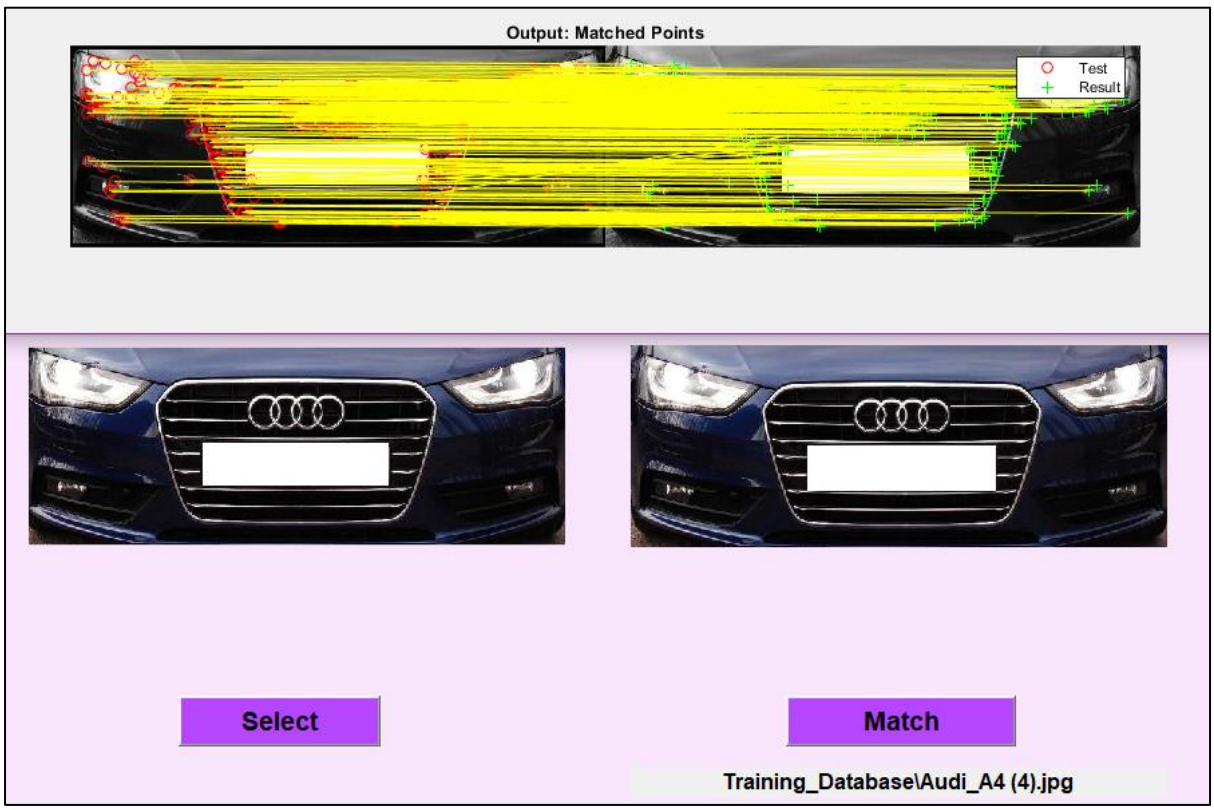


Fig 2.3 Output: Audi A4

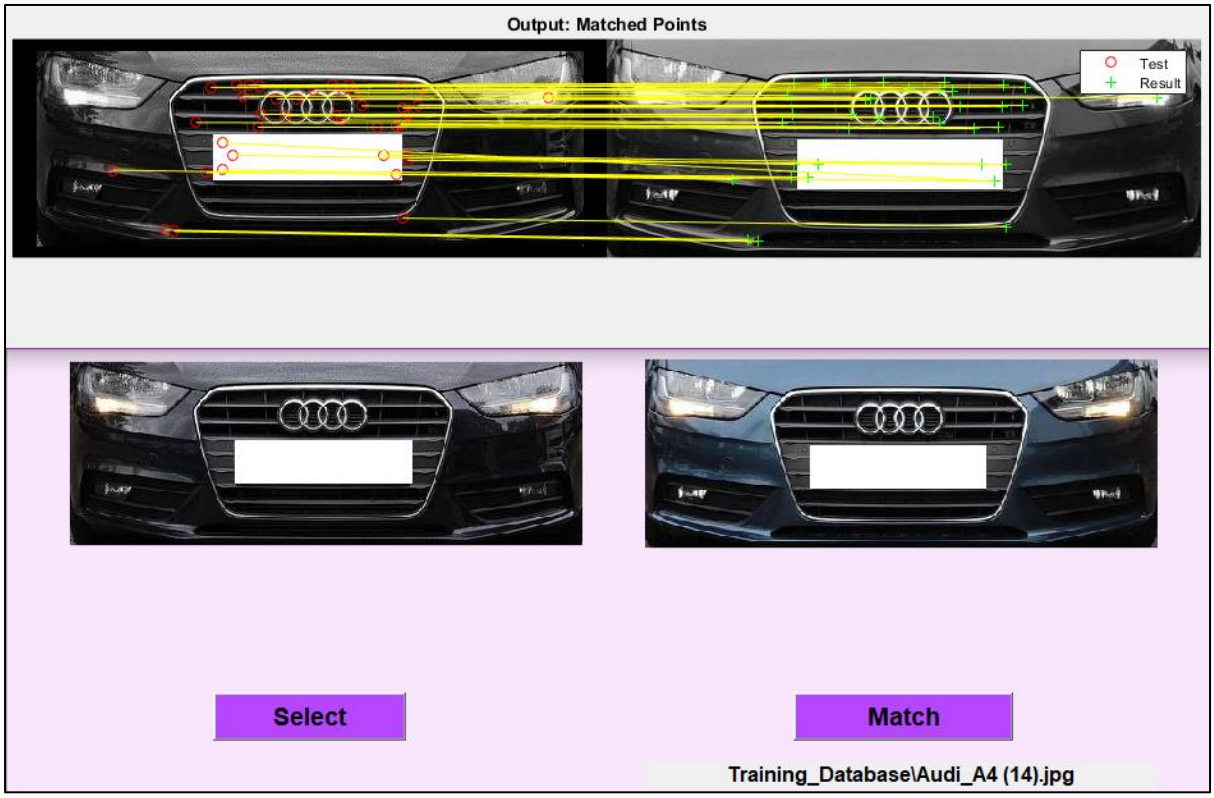


Fig 2.4 Output: Audi A4

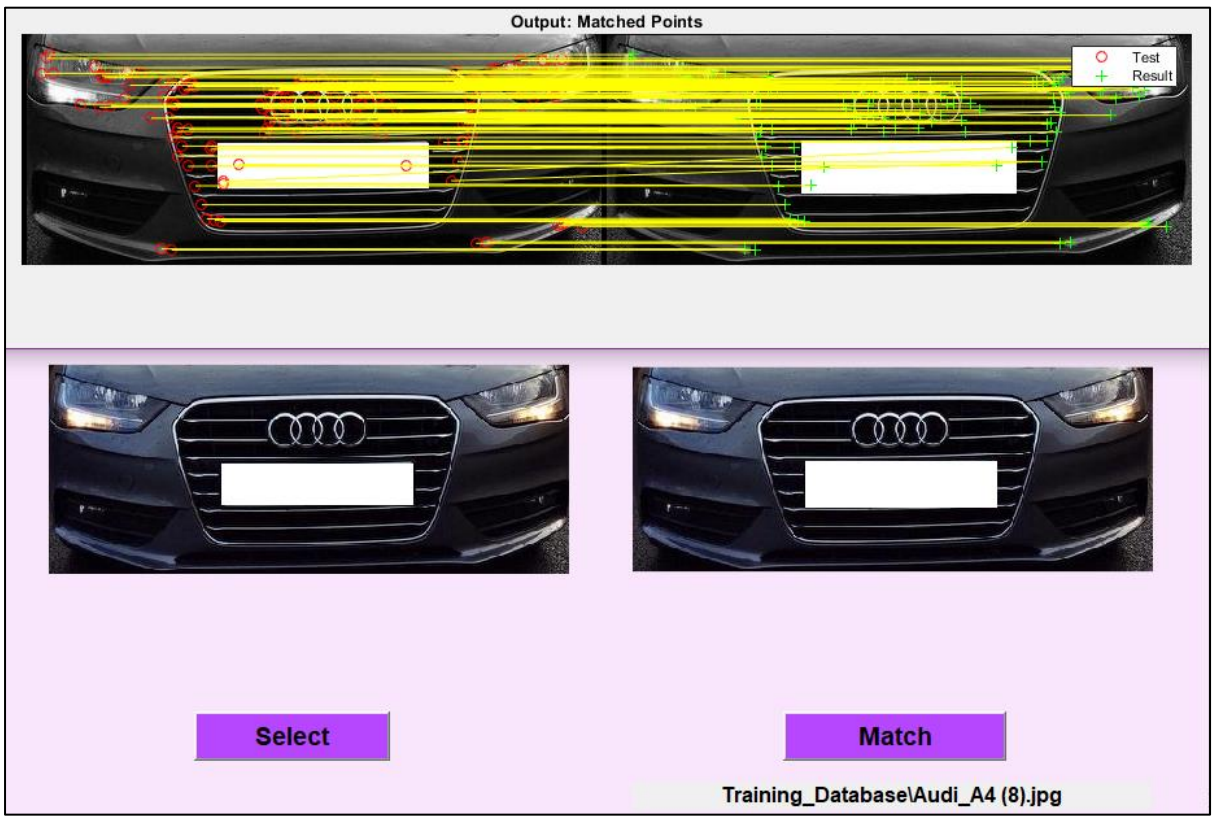


Fig 2.5 Output: Audi A4

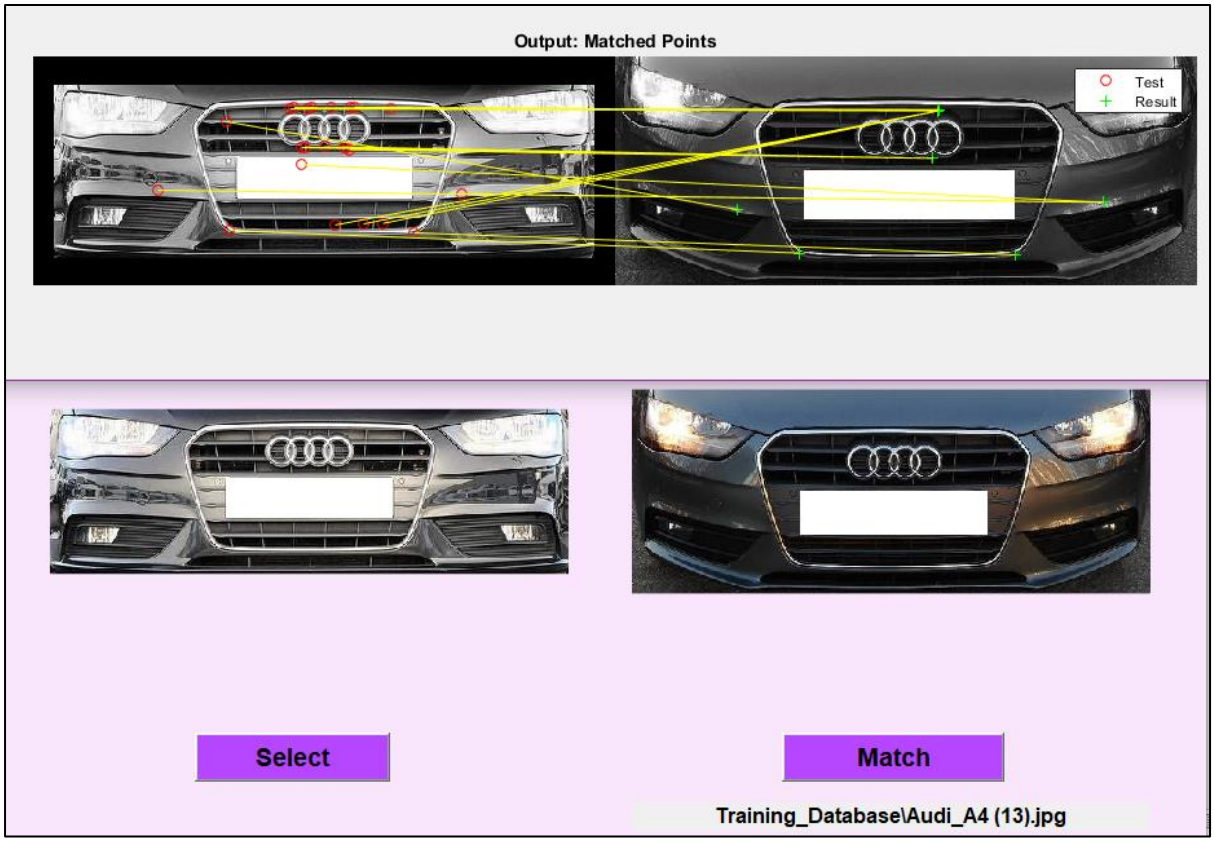


Fig 2.6 Output: Audi A4

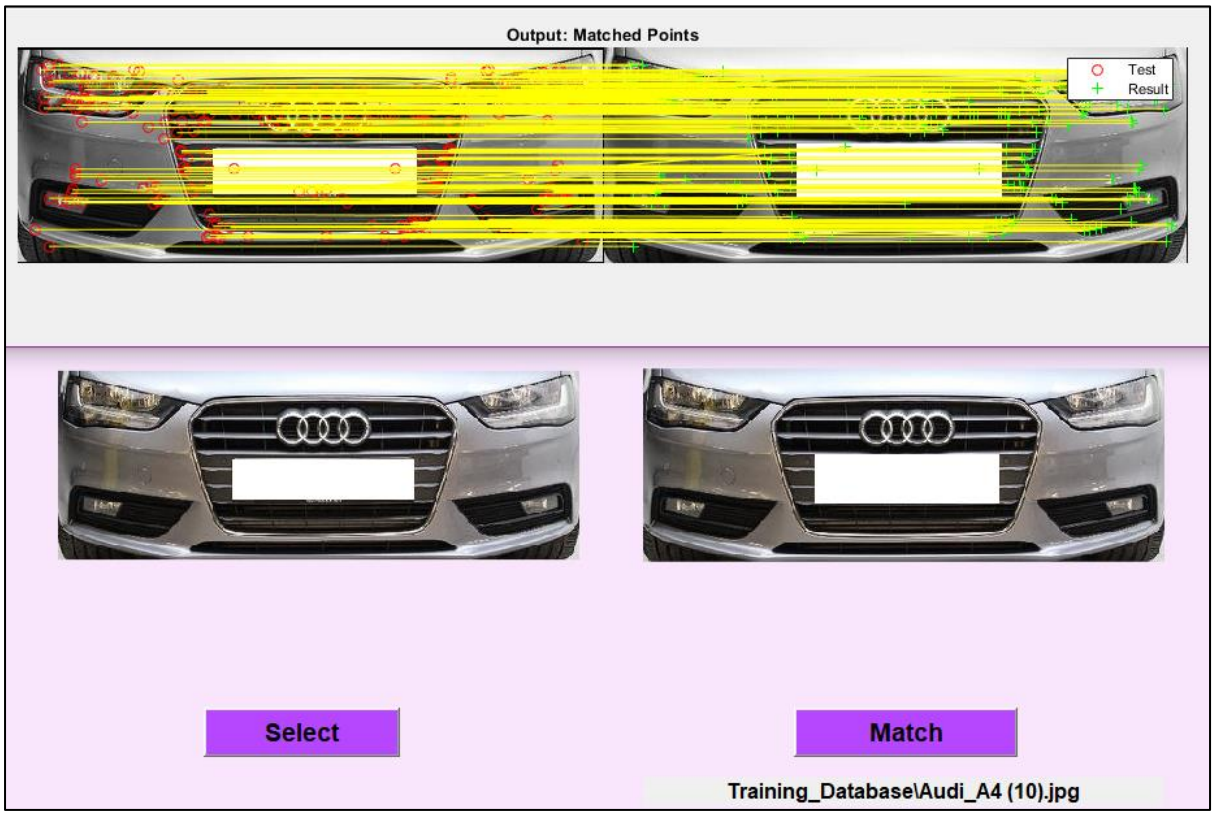


Fig 2.7 Output: Audi A4: Perfect result

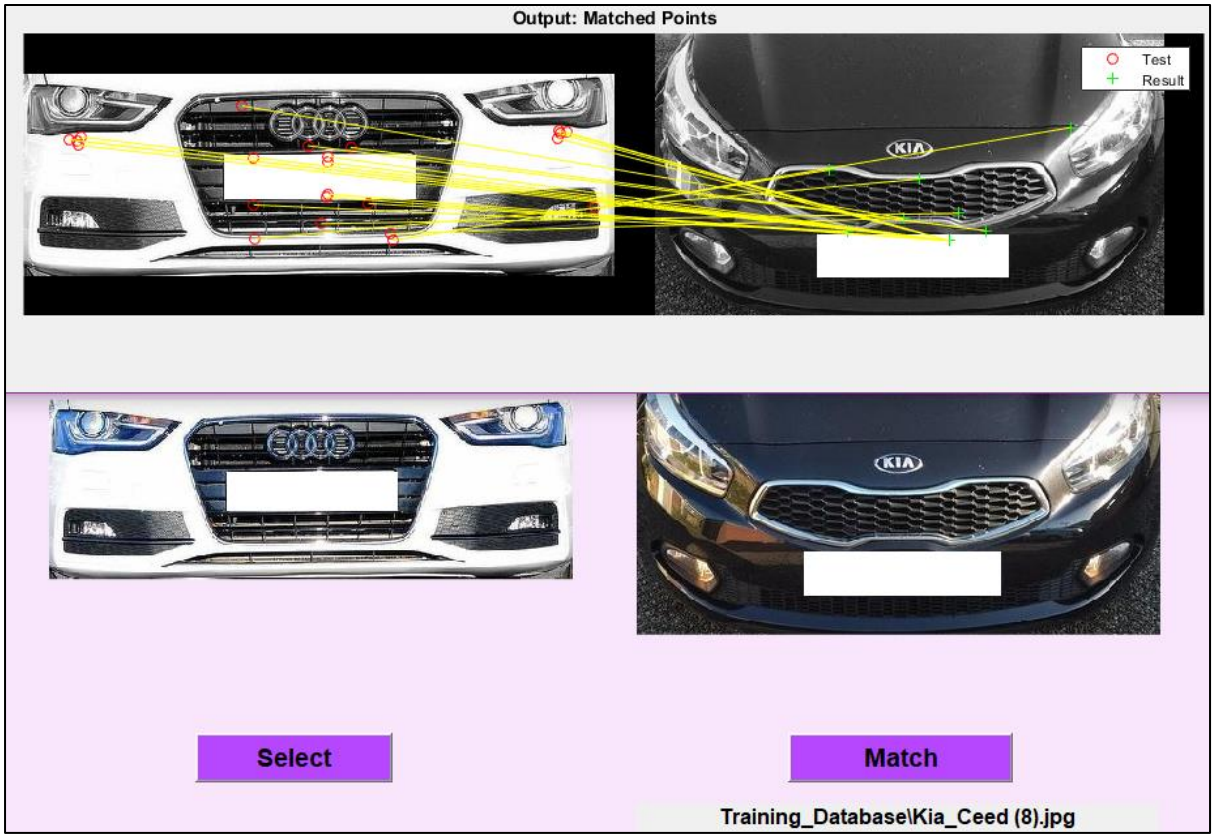


Fig 2.8 Output: **Kia Ceed:** Wrong Result

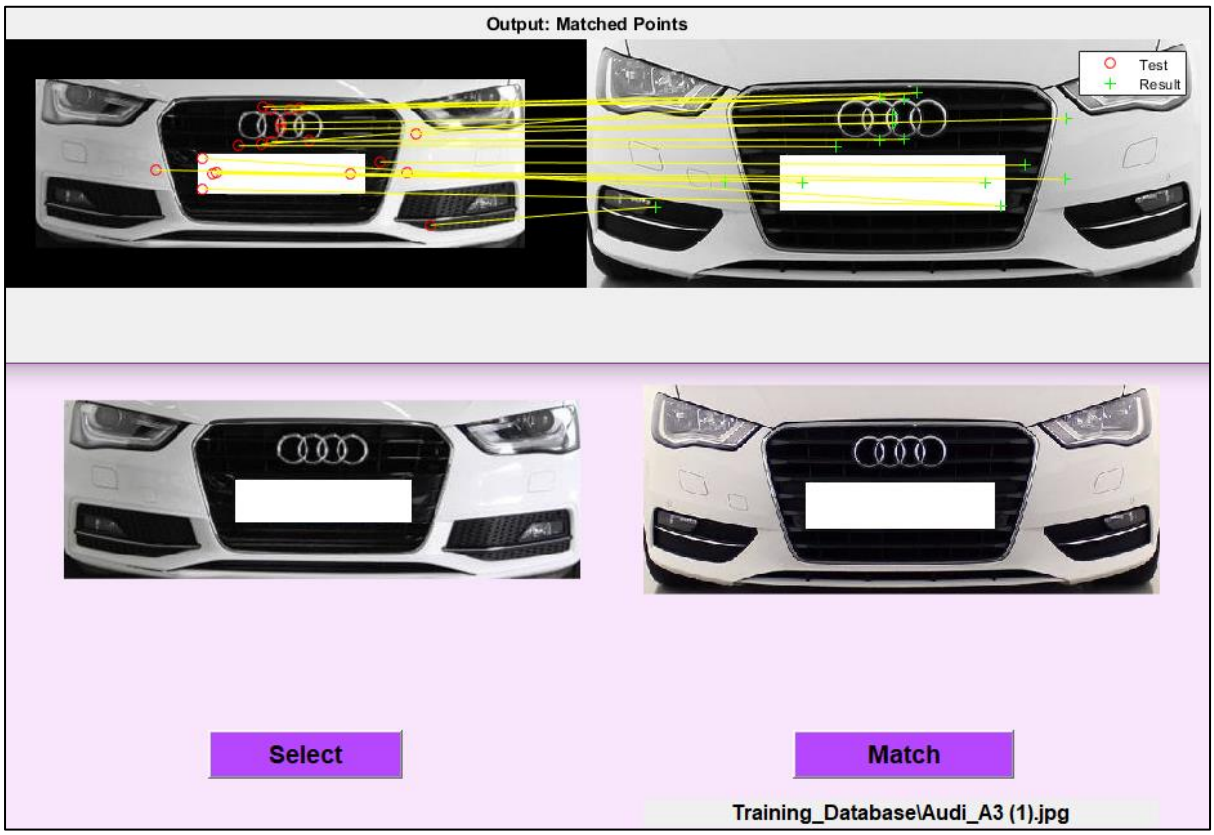


Fig 2.9 Output: **Audi A3:** Wrong Result

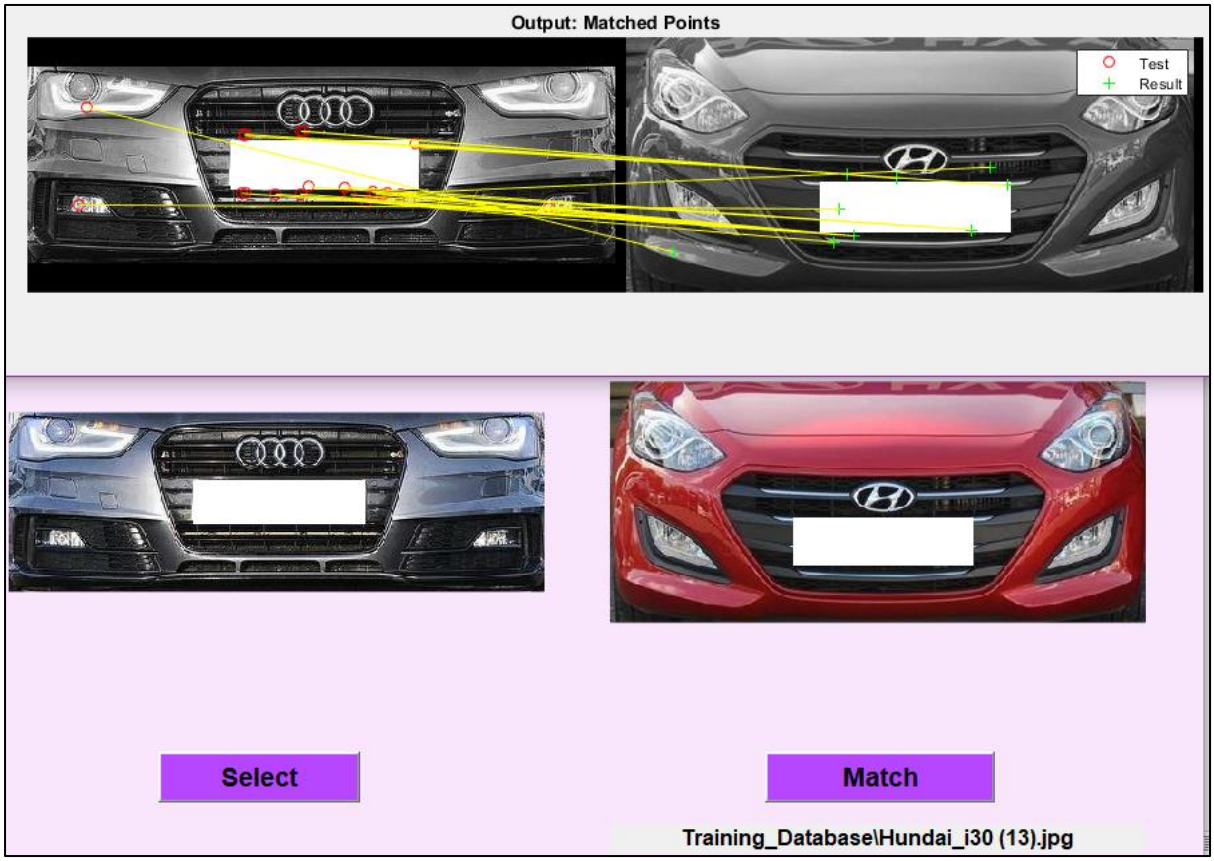


Fig 2.10 Output: **Hundai_i30 :**Wrong result

3. Audi A6

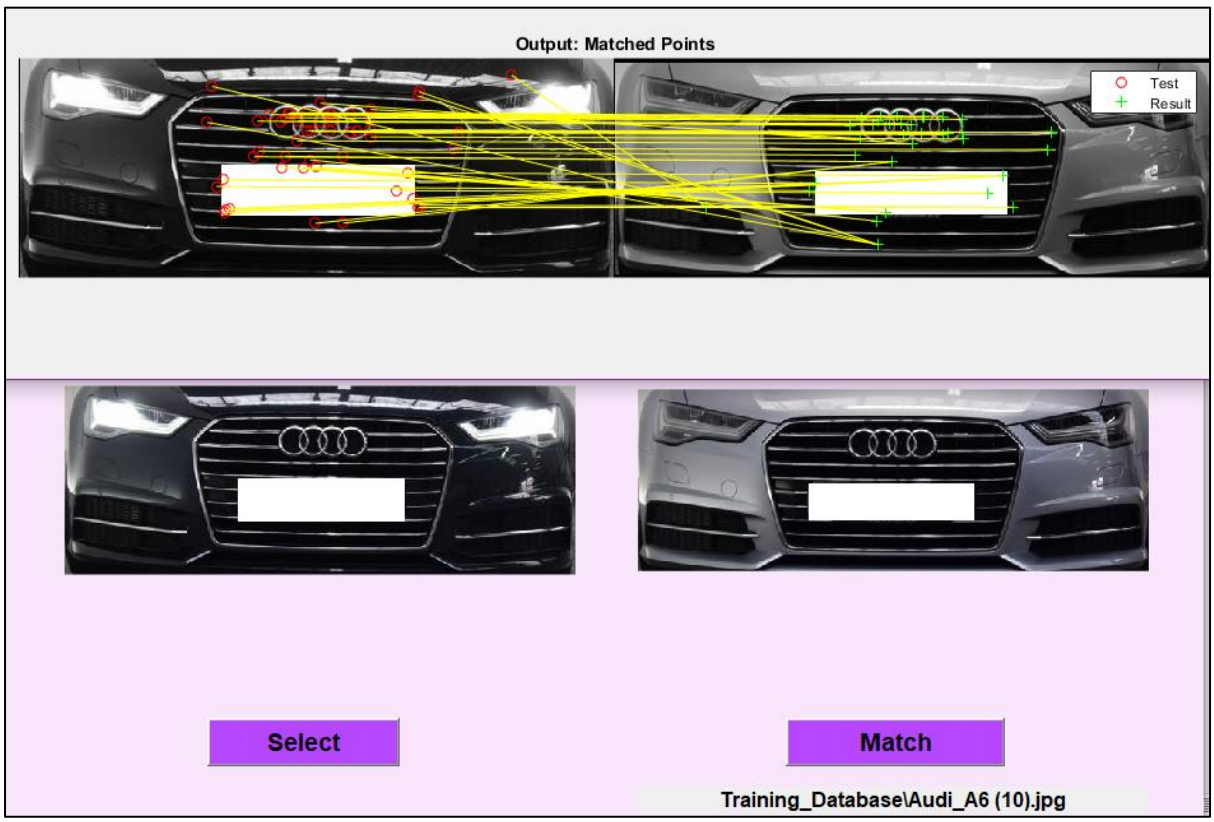


Fig 3.1 Output: Audi A6

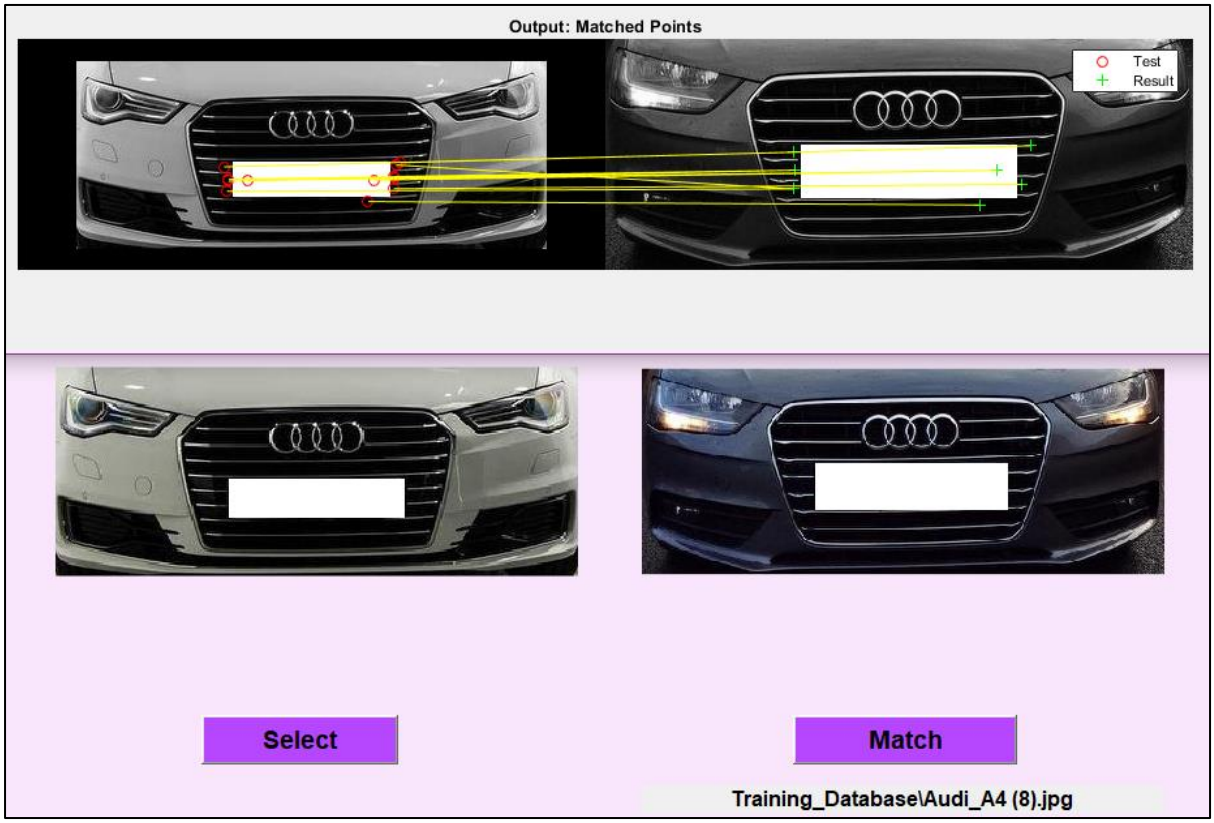


Fig 3.2 Output: **Audi A4**: Wrong Result

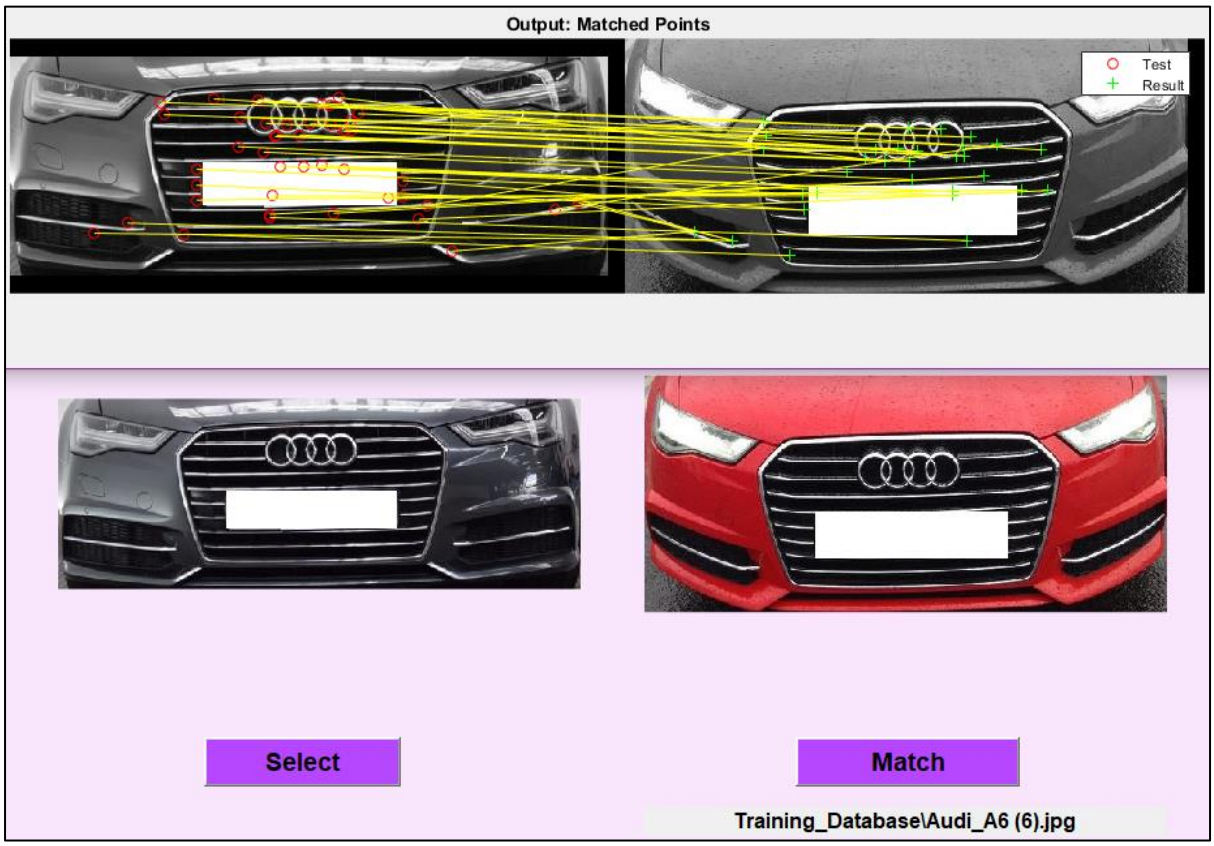


Fig 3.3 Output: Audi A6

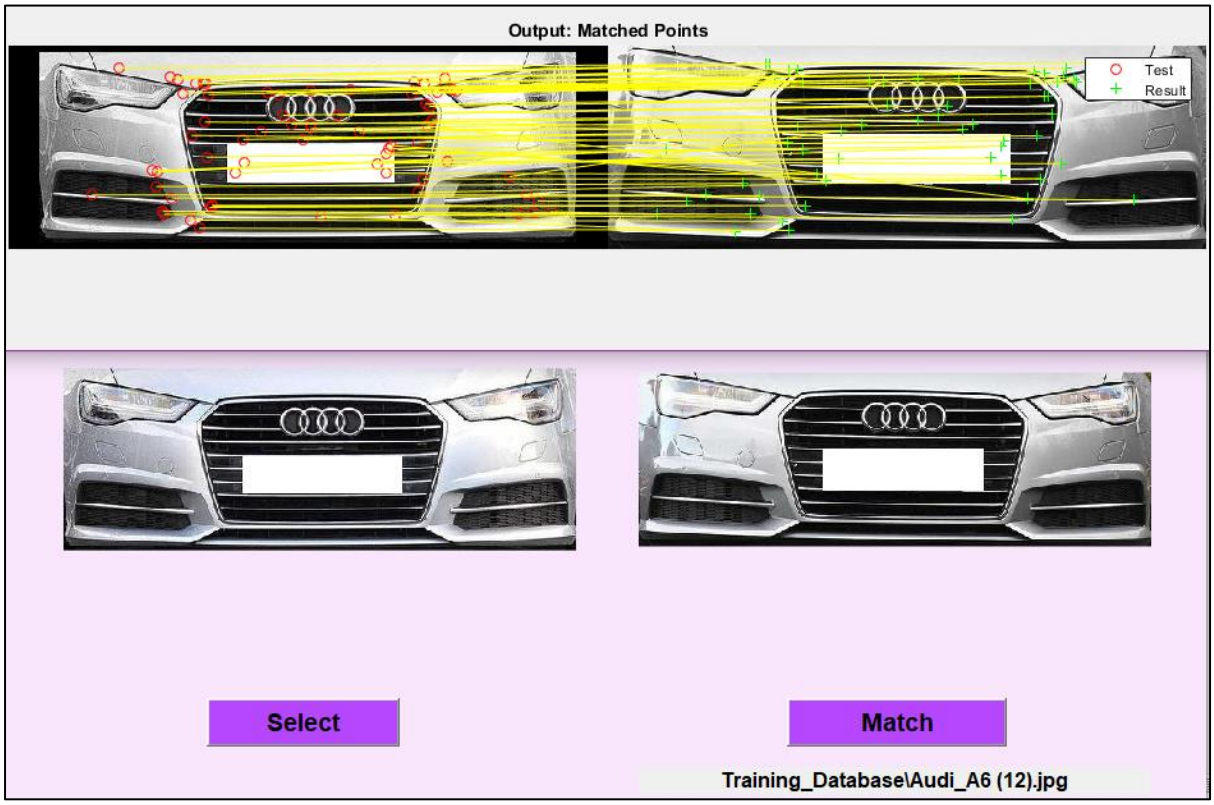


Fig 3.4 Output: Audi A3

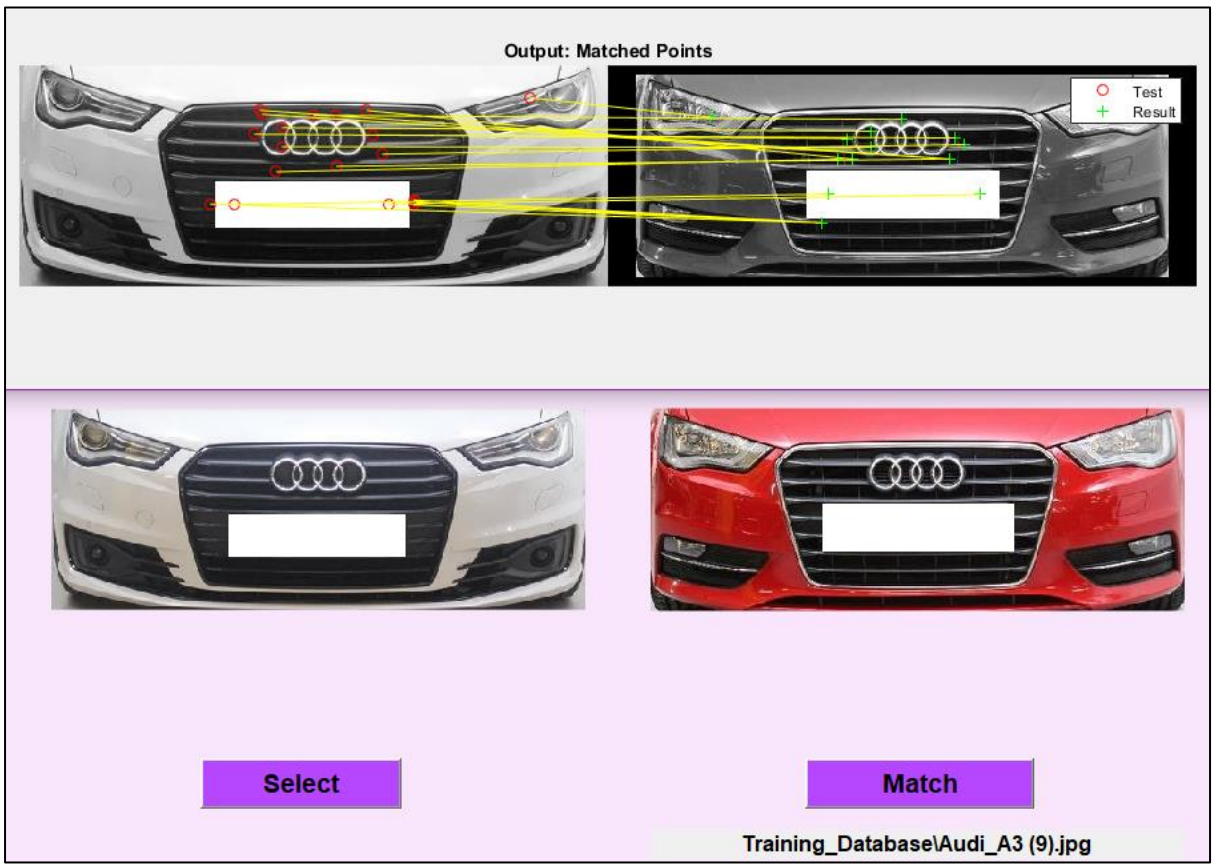


Fig 3.5 Output: **Audi A3** : Wrong Result

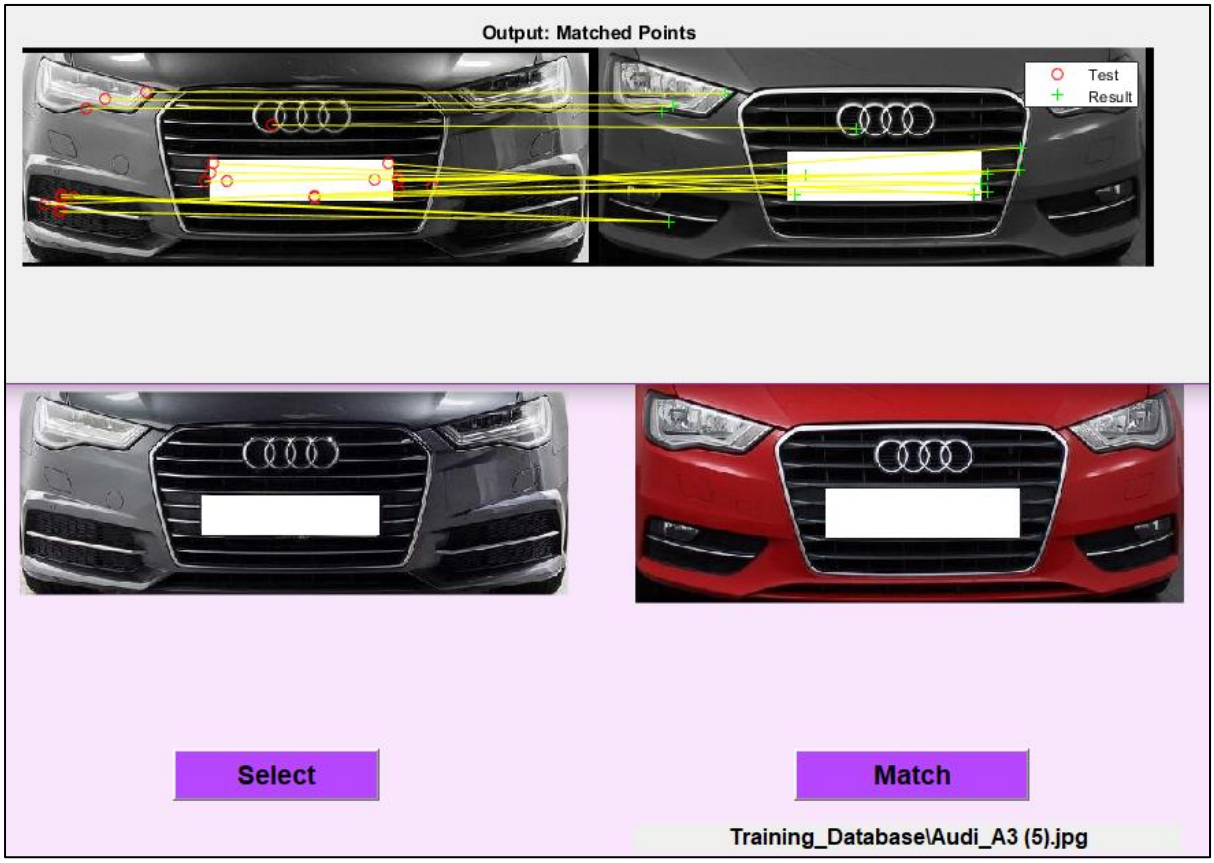


Fig 3.6 Output: **Audi A3** : Wrong Result

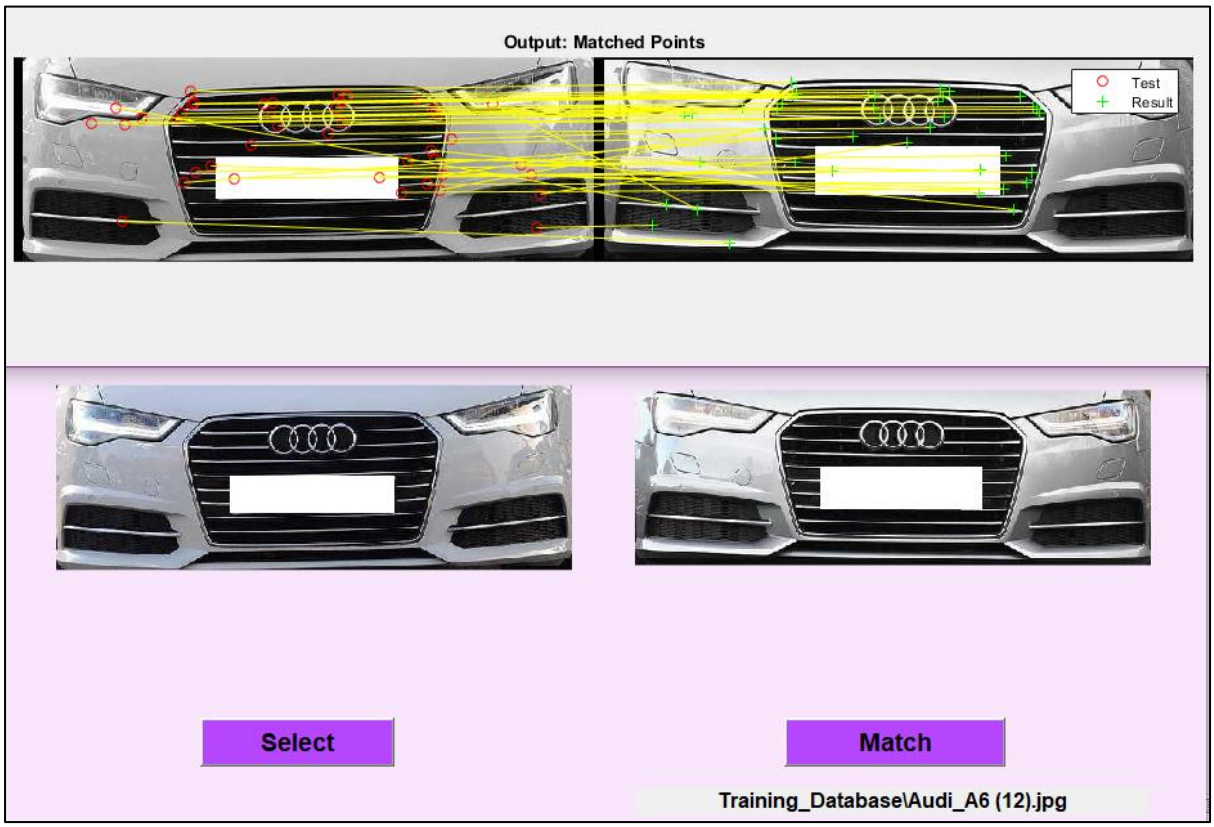


Fig 3.7 Output: Audi A6

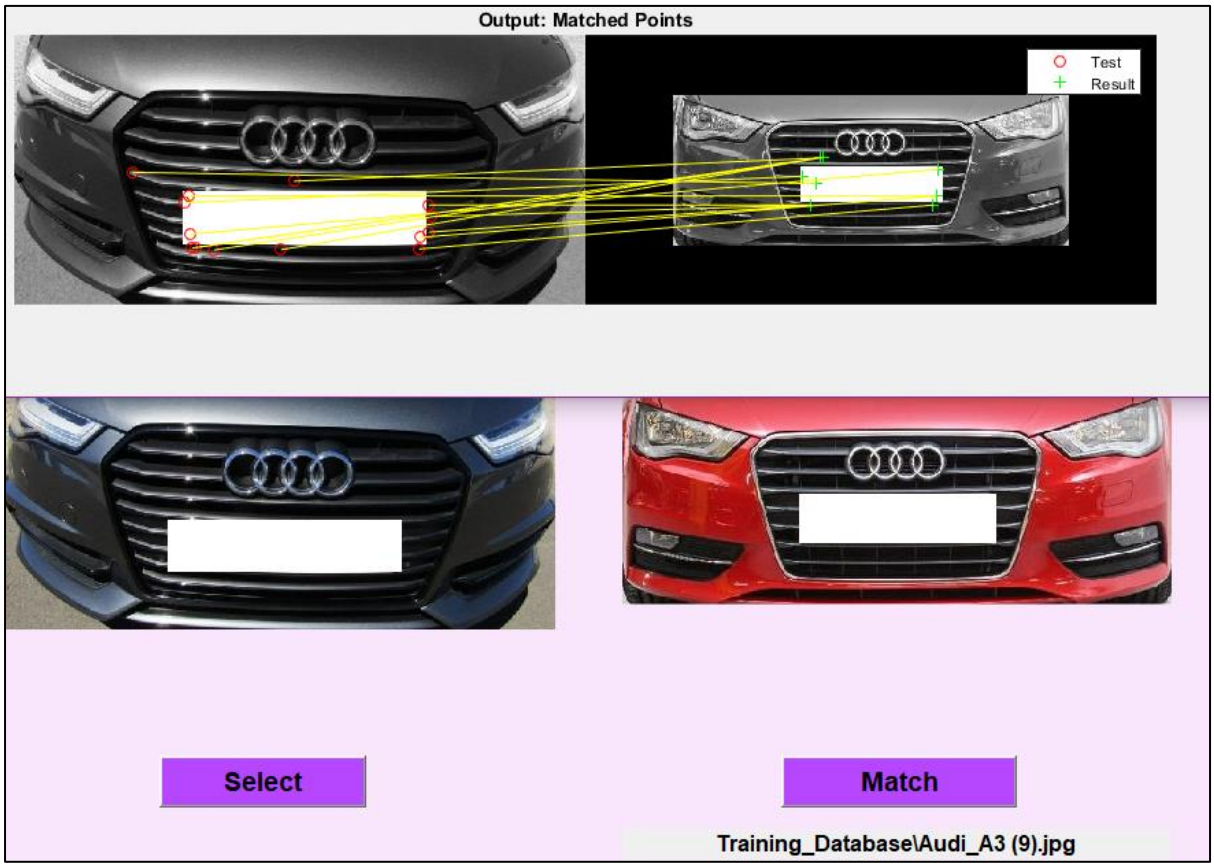


Fig 3.8 Output: **Audi A3**: Wrong Result

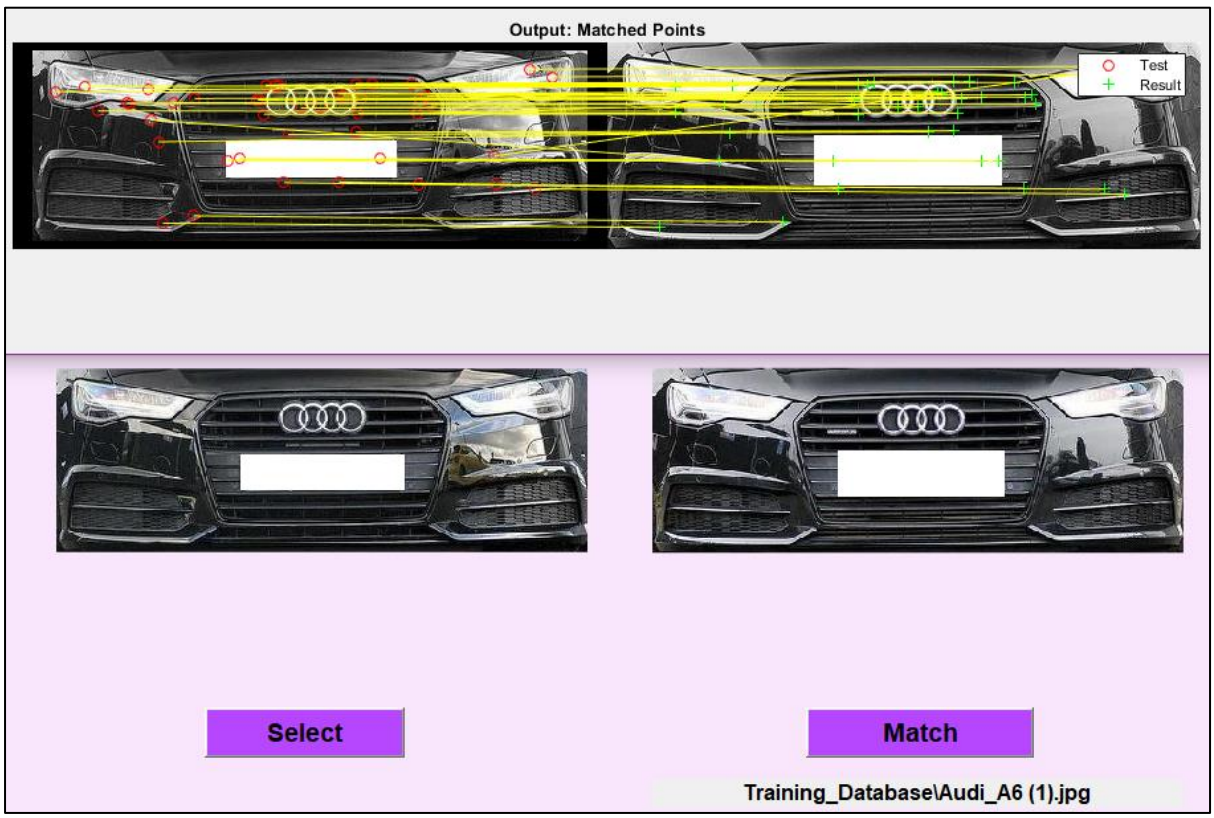


Fig 3.9 Output: Audi A6

Fig 3.10 Output: **Audi A4 :** Wrong Result

4. BMW 3

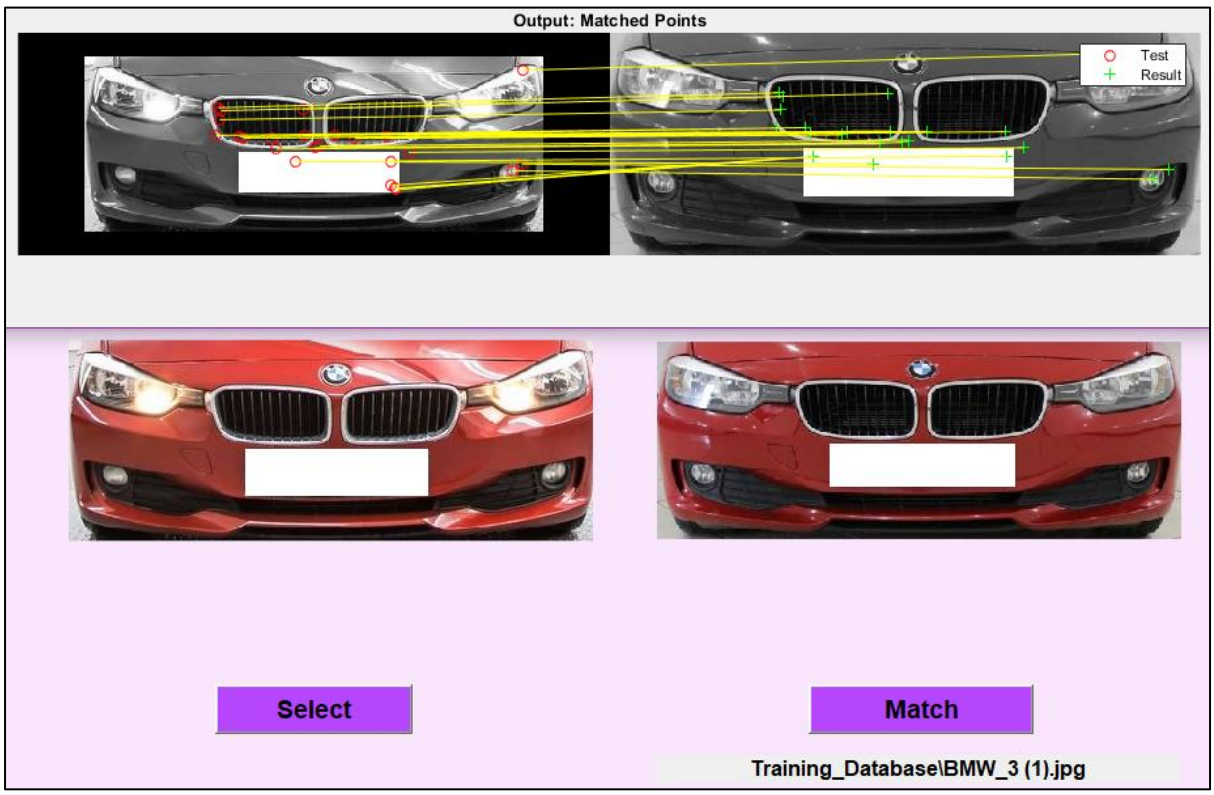


Fig 4.1 Output: BMW 3

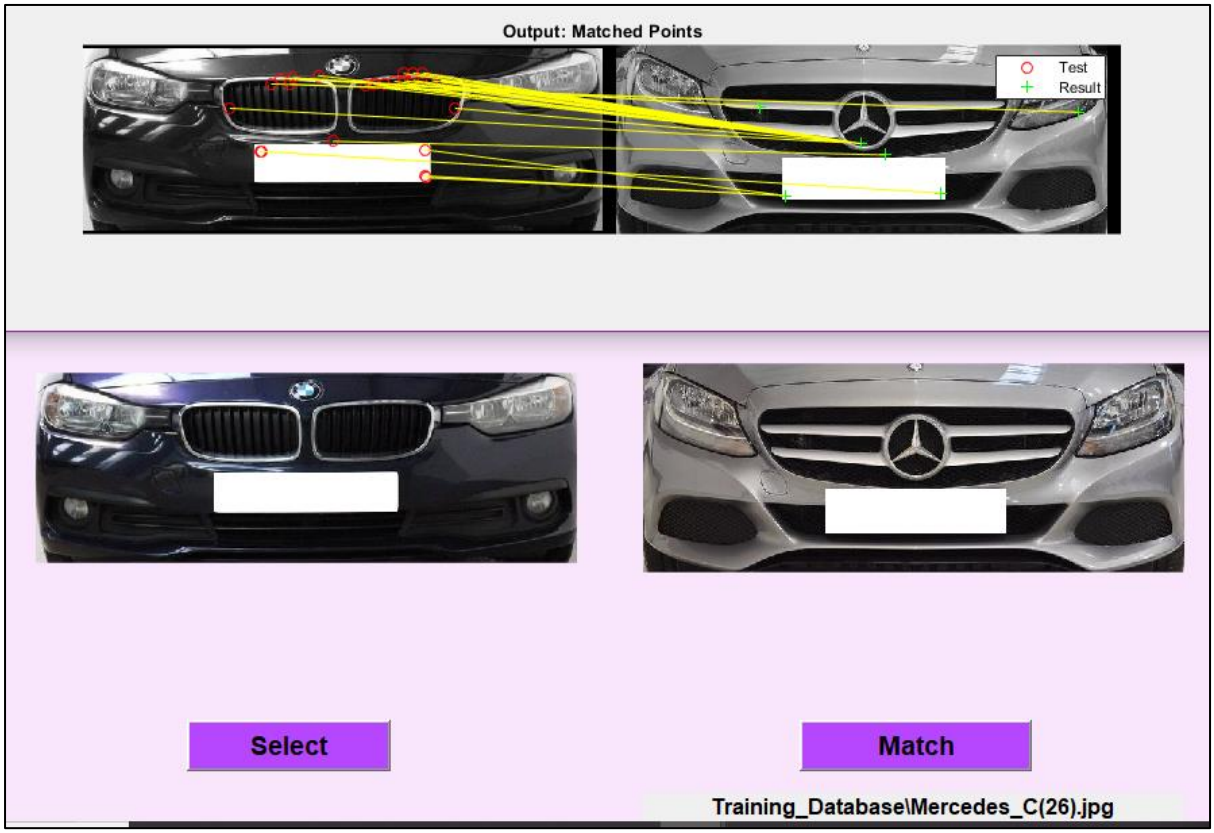


Fig 4.2 Output: **Mercedes C** wrong result

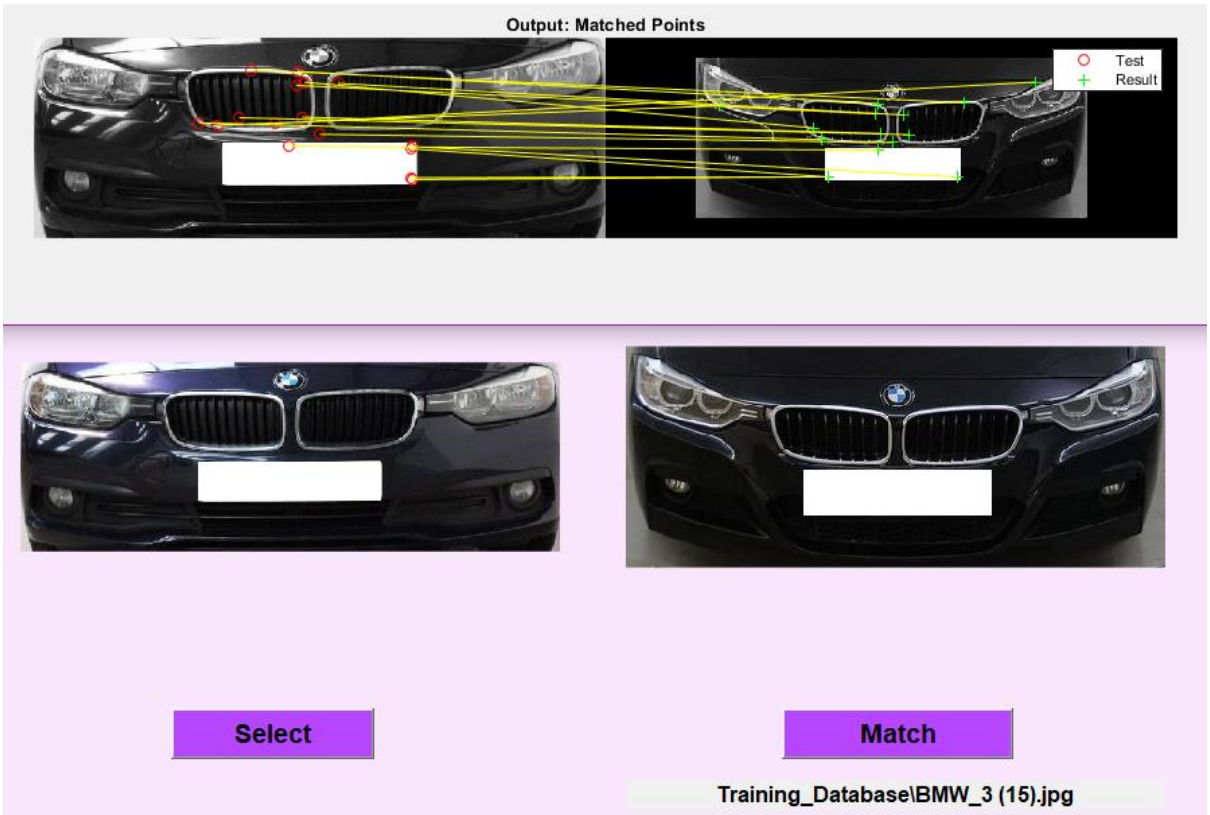


Fig 4.3 Output: BMW 3

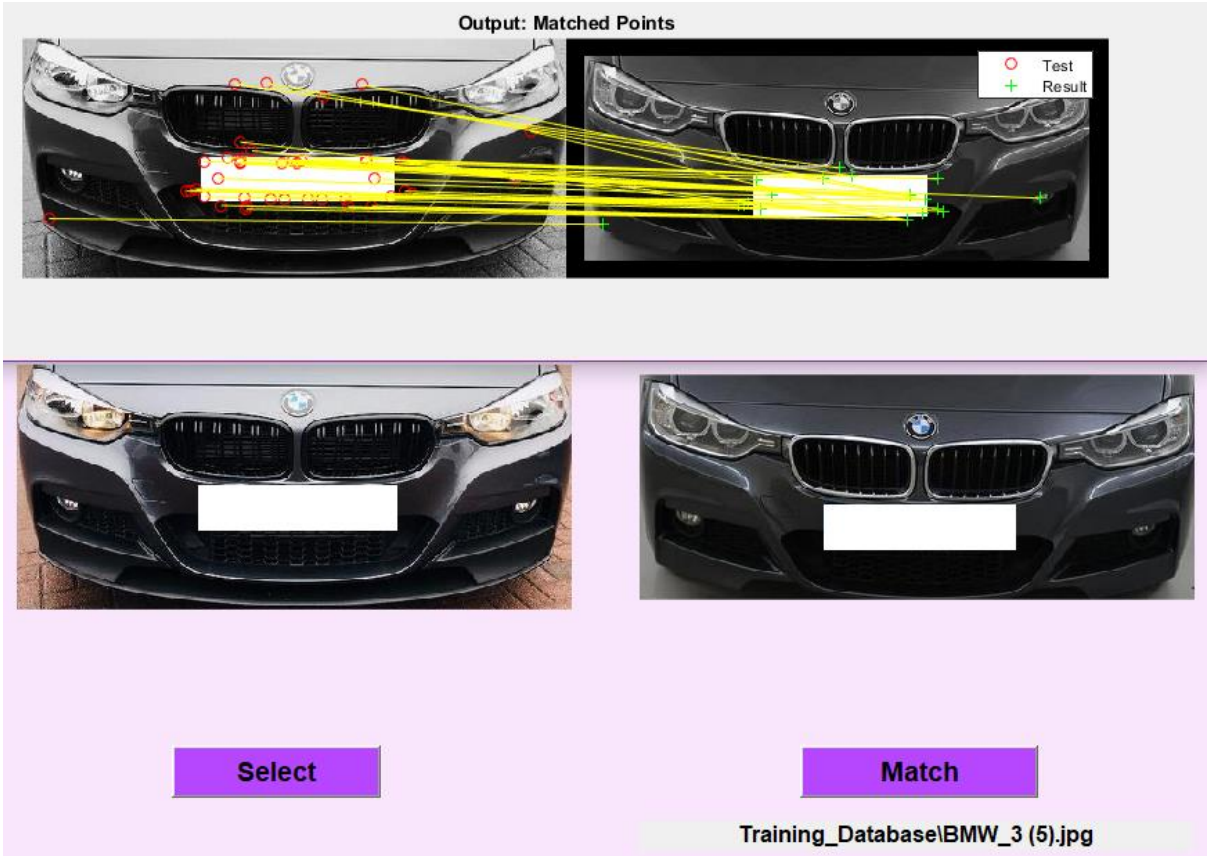


Fig 4.4 Output: BMW 3

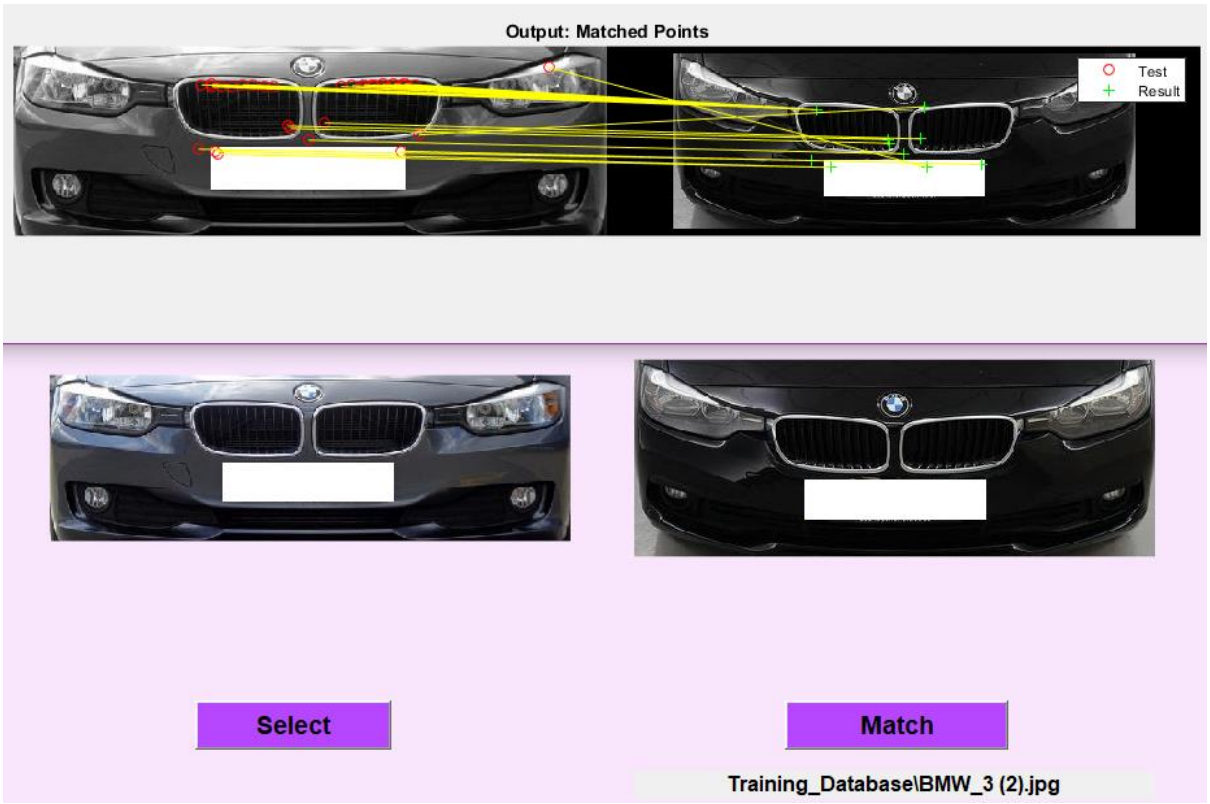


Fig 4.5 Output: BMW 3

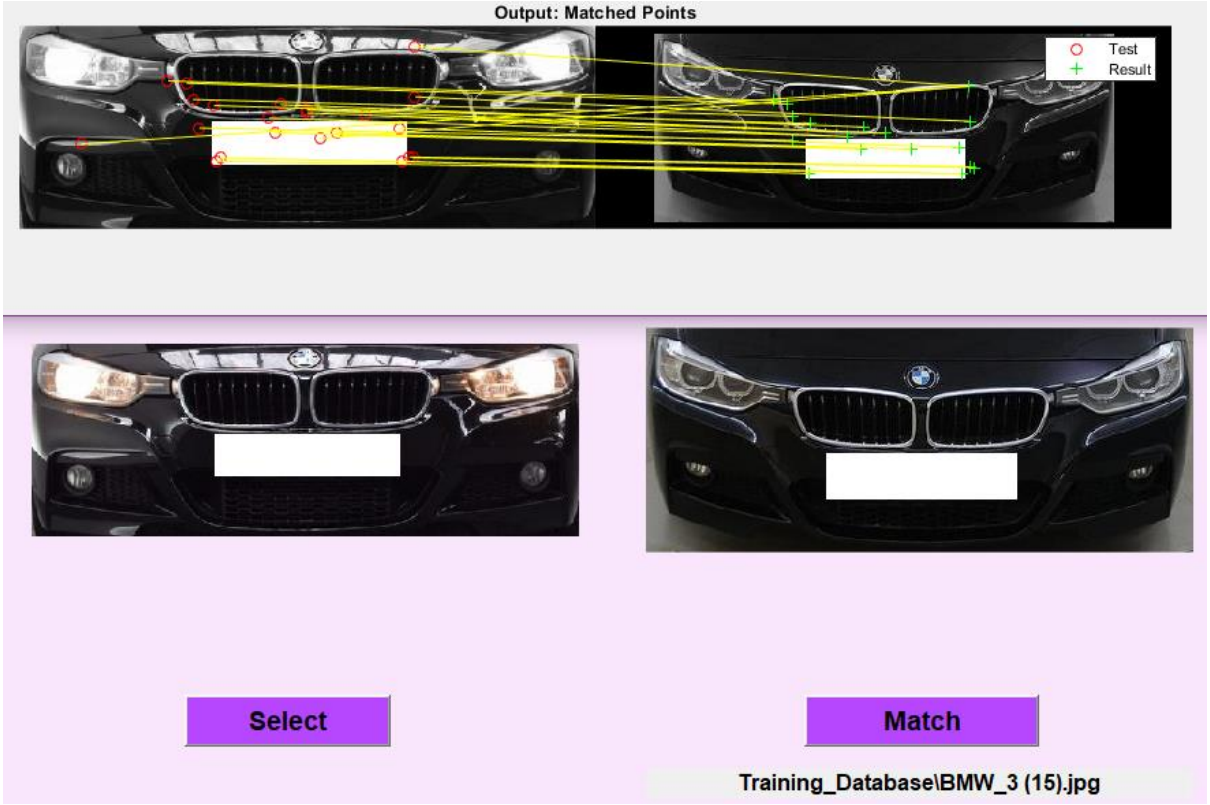


Fig 4.6 Output: BMW 3

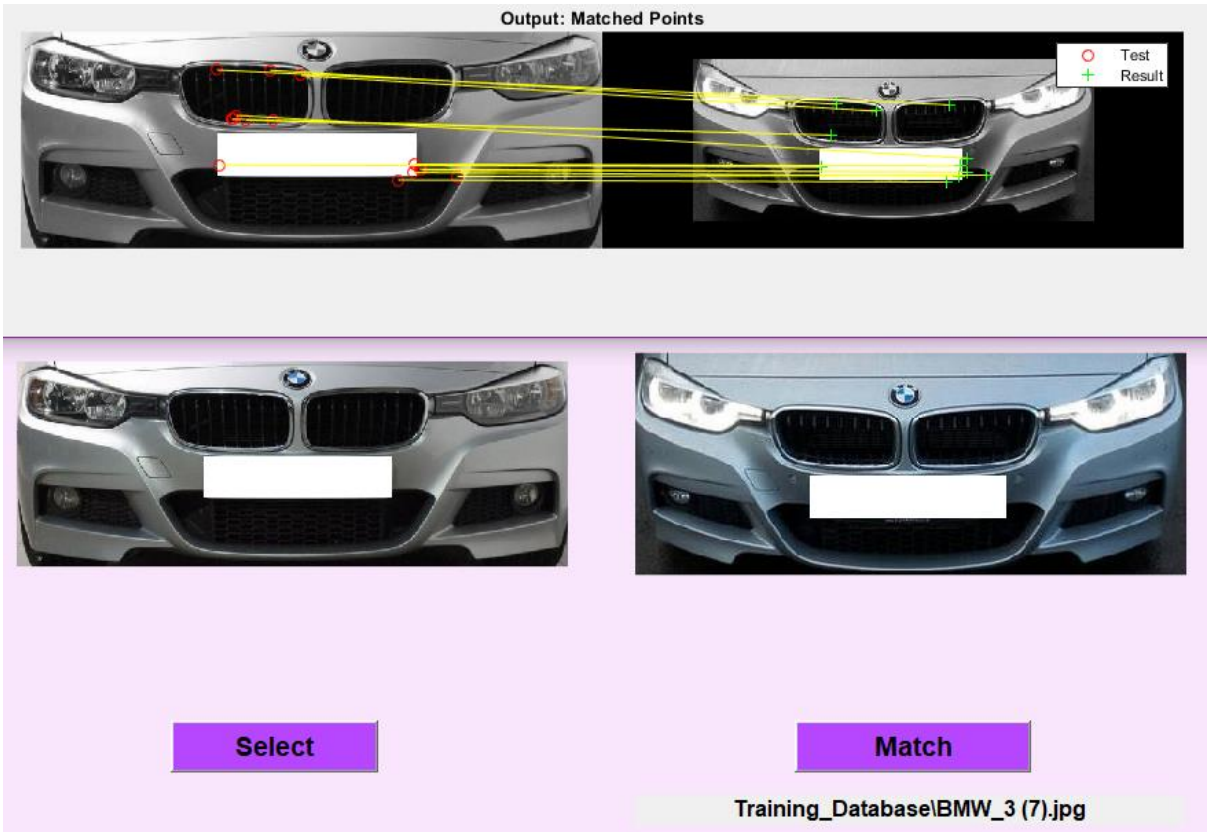


Fig 4.7 Output: BMW 3

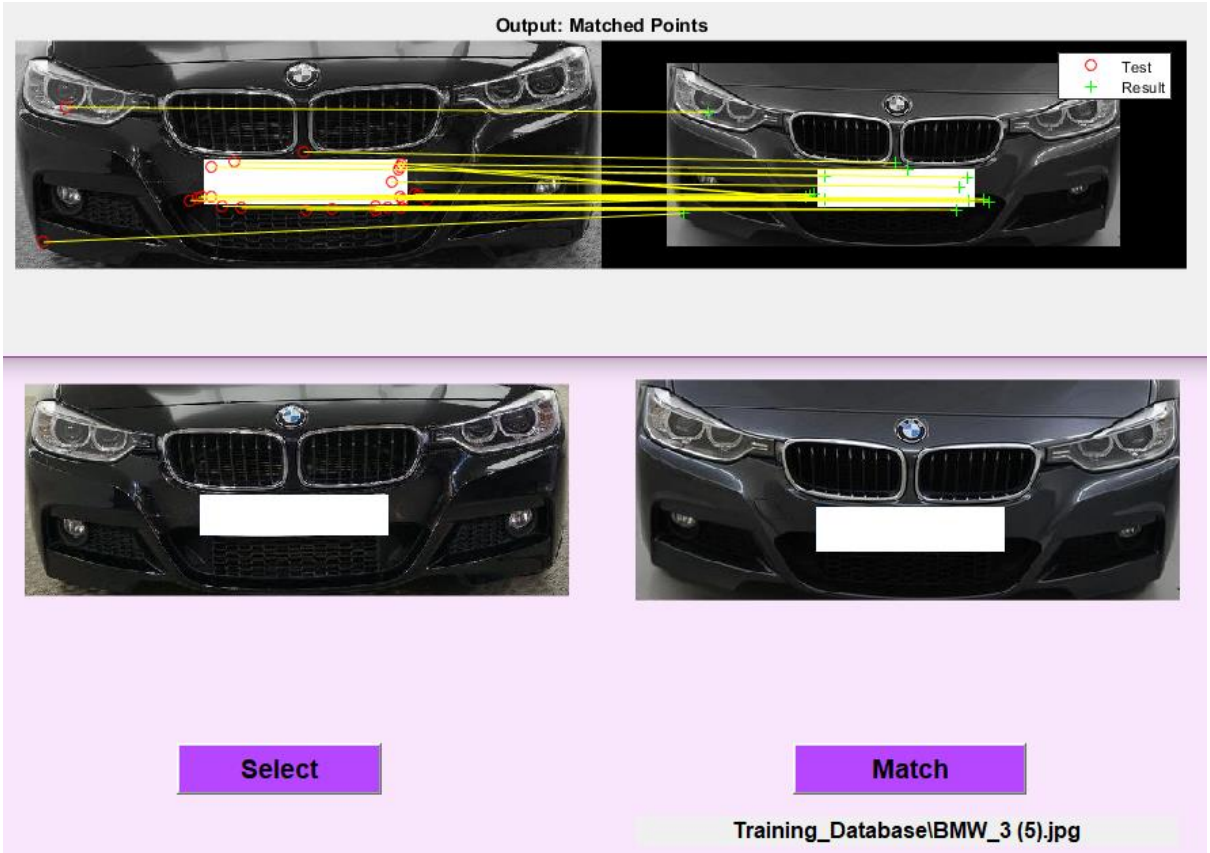


Fig 4.8 Output: BMW 3

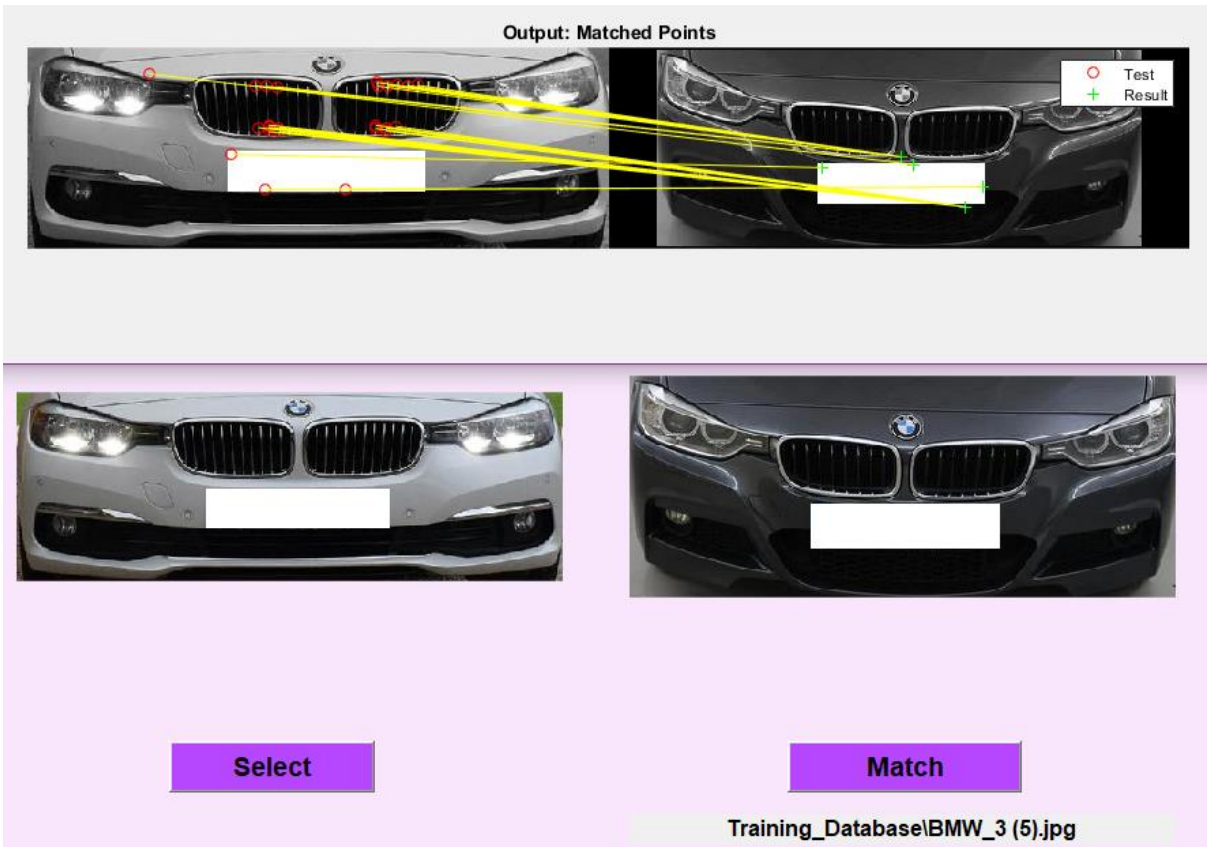


Fig 4.9 Output: BMW 3

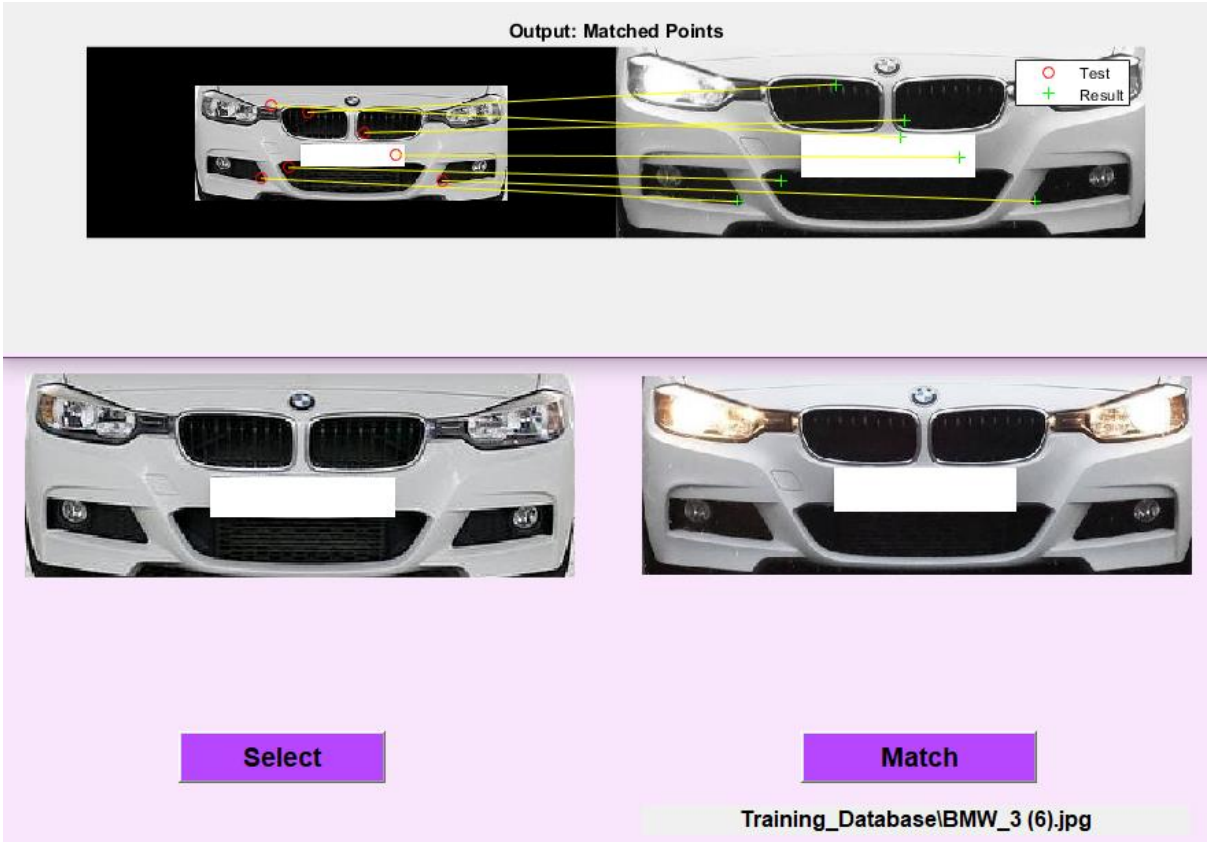


Fig 4.10 Output: BMW 3

5. BMW 5

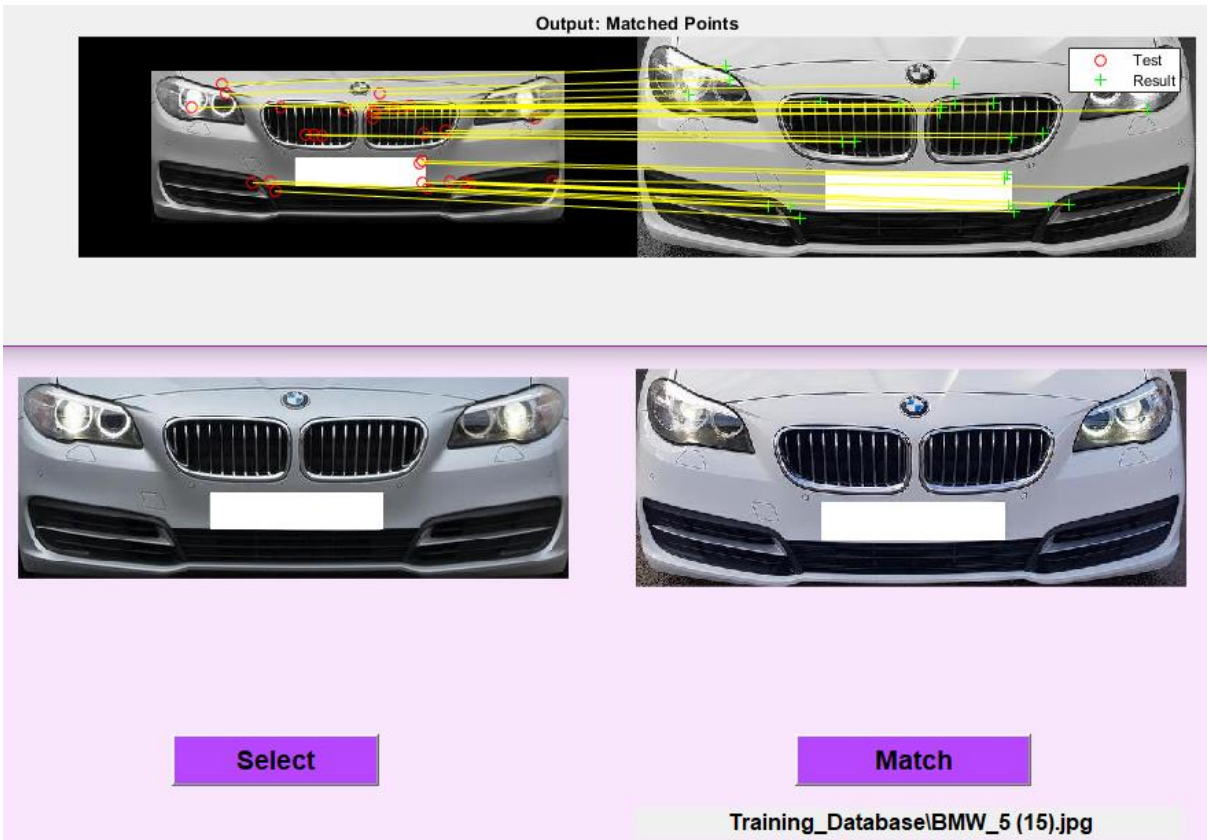


Fig 5.1 Output: BMW 5

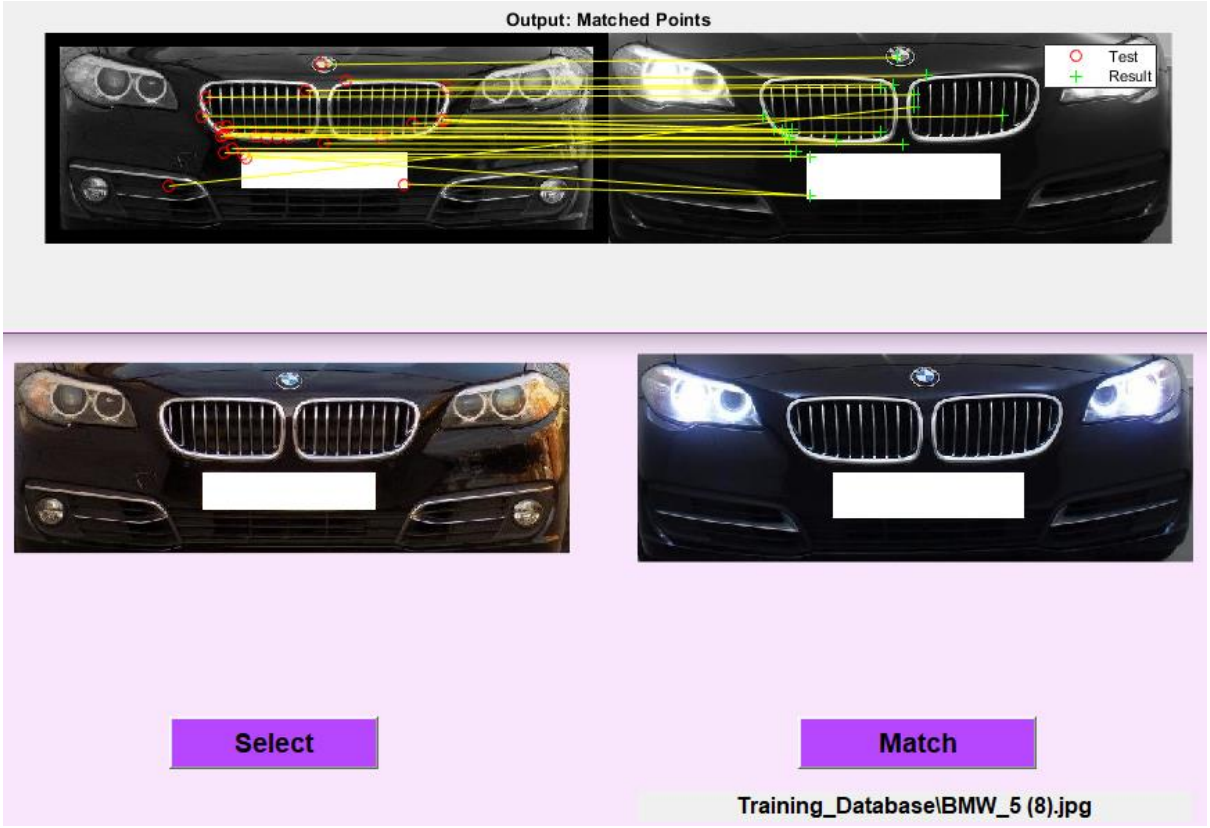


Fig 5.2 Output: BMW 5

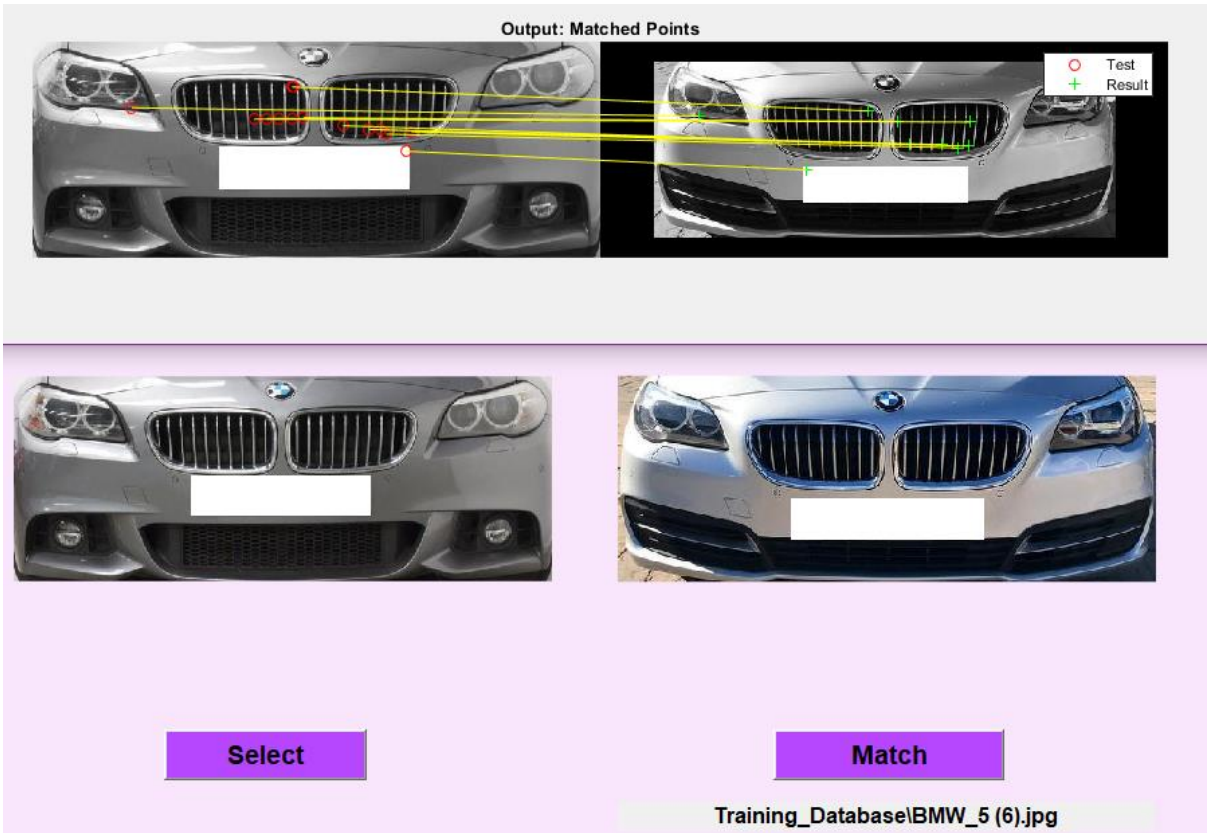


Fig 5.3 Output: BMW 5

Fig 5.4 Output: BMW 5

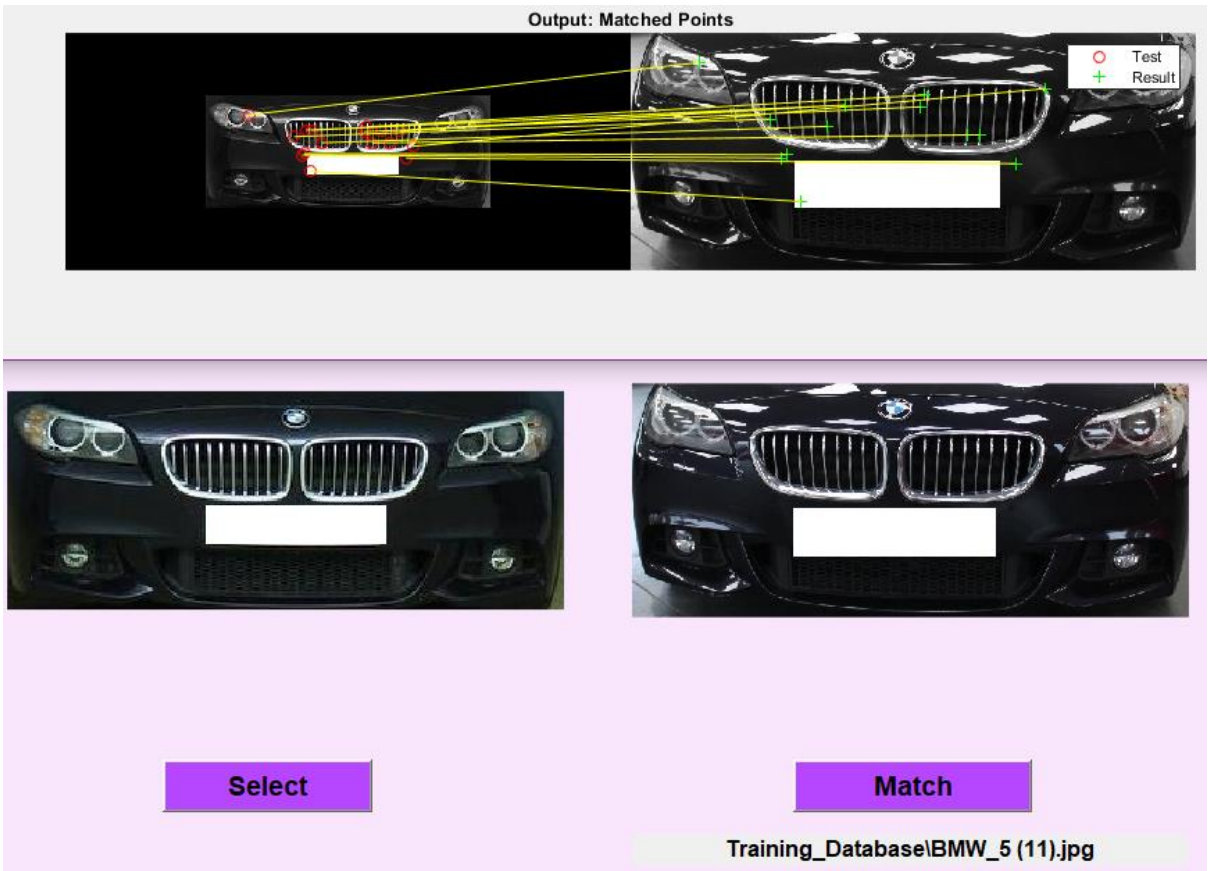


Fig 5.5 Output: BMW 5

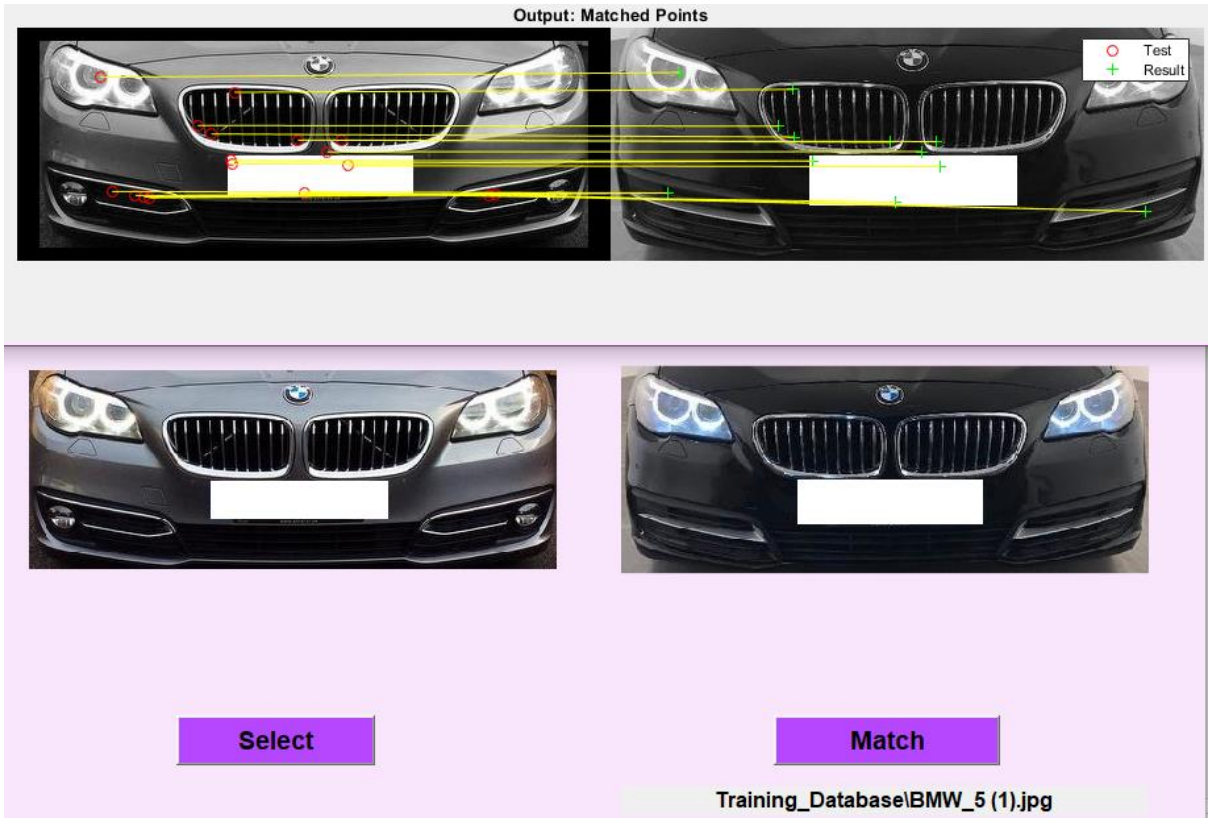


Fig 5.6 Output: BMW

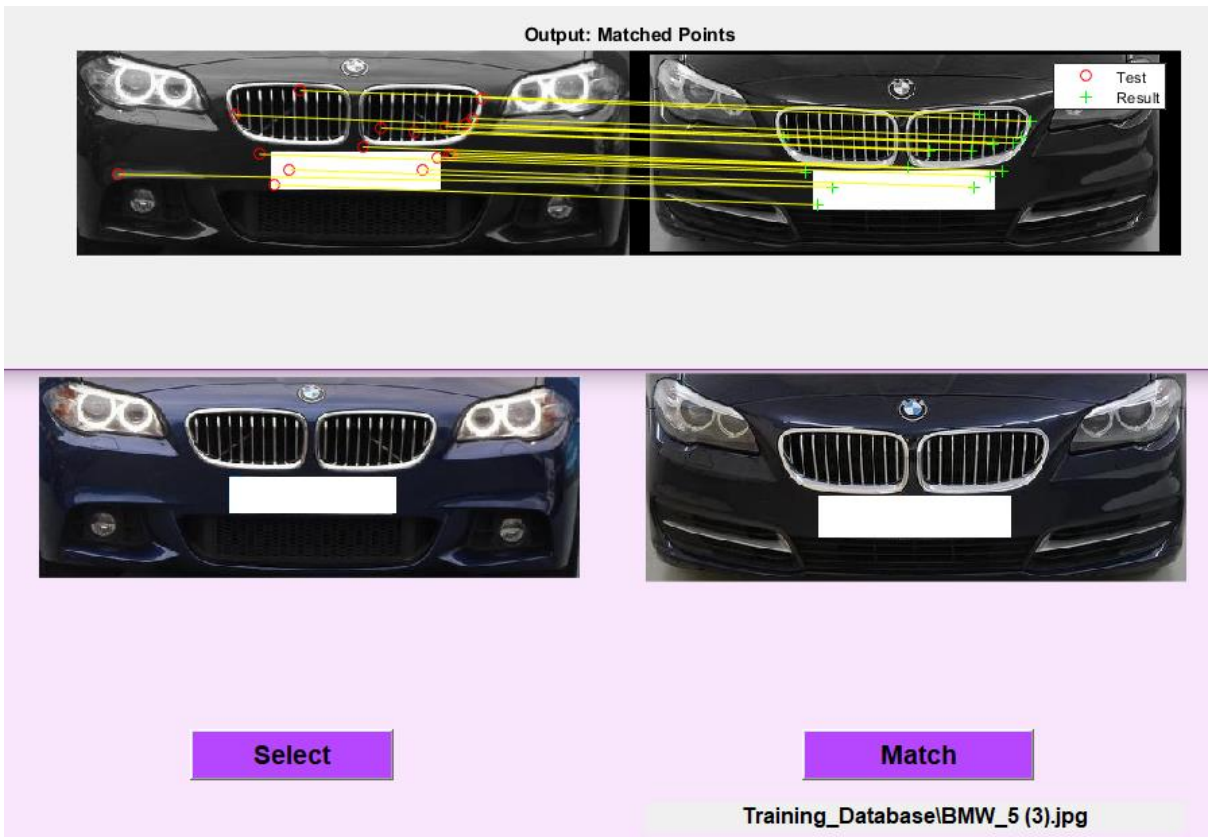


Fig 5.7 Output: BMW 5

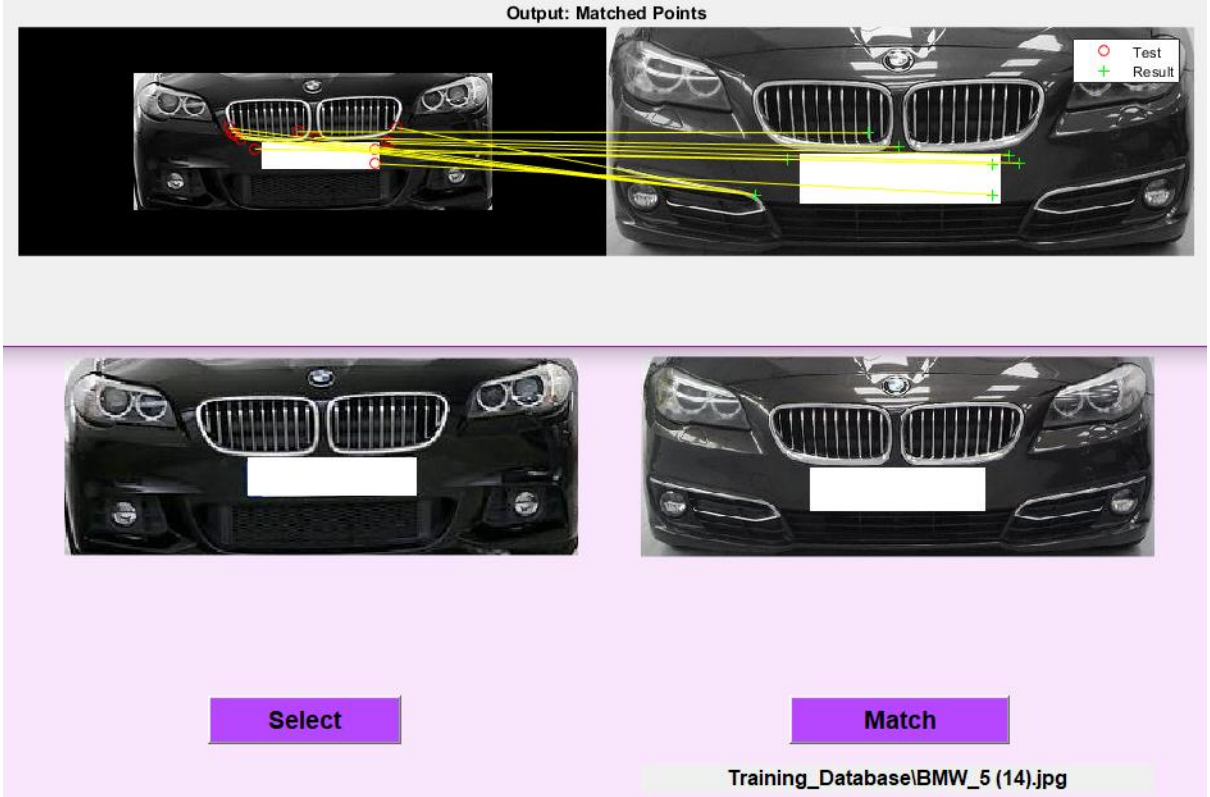


Fig 5.8 Output: BMW 5

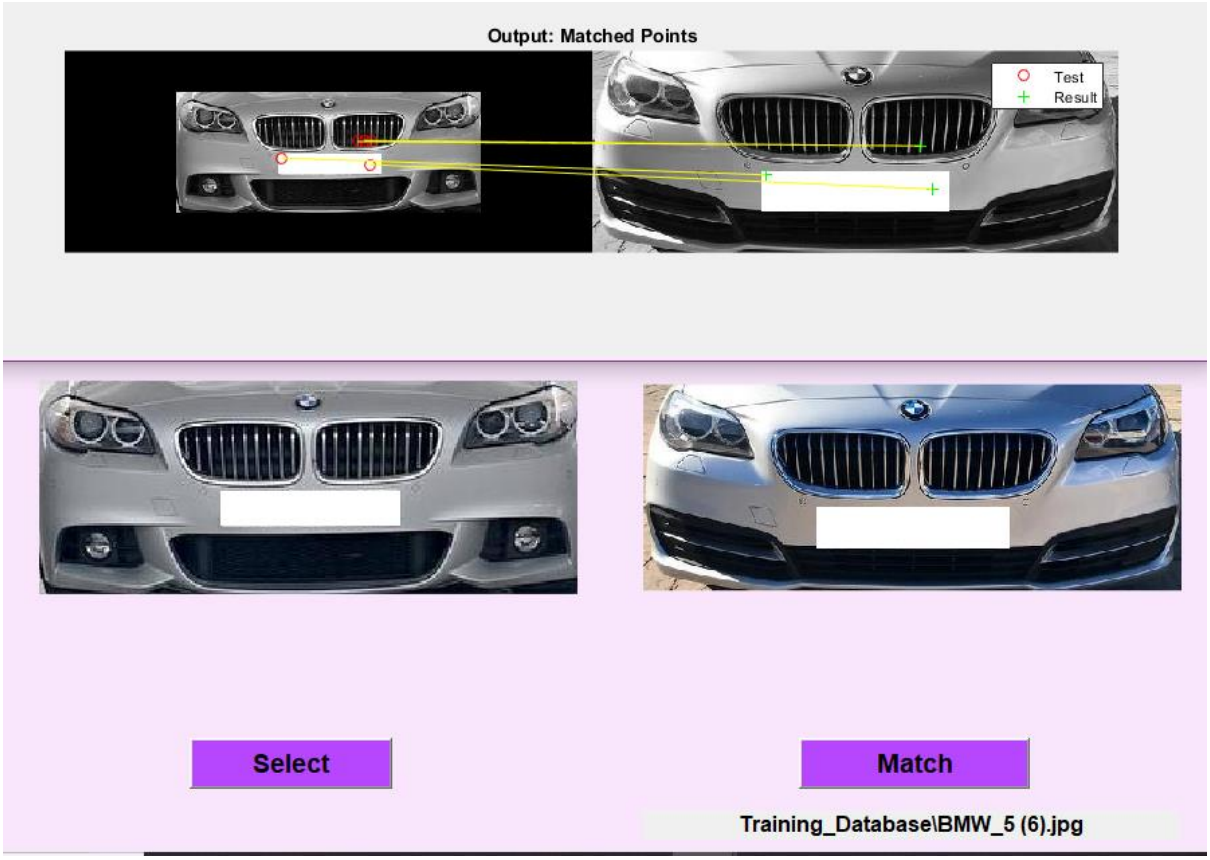


Fig 5.9 Output: BMW 5

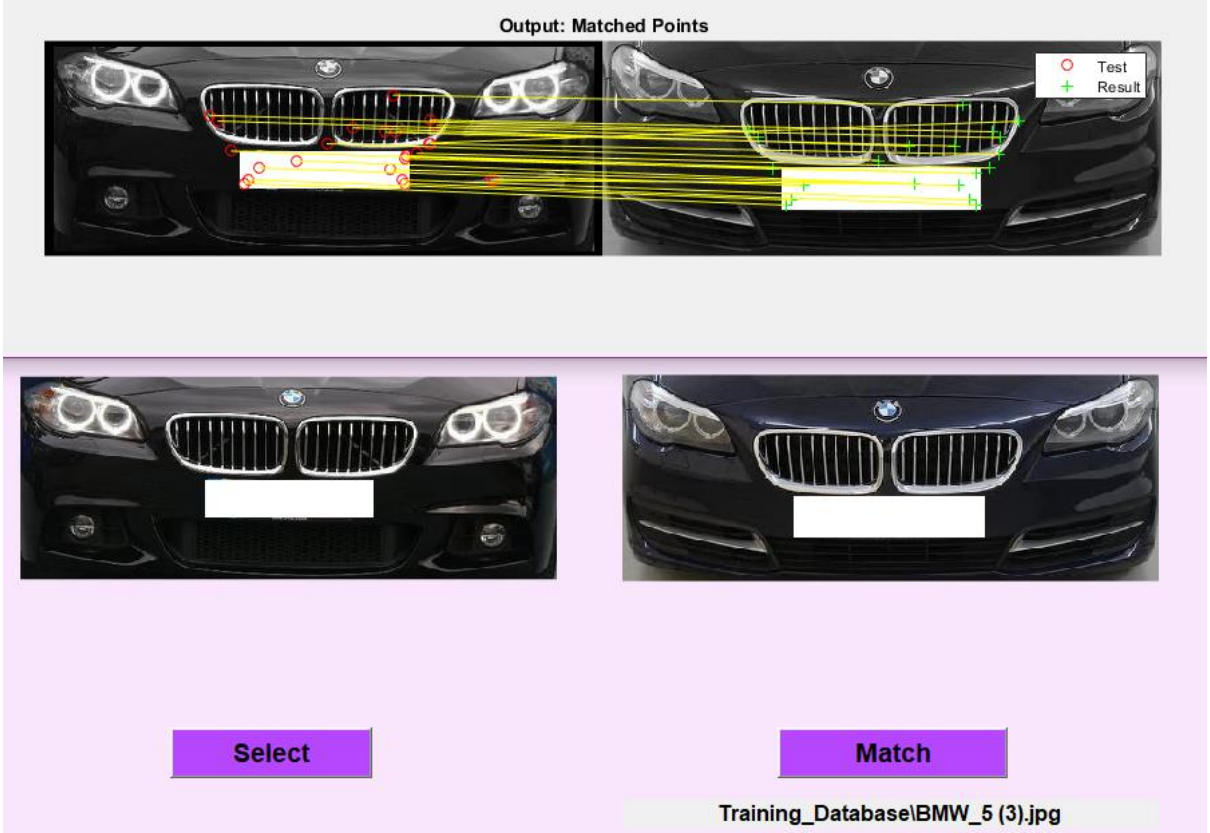


Fig 5.10 Output: BMW 5

6. BMW 6

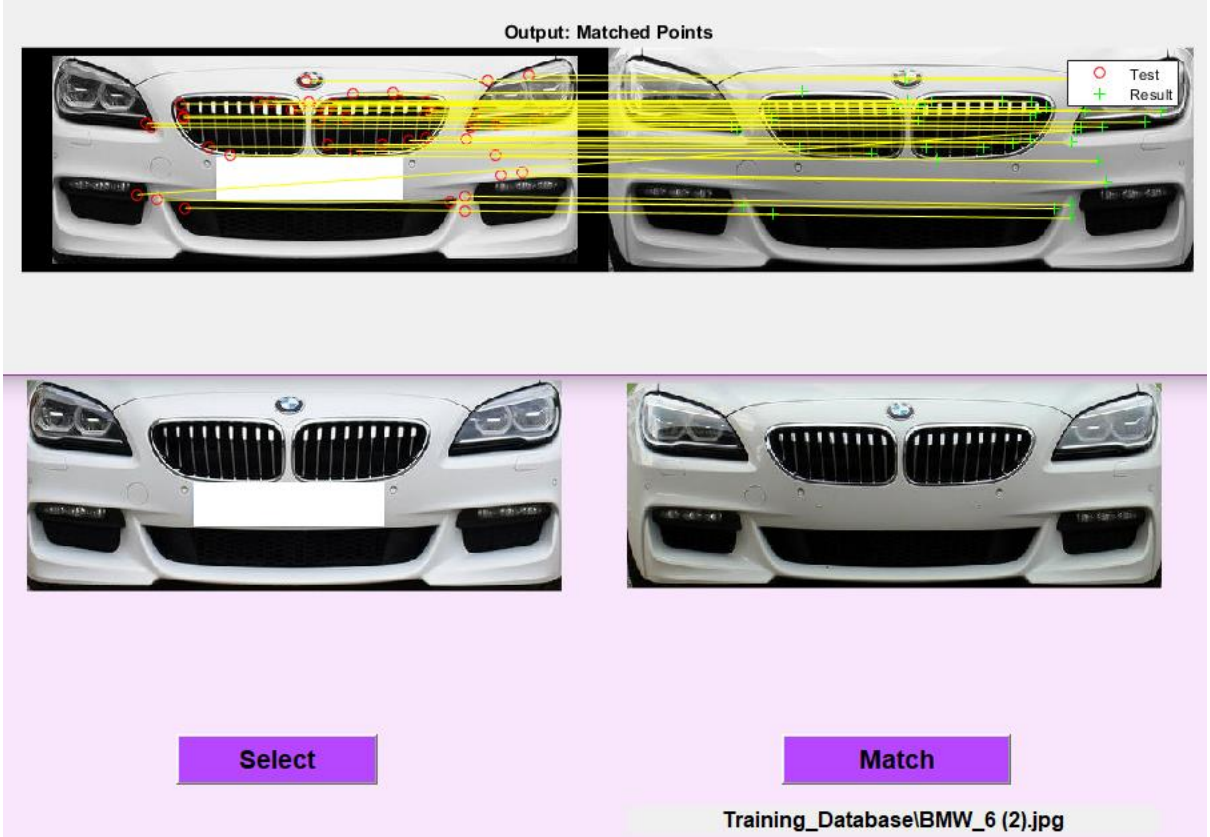


Fig 6.1 Output: BMW 6

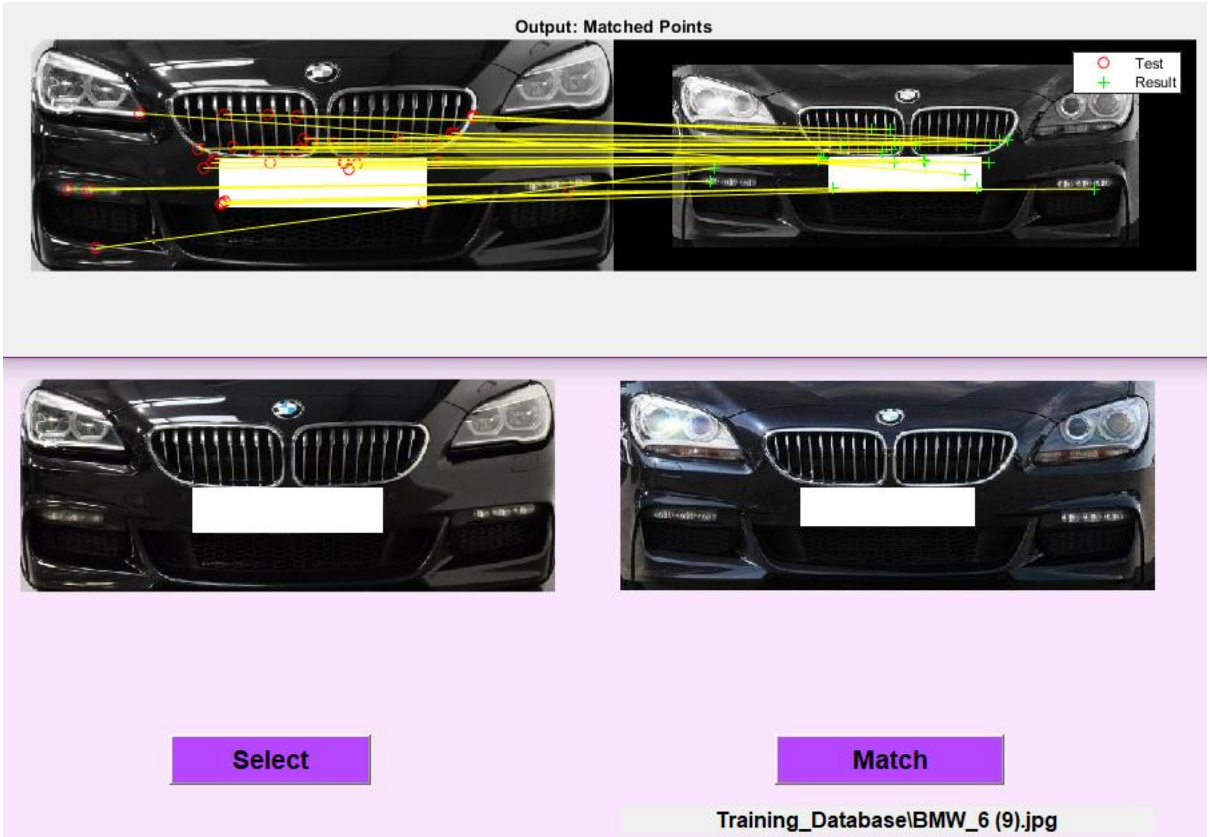


Fig 6.2 Output: BMW 6

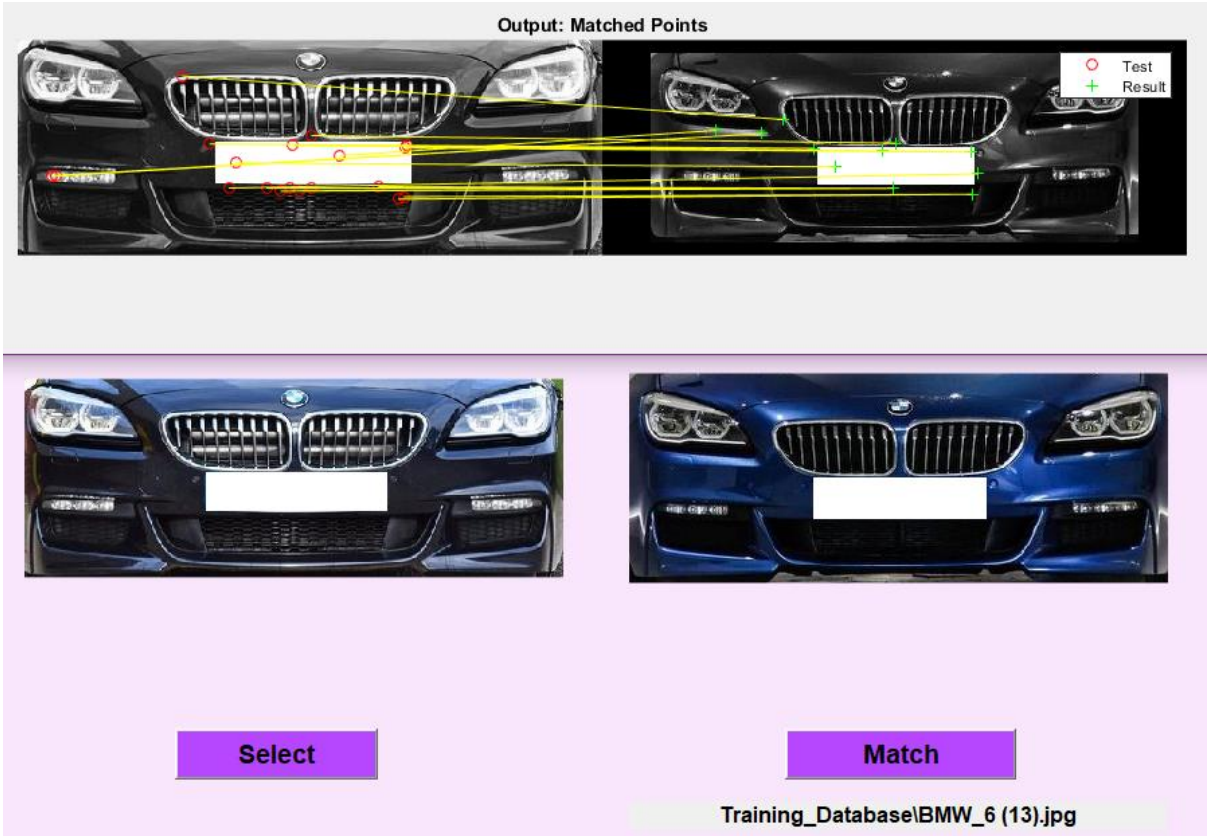


Fig 6.3 Output: BMW 6

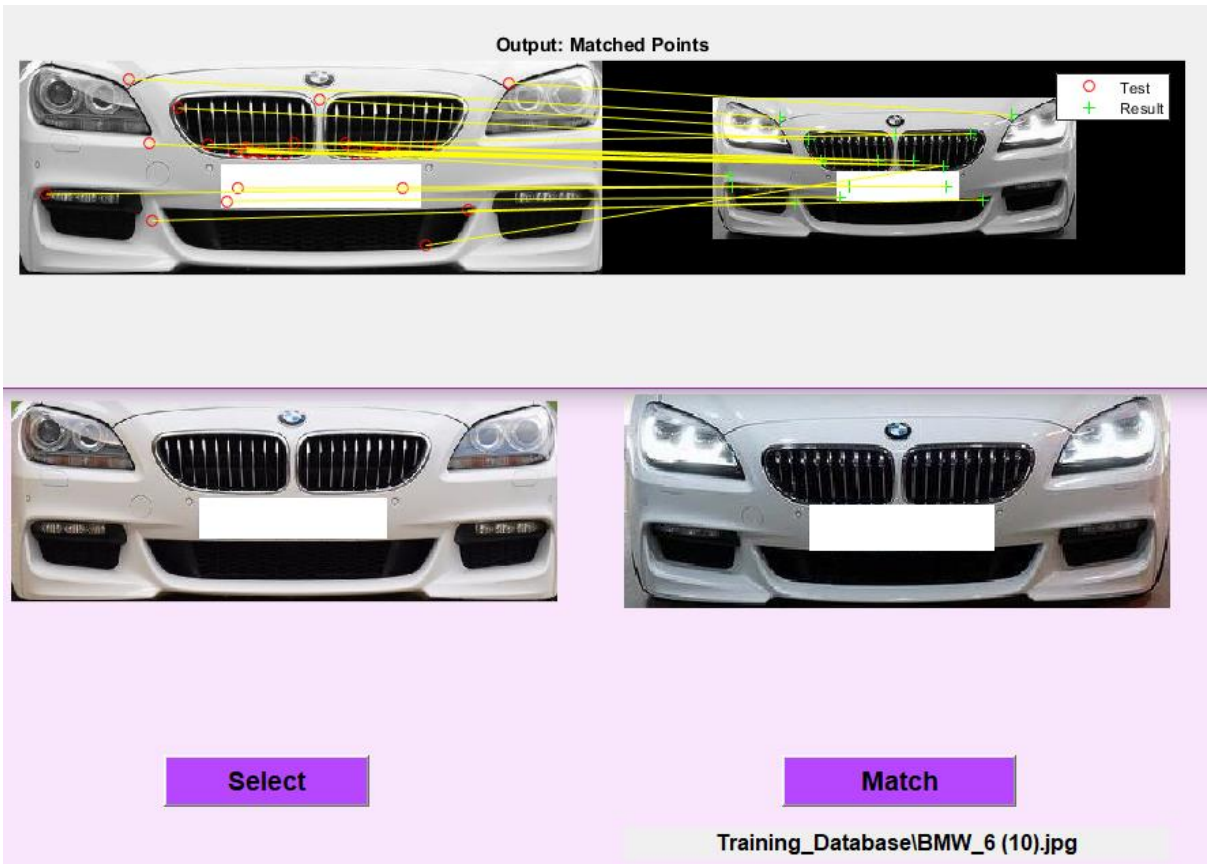


Fig 6.4 Output: BMW 6

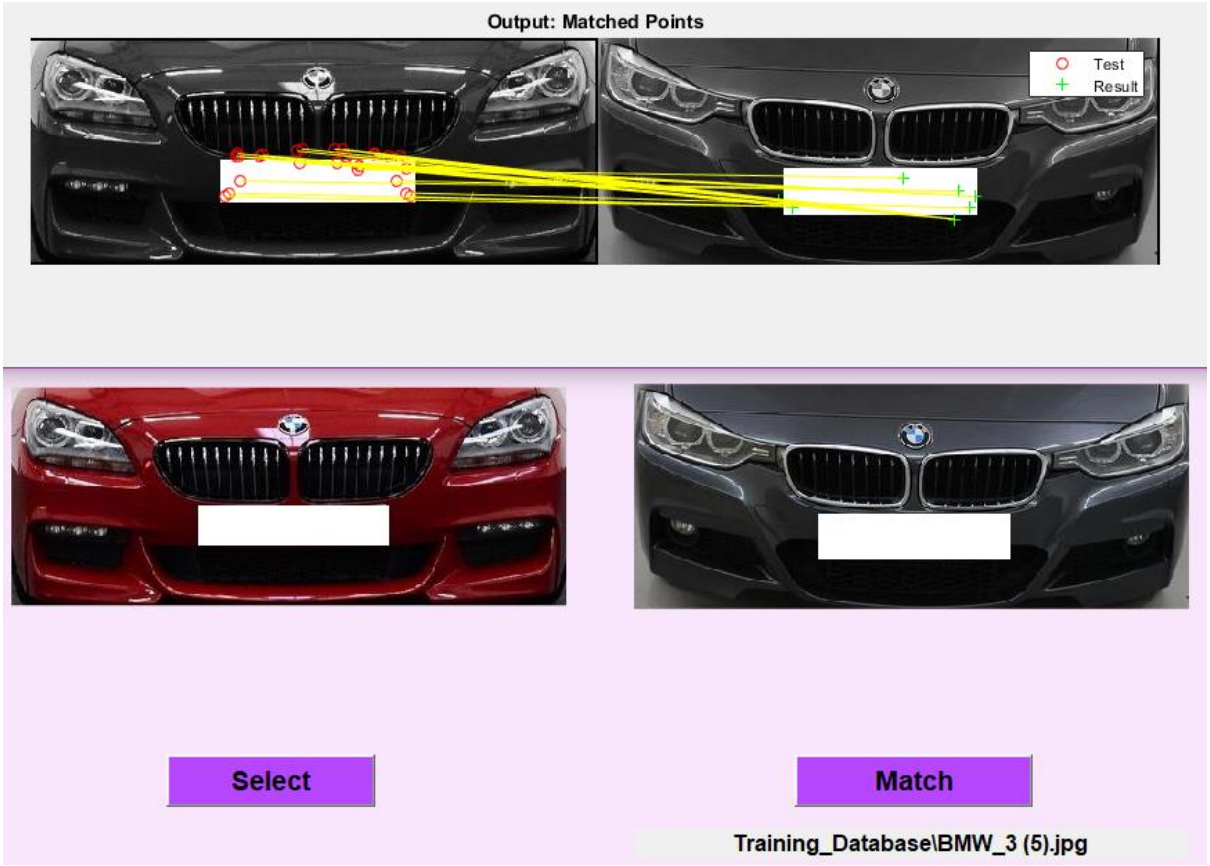


Fig 6.5 Output: **BMW 3** : Wrong result

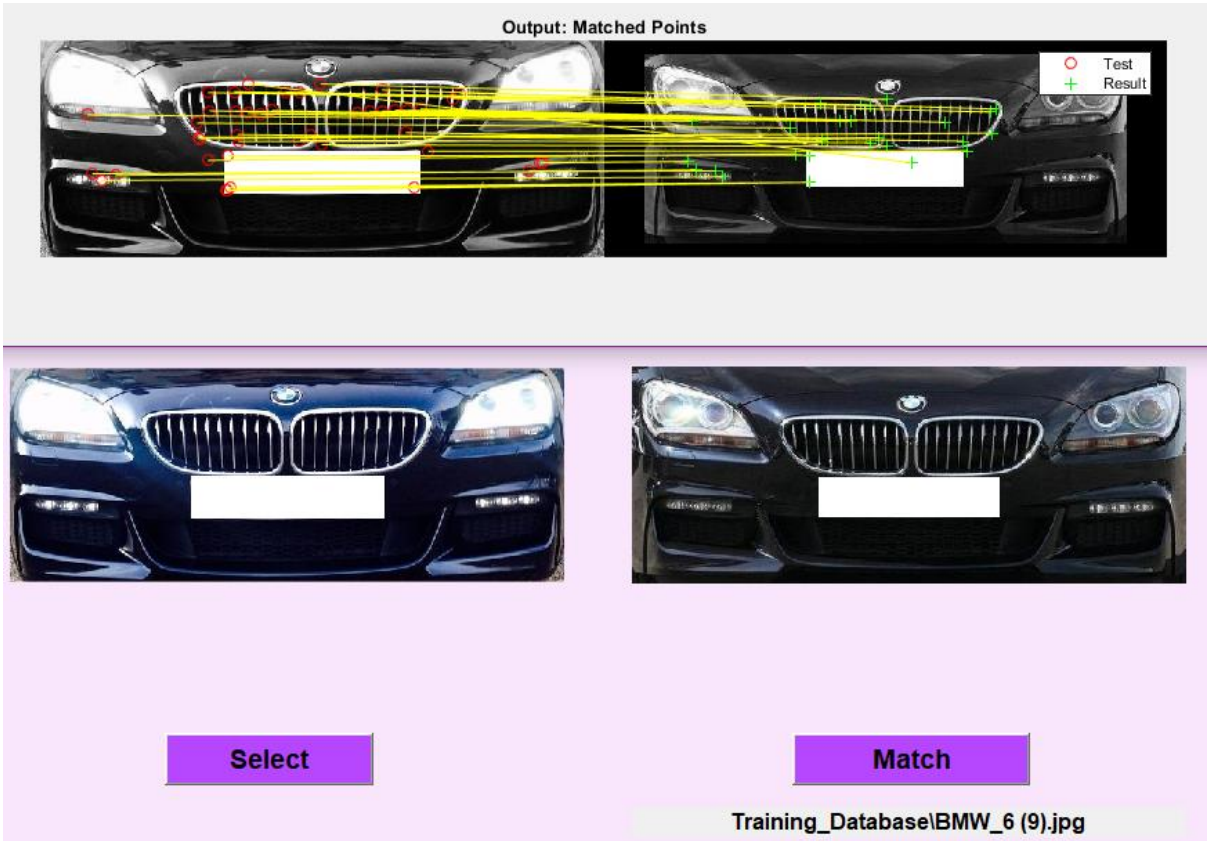


Fig 6.6 Output: BMW 6

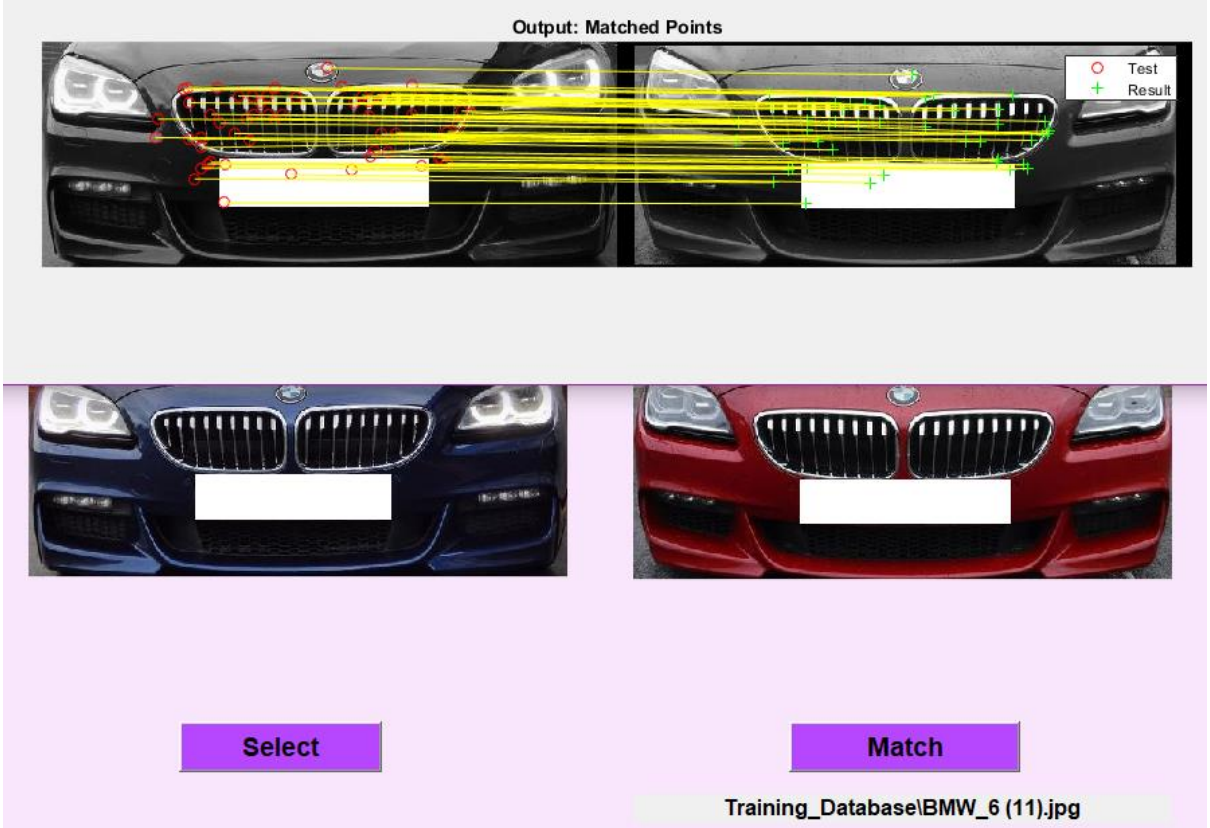


Fig 6.7 Output: BMW 6

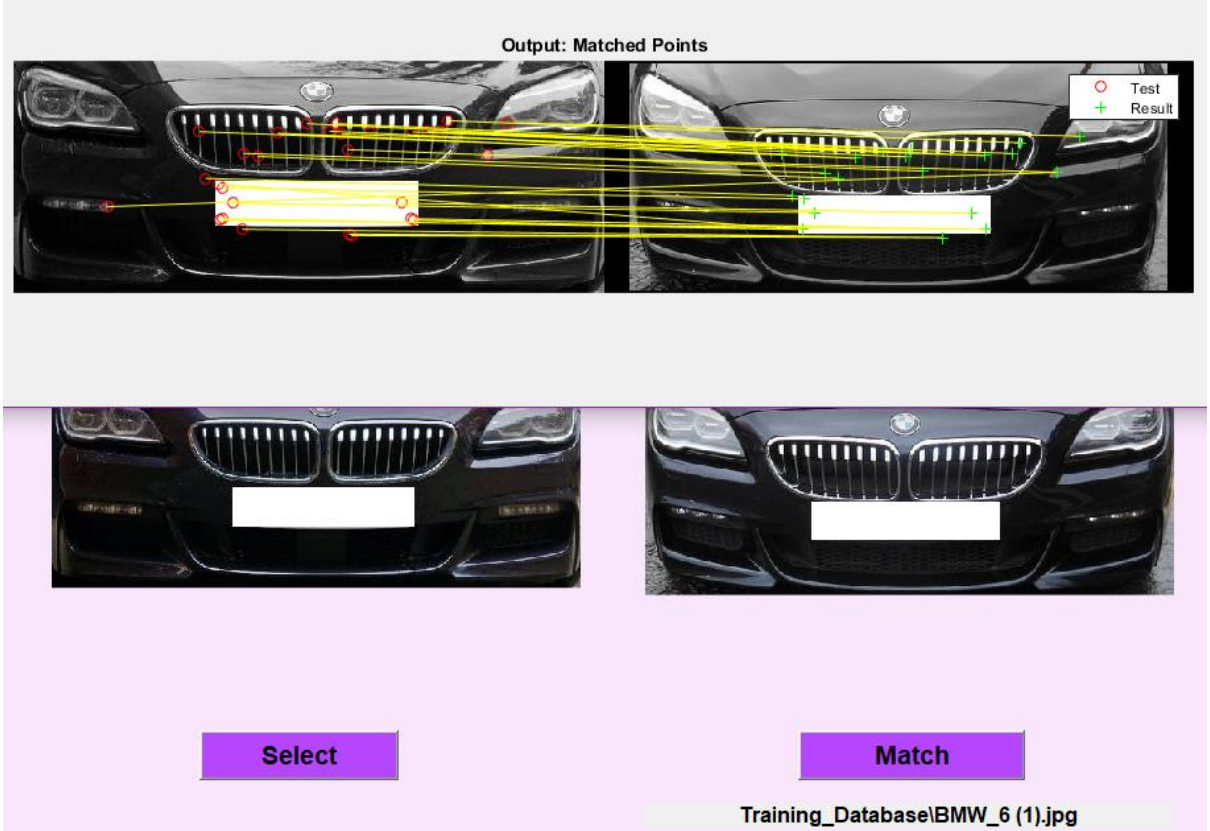


Fig 6.8 Output: BMW 6

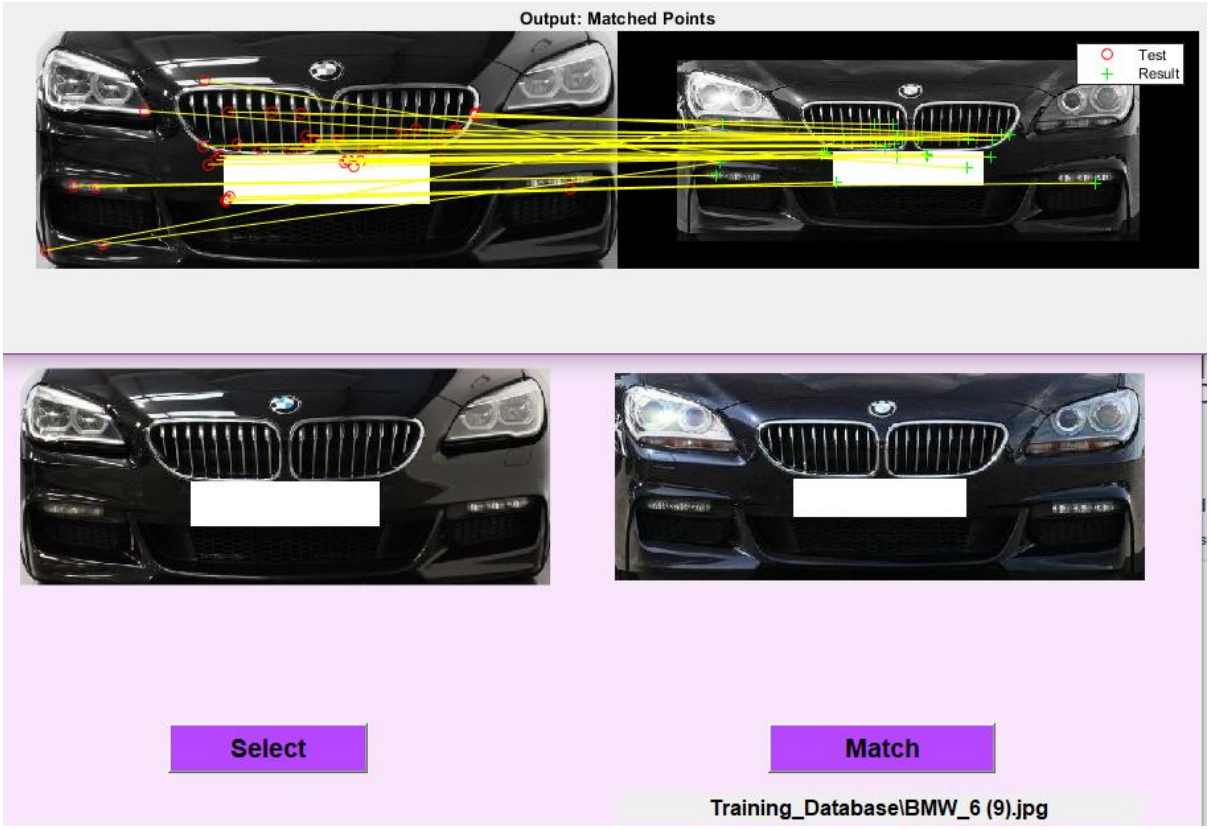


Fig 6.9 Output: BMW 6

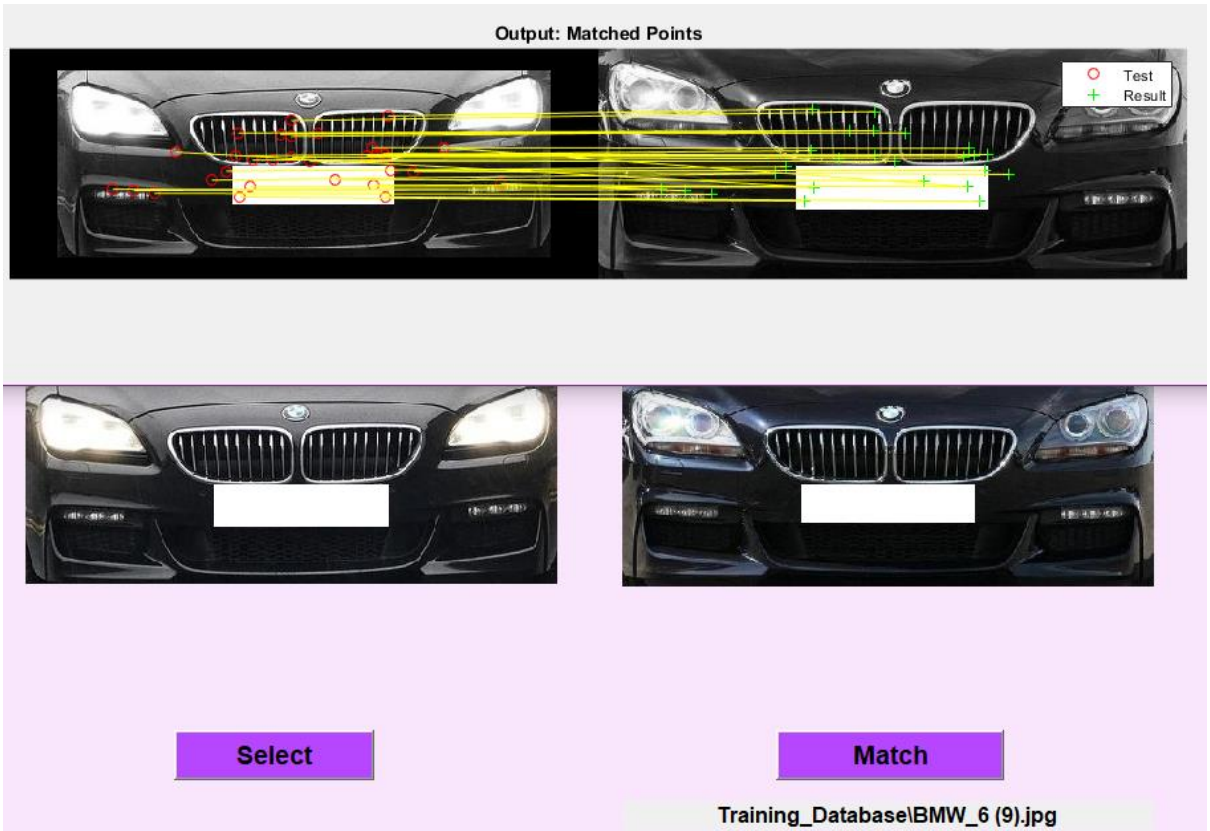


Fig 6.10 Output: BMW 6

7. Citroen C1

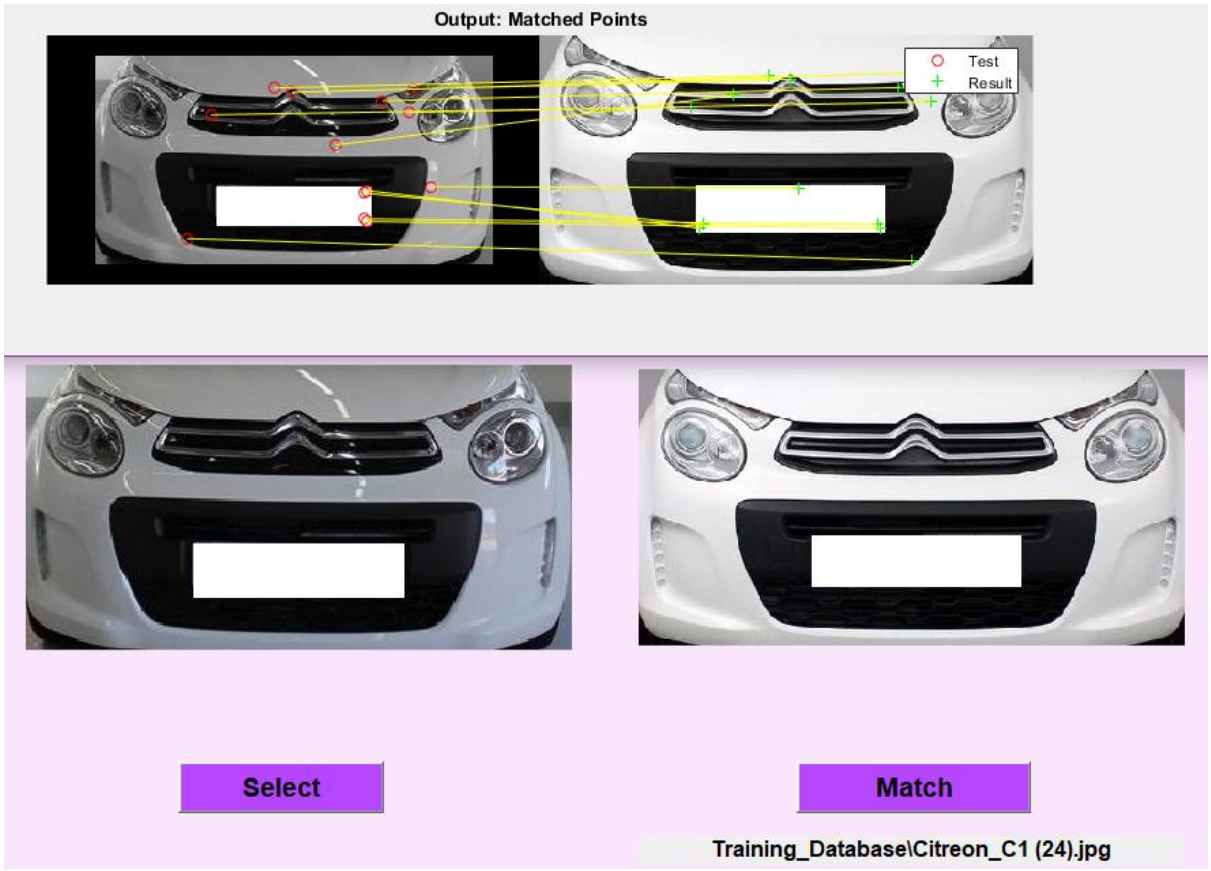


Fig 7.1 Output: Citroen C1

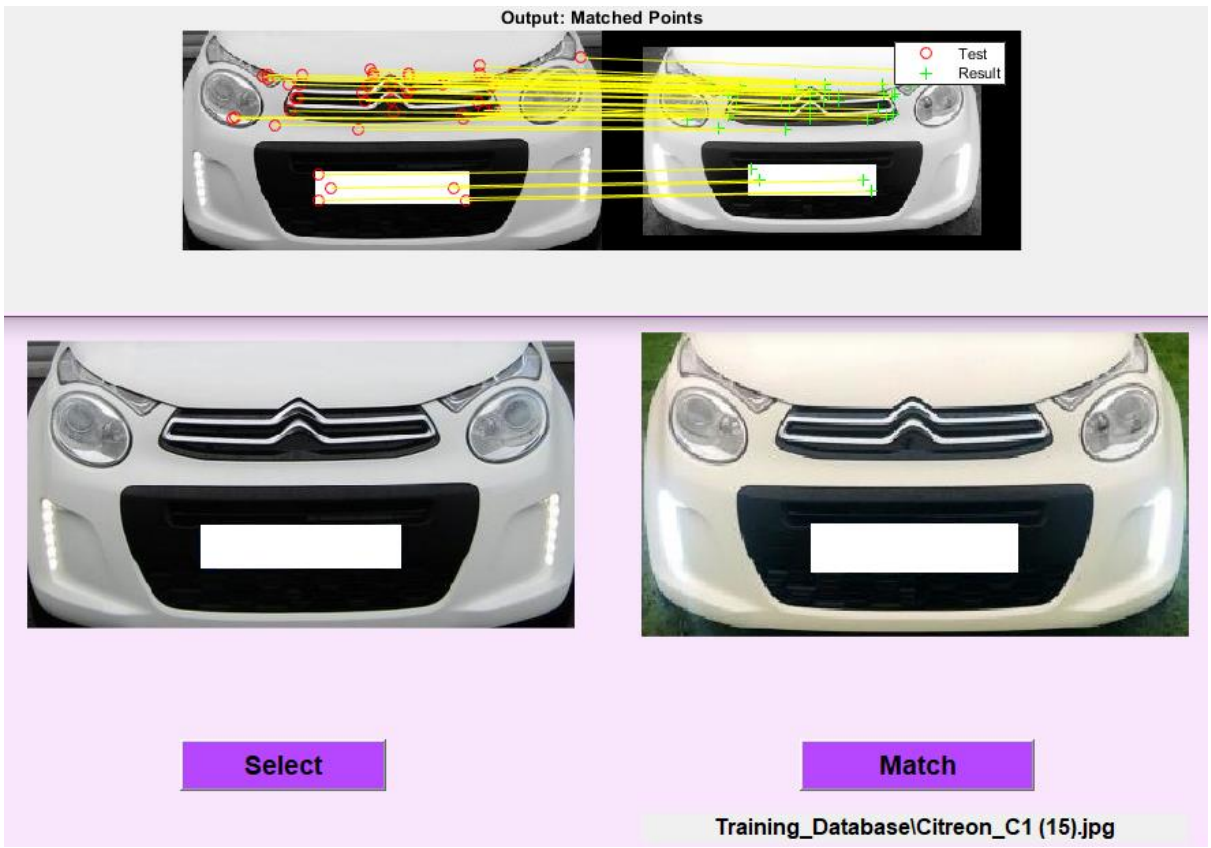


Fig 7.2 Output: Citroen C1

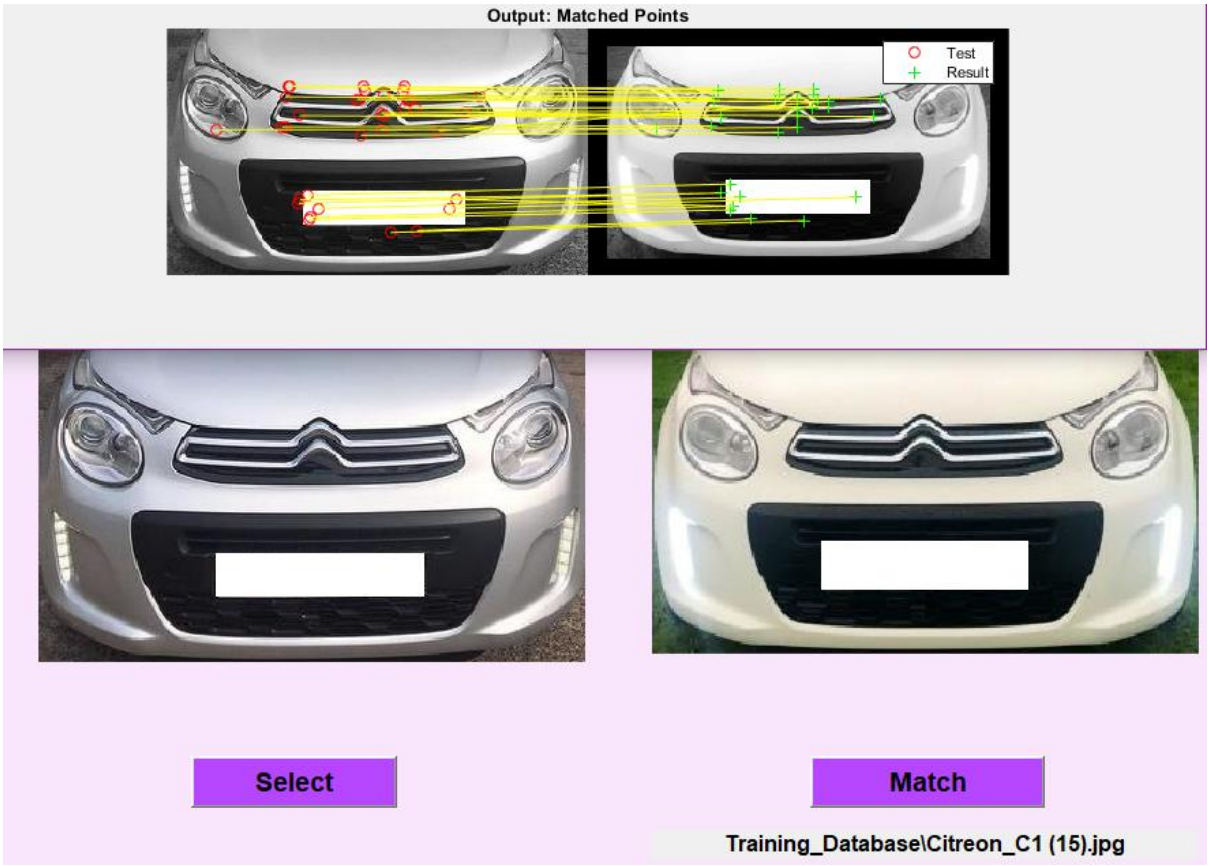


Fig 7.3 Output: Citroen C1

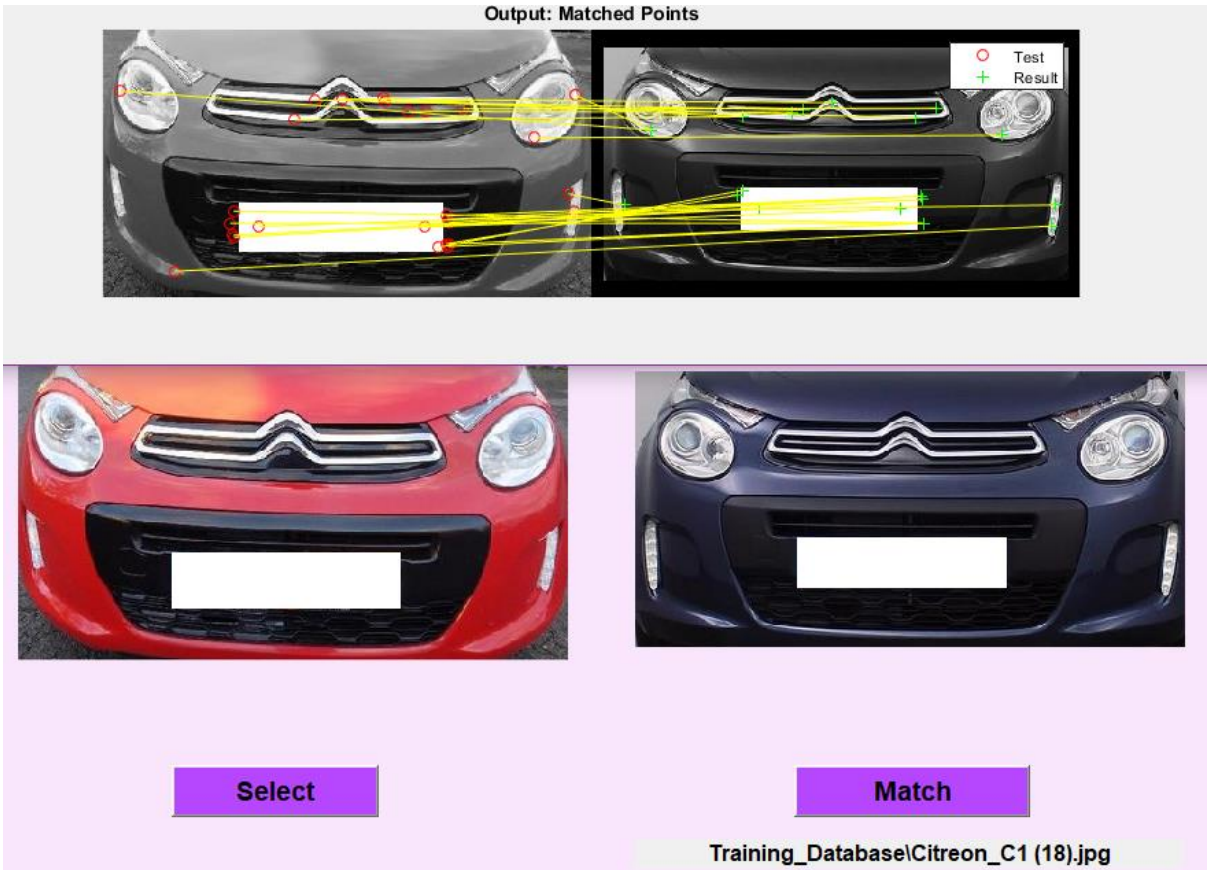


Fig 7.4 Output: Citroen C1

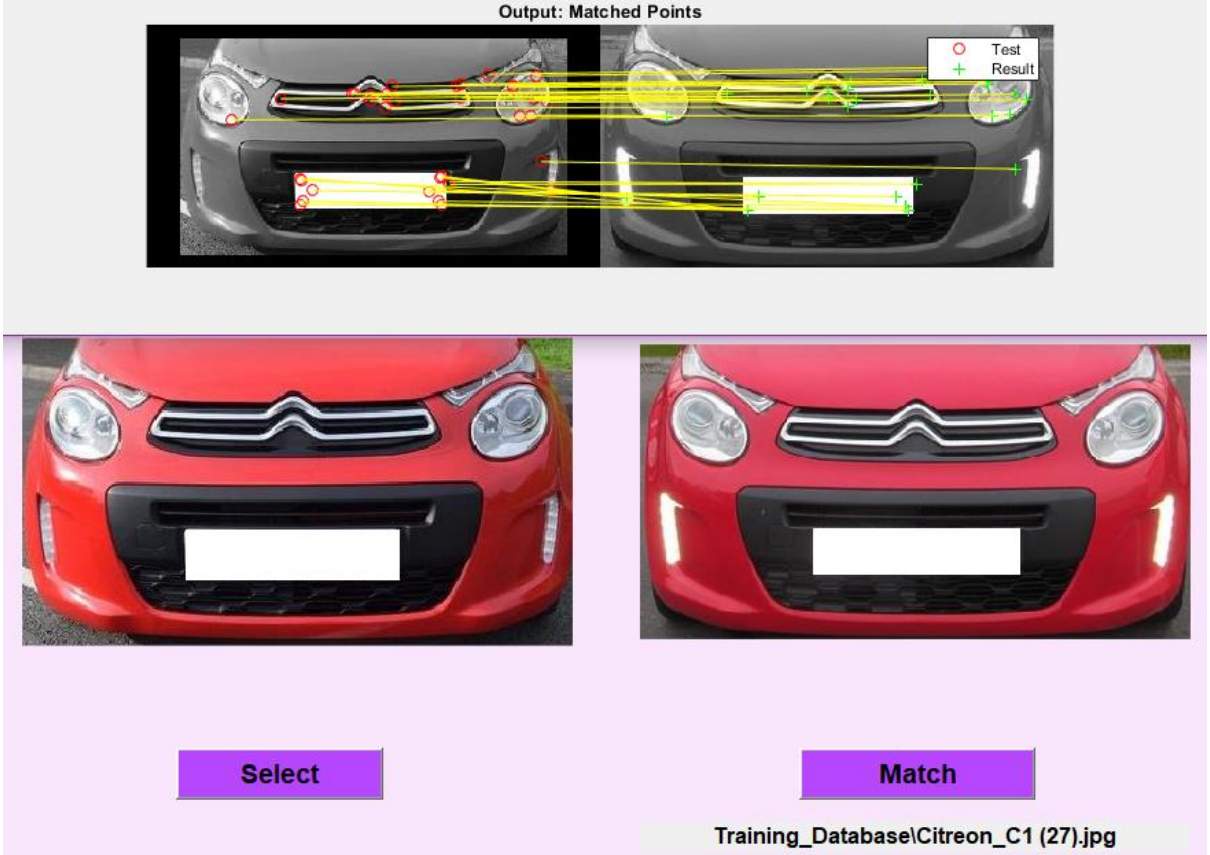


Fig 7.5 Output: Citroen C1

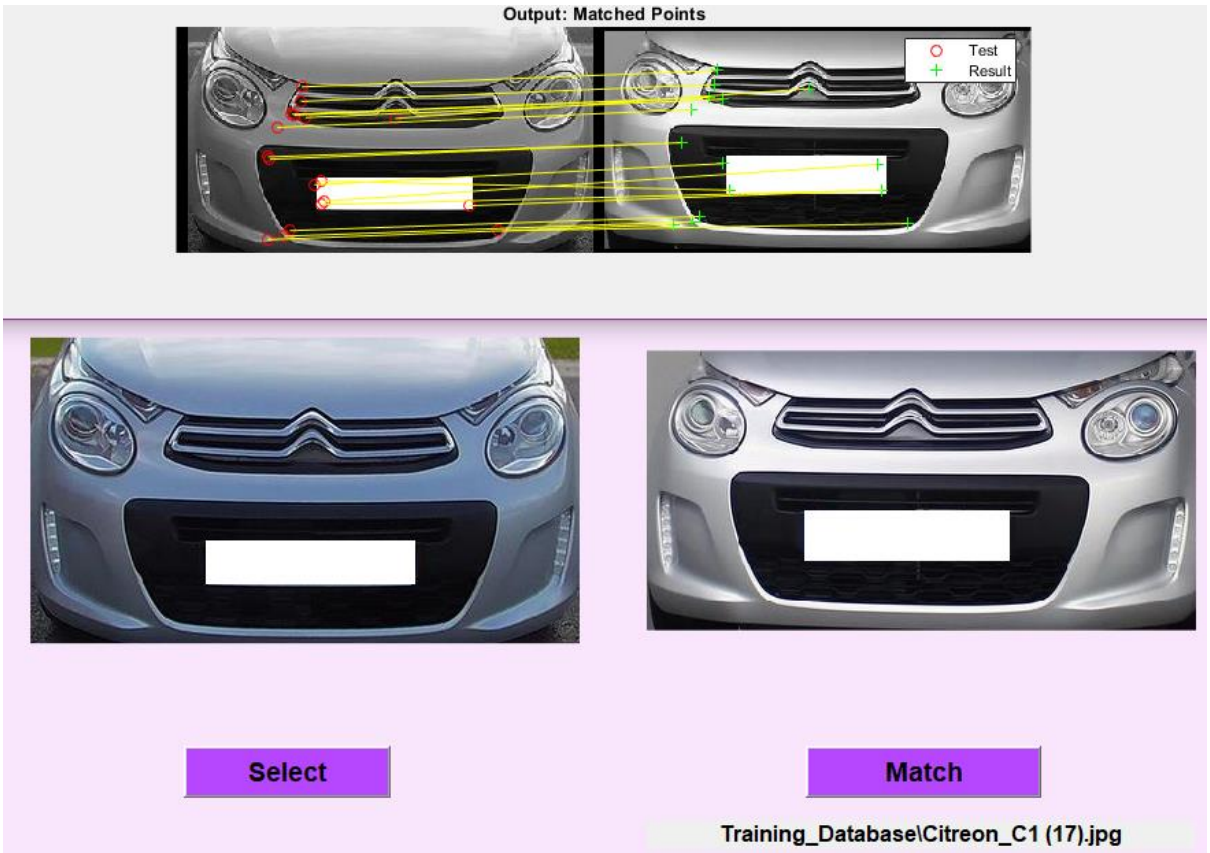


Fig 7.6 Output: Citroen C1

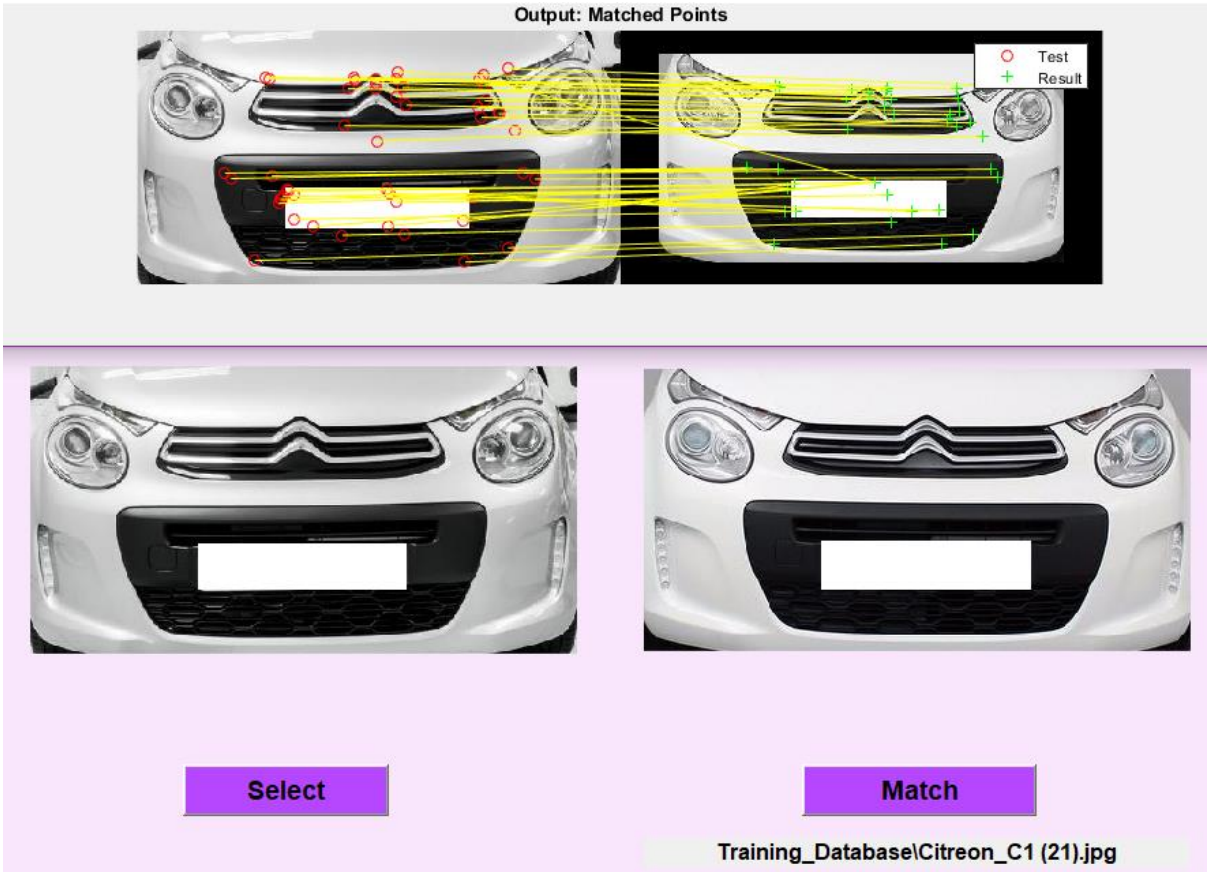


Fig 7.7 Output: Citroen C1

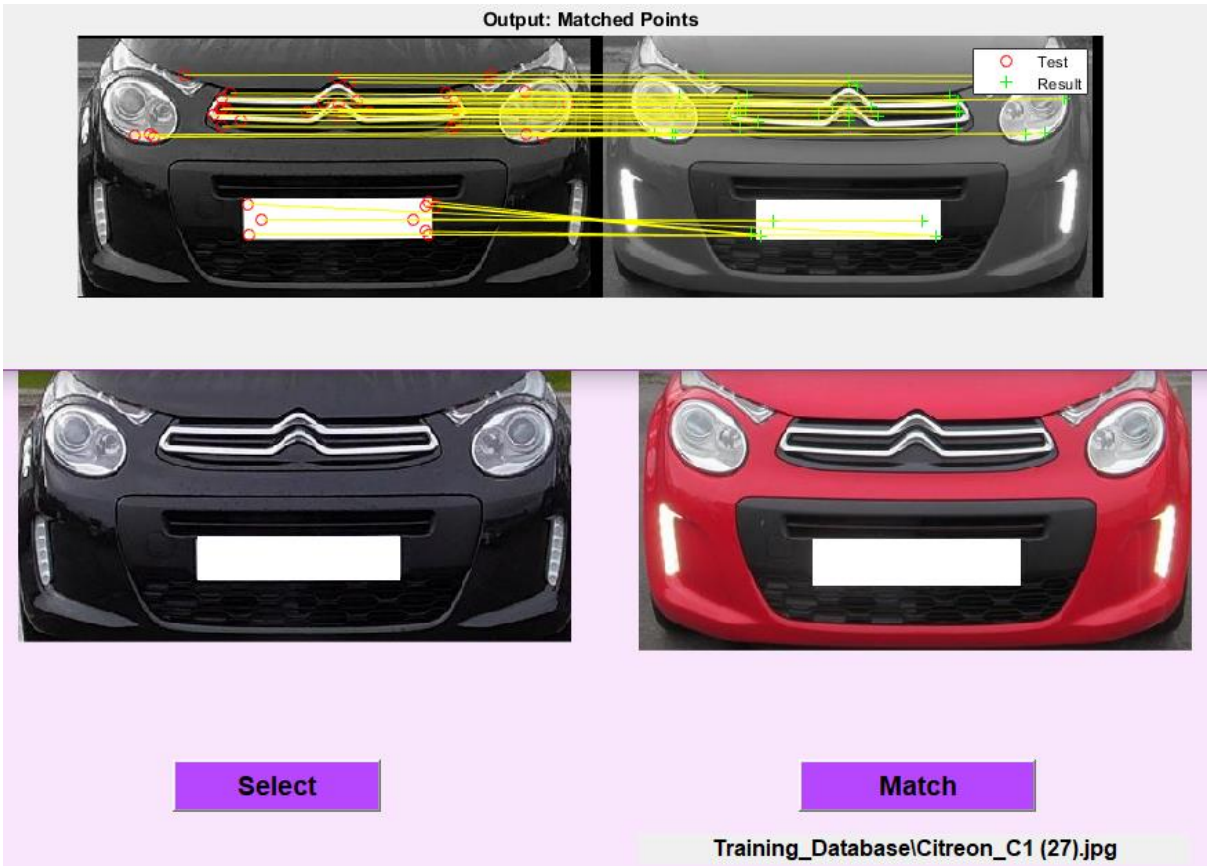


Fig 7.8 Output: Citroen C1

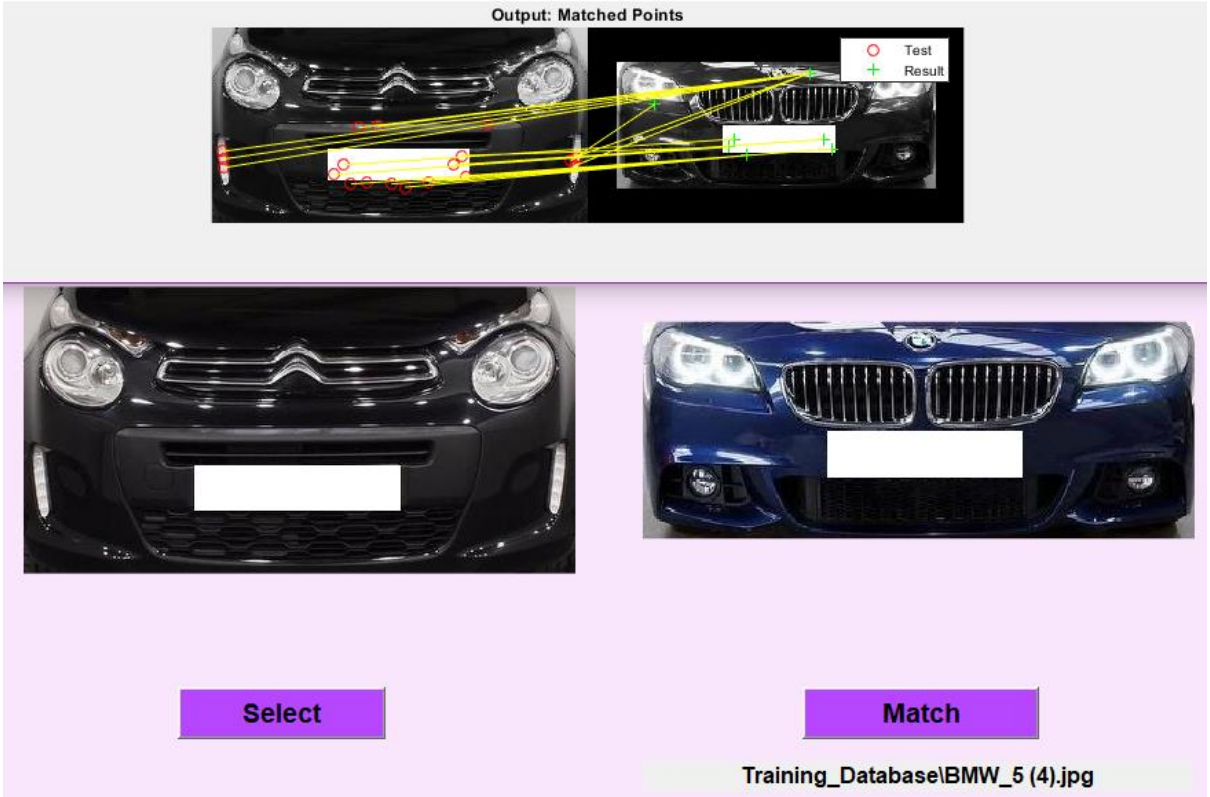


Fig 7.9 Output: **BWM 5** : Wrong Result

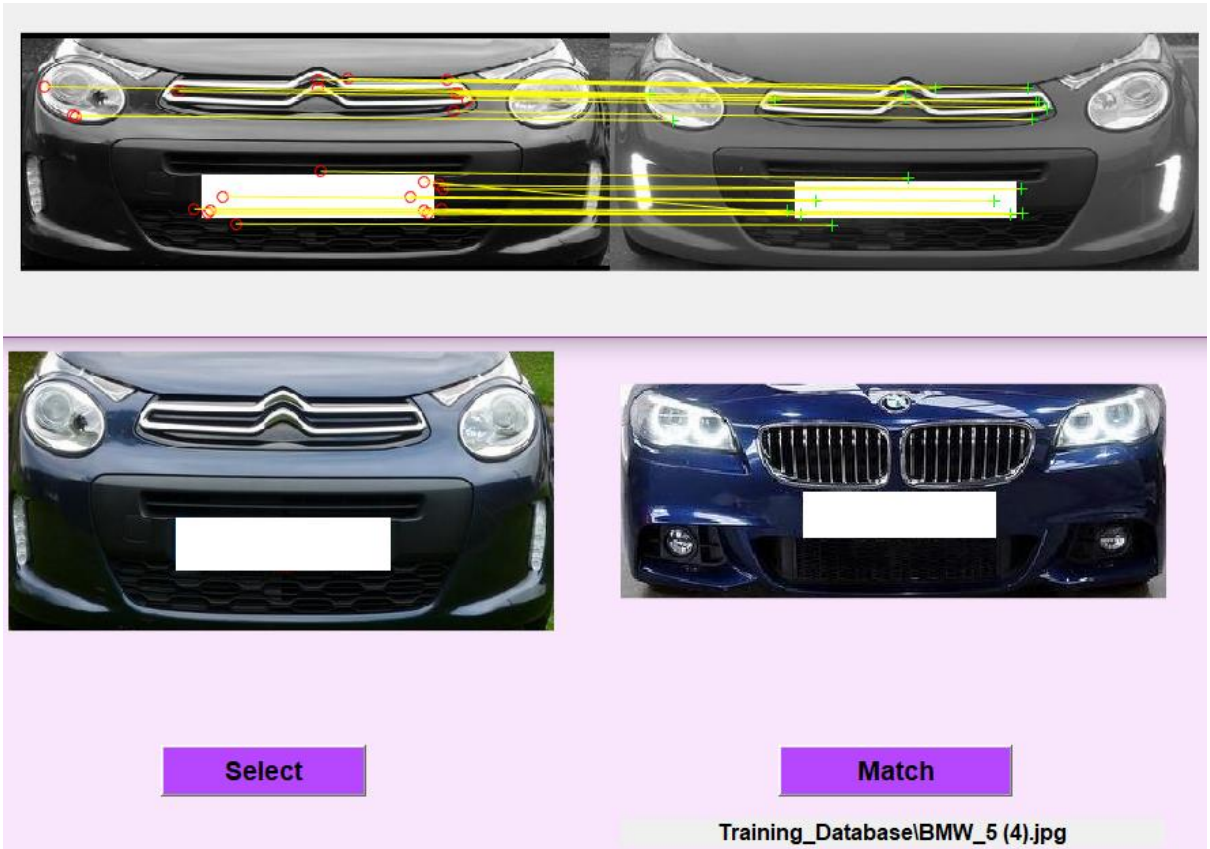


Fig 7.10 Output: **BMW 5**: Wrong Result

8. Citroen C3

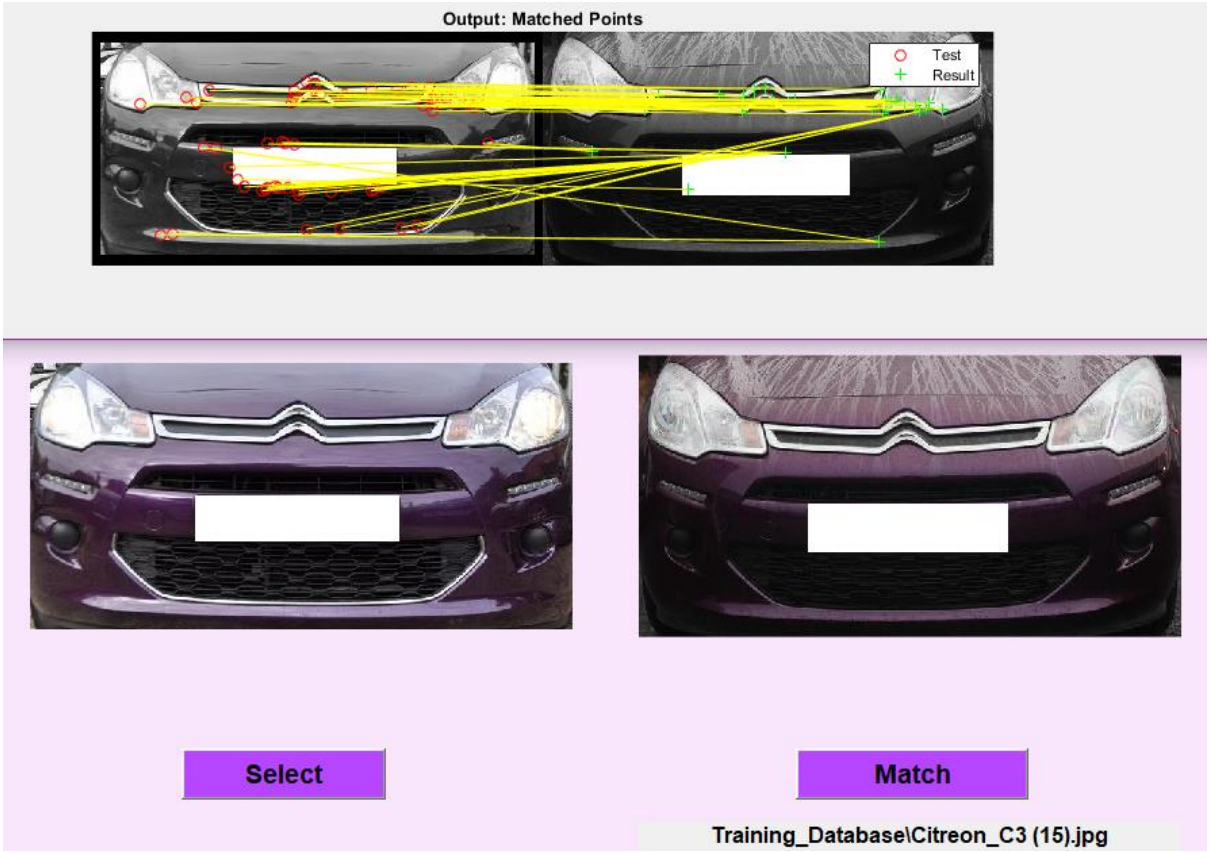


Fig 8.1 Output: Citroen C3

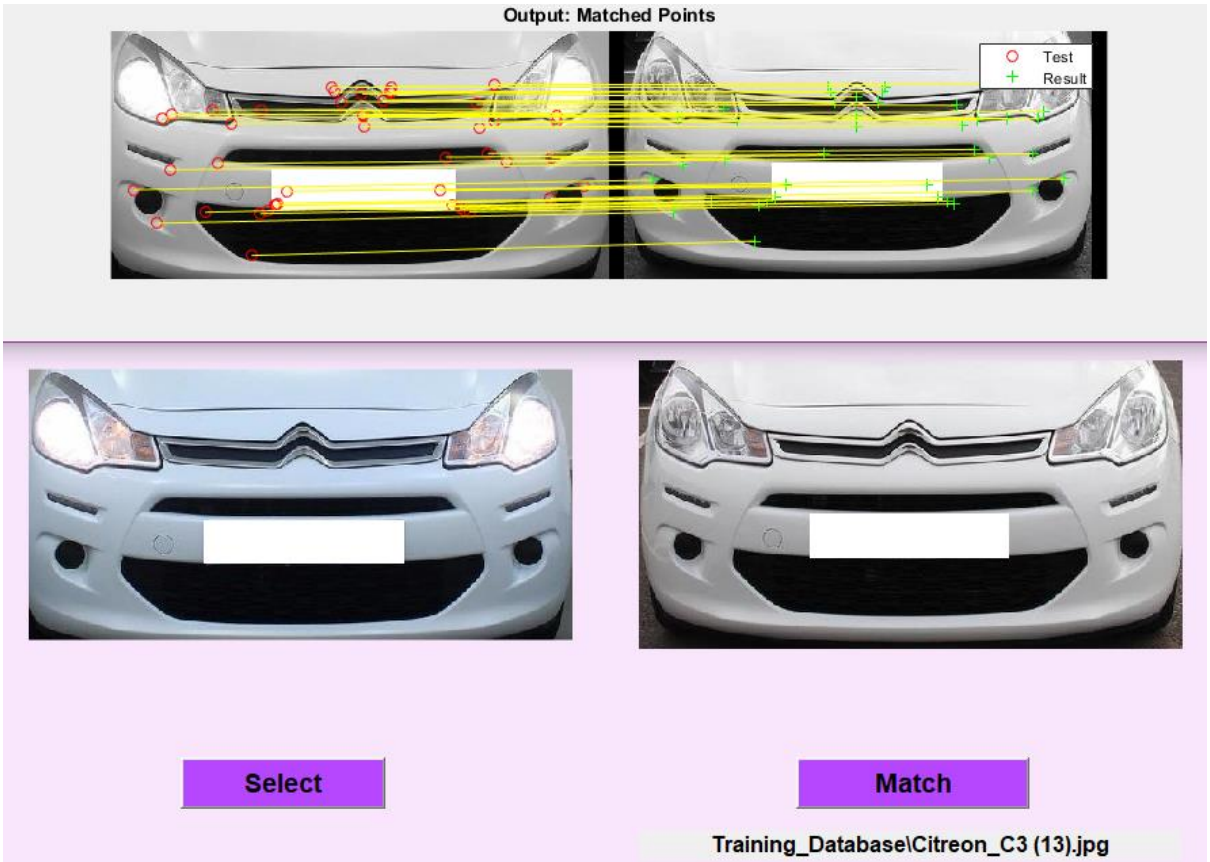


Fig 8.2 Output: Citroen C3

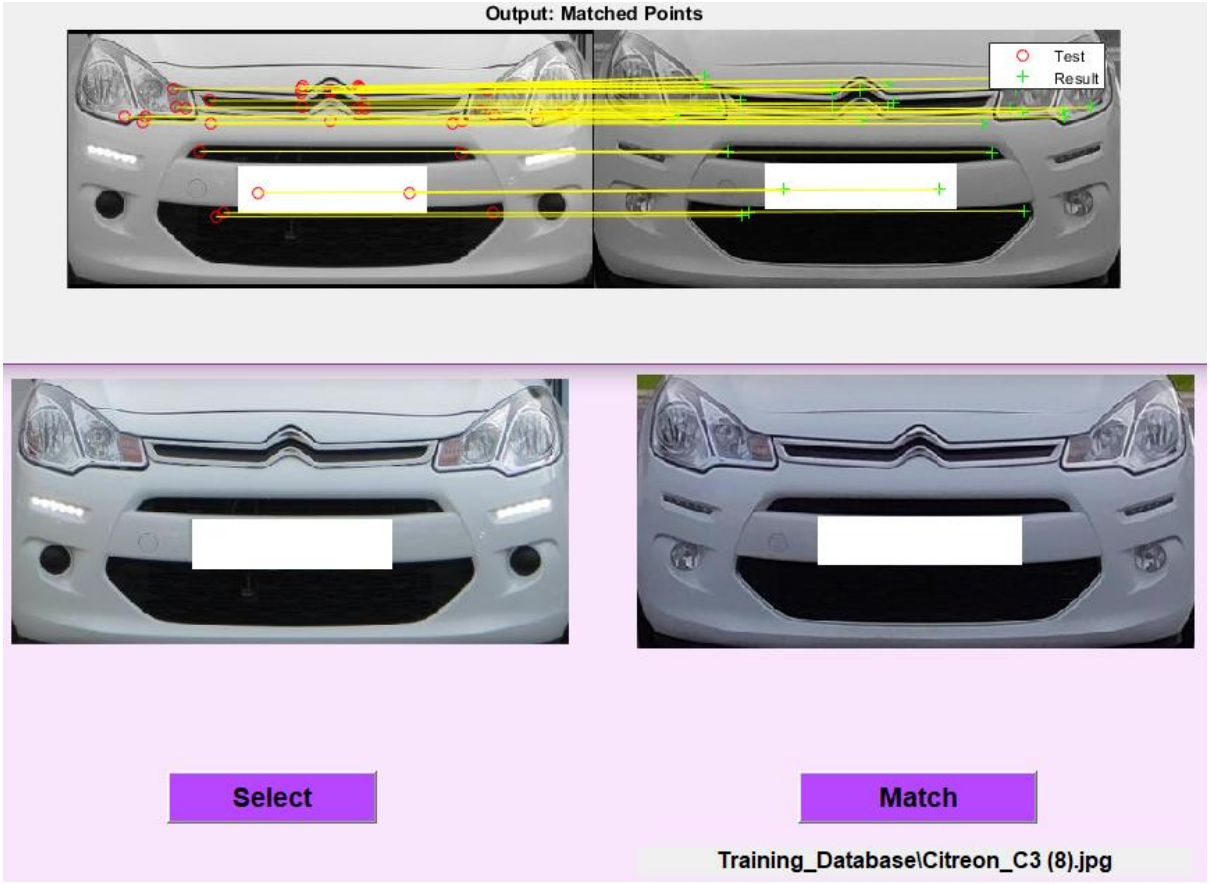


Fig 8.3 Output: Citroen C3

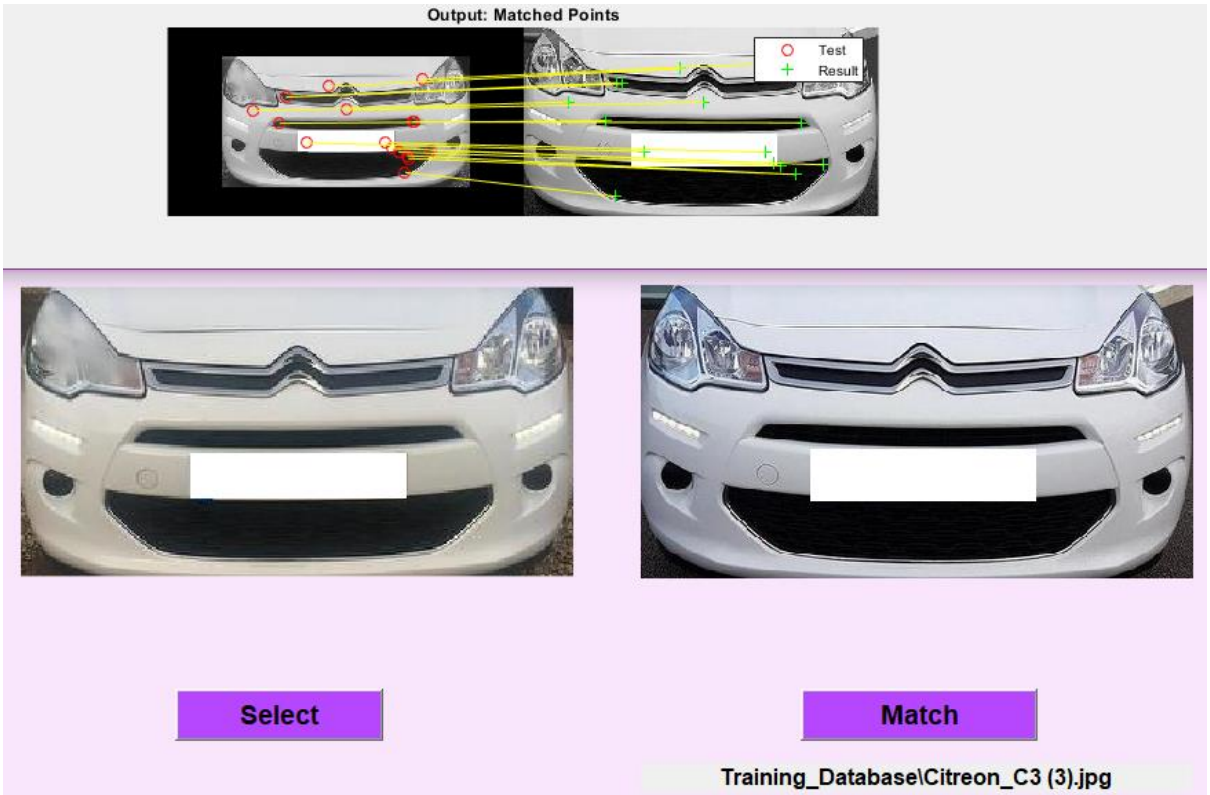


Fig 8.4 Output: Citroen C3

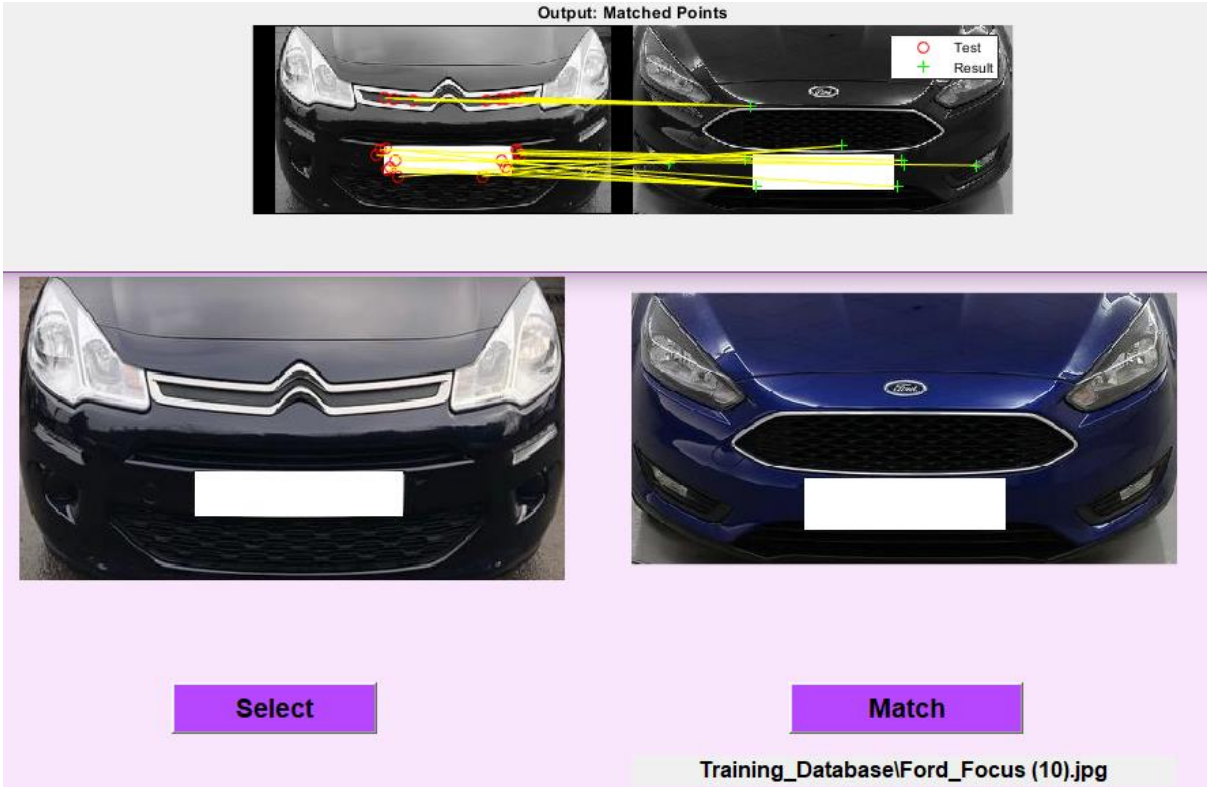


Fig 8.5 Output: **Ford Focus**: Wrong Result

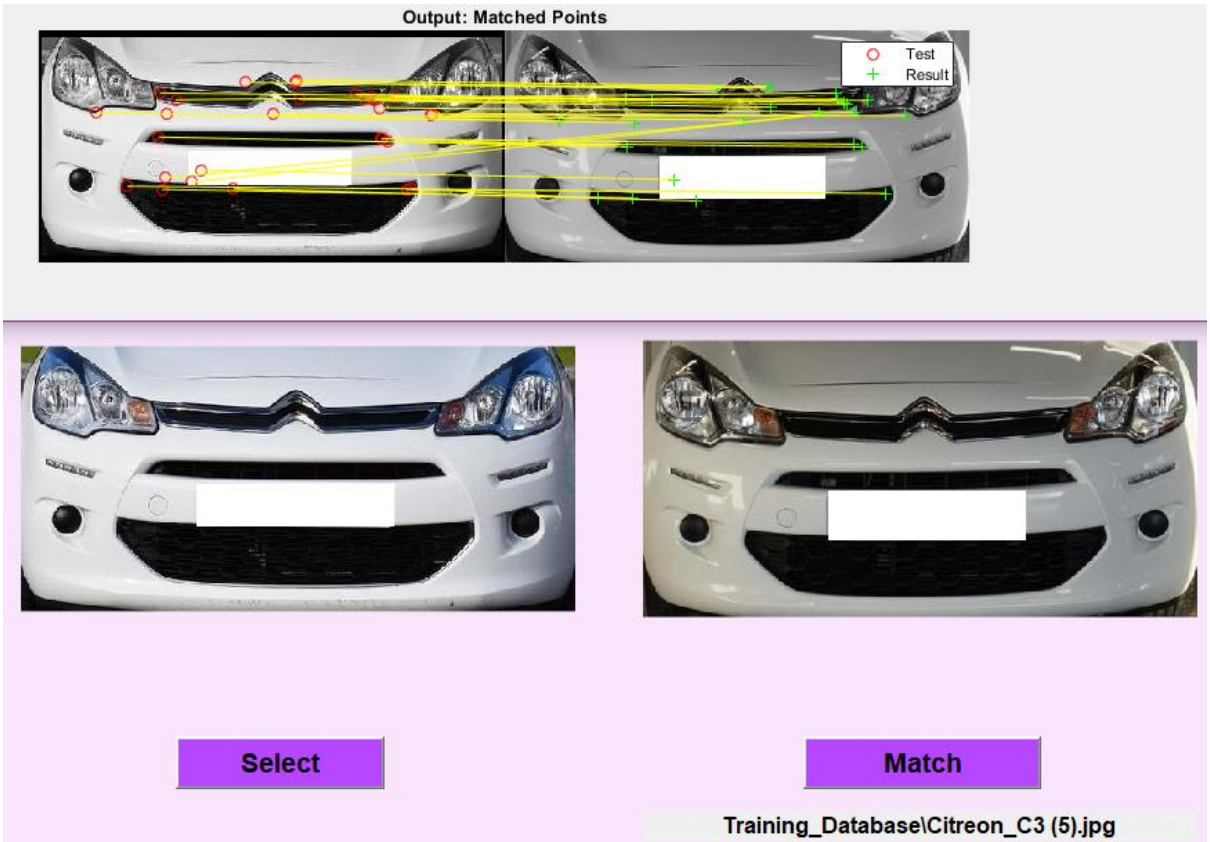


Fig 8.6 Output: Citroen C3

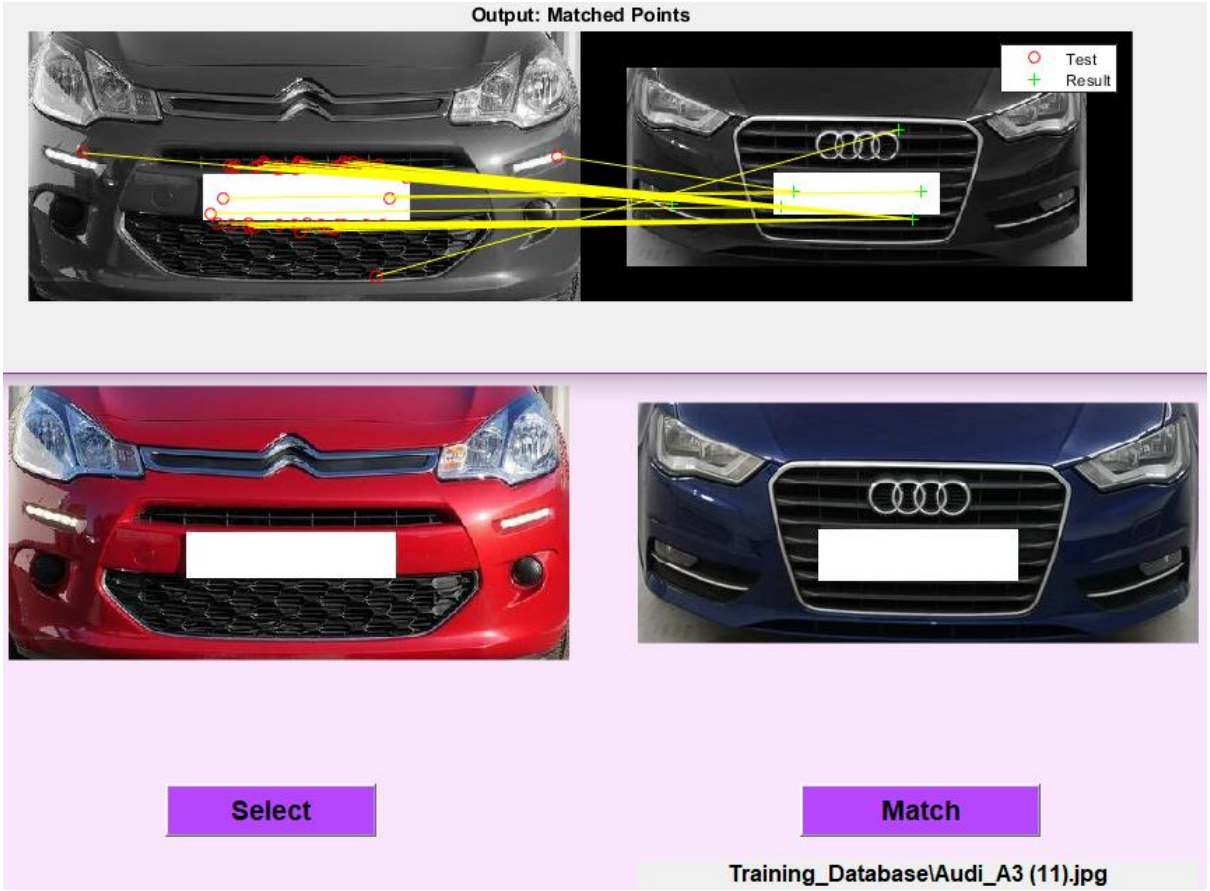


Fig 8.7 Output: **Audi A3**: Wrong Result

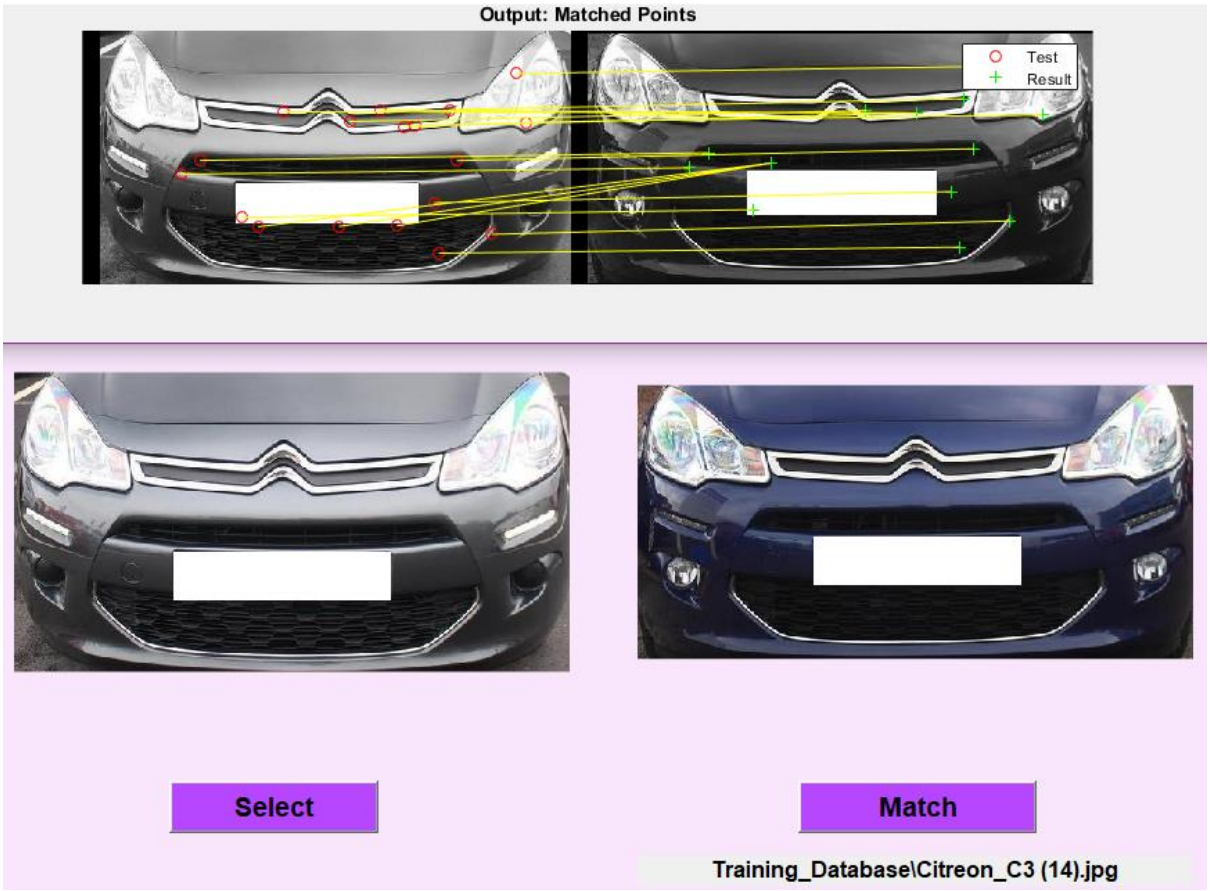


Fig 8.8 Output: Citroen C3

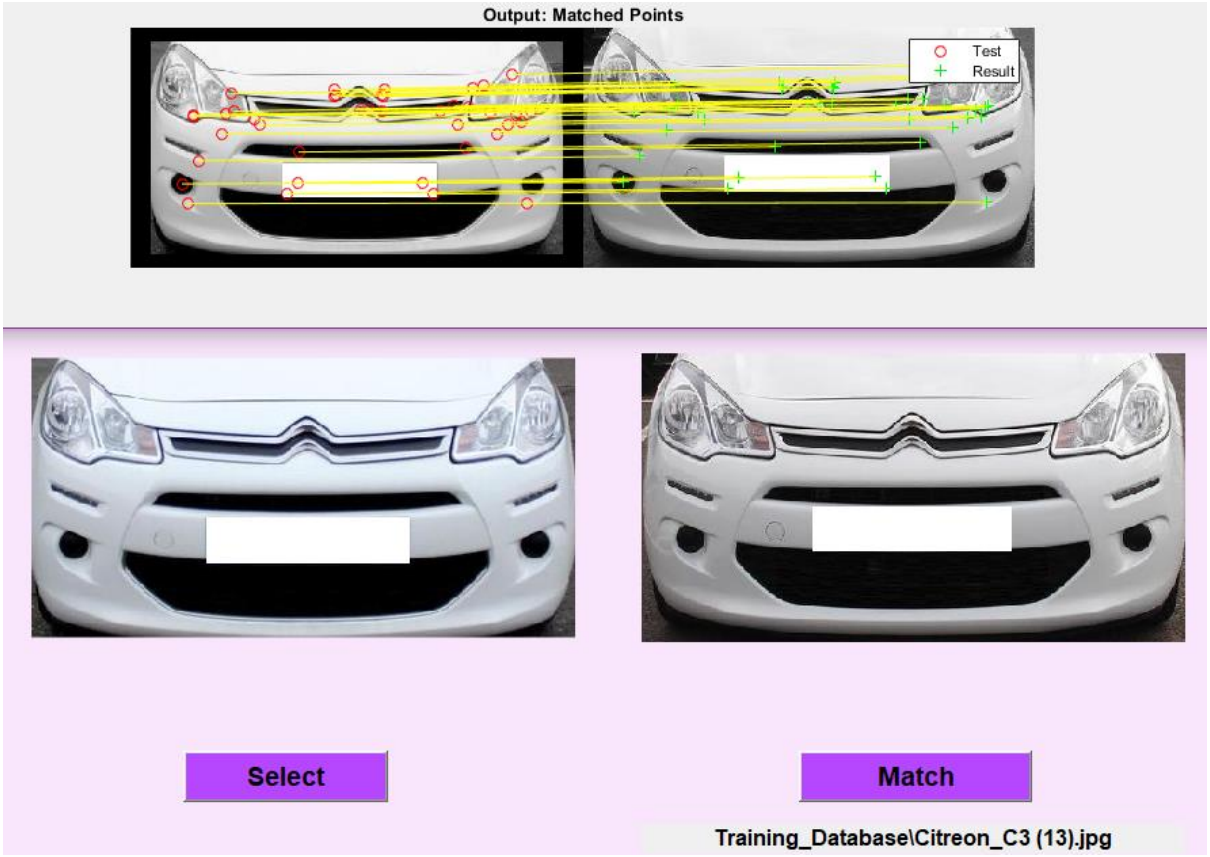


Fig 8.9 Output: Citroen C3

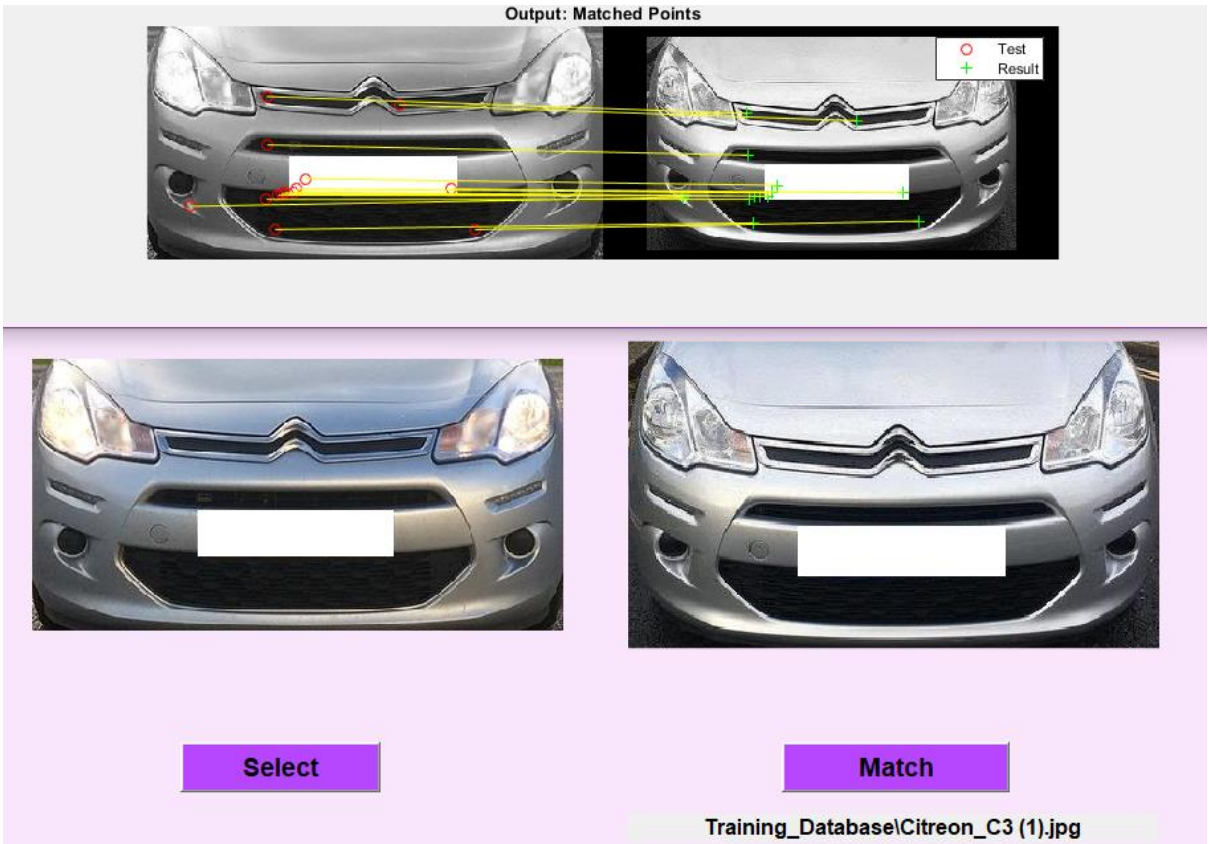


Fig 8.10 Output: Citroen C3

9. Ford Fiesta

Fig 9.1 Output: Ford Fiesta

10. Ford Focus

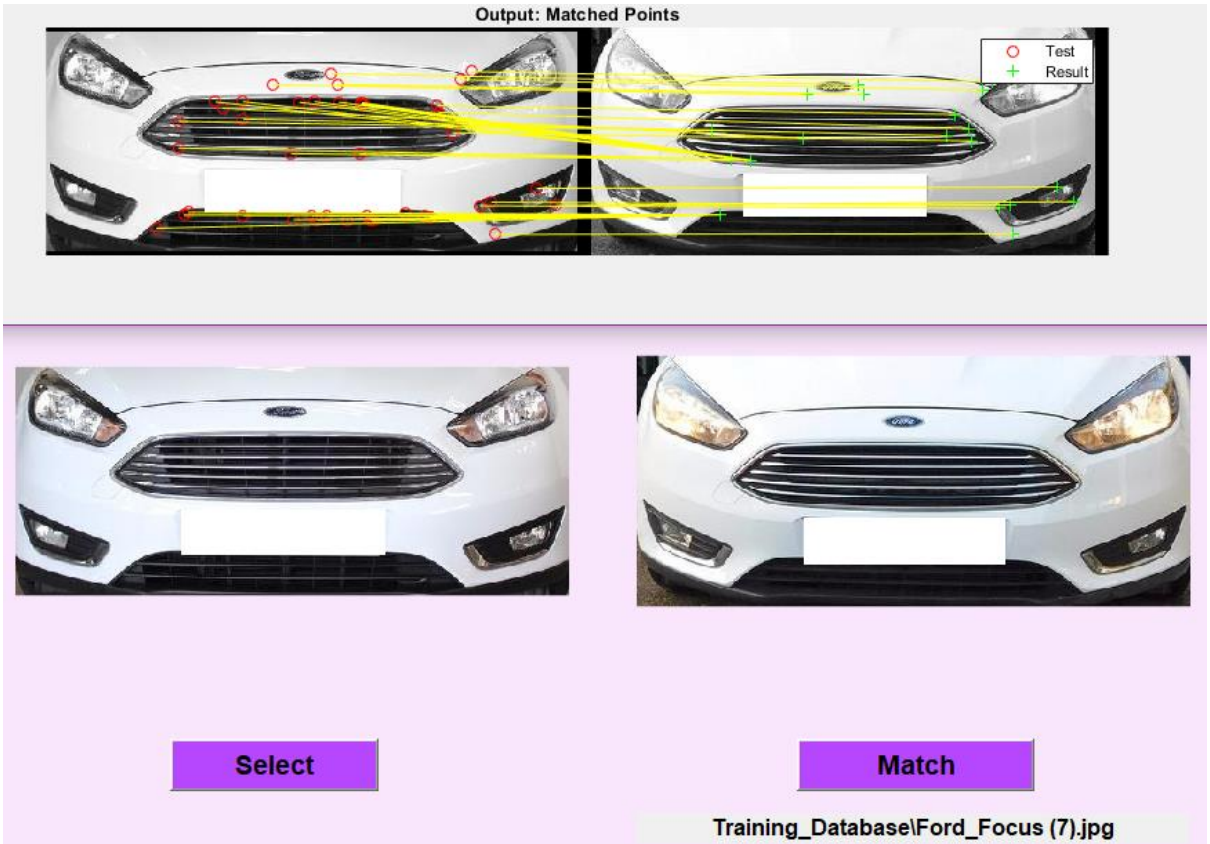


Fig 10.1 Output: Ford Focus

11. Hyundai i10

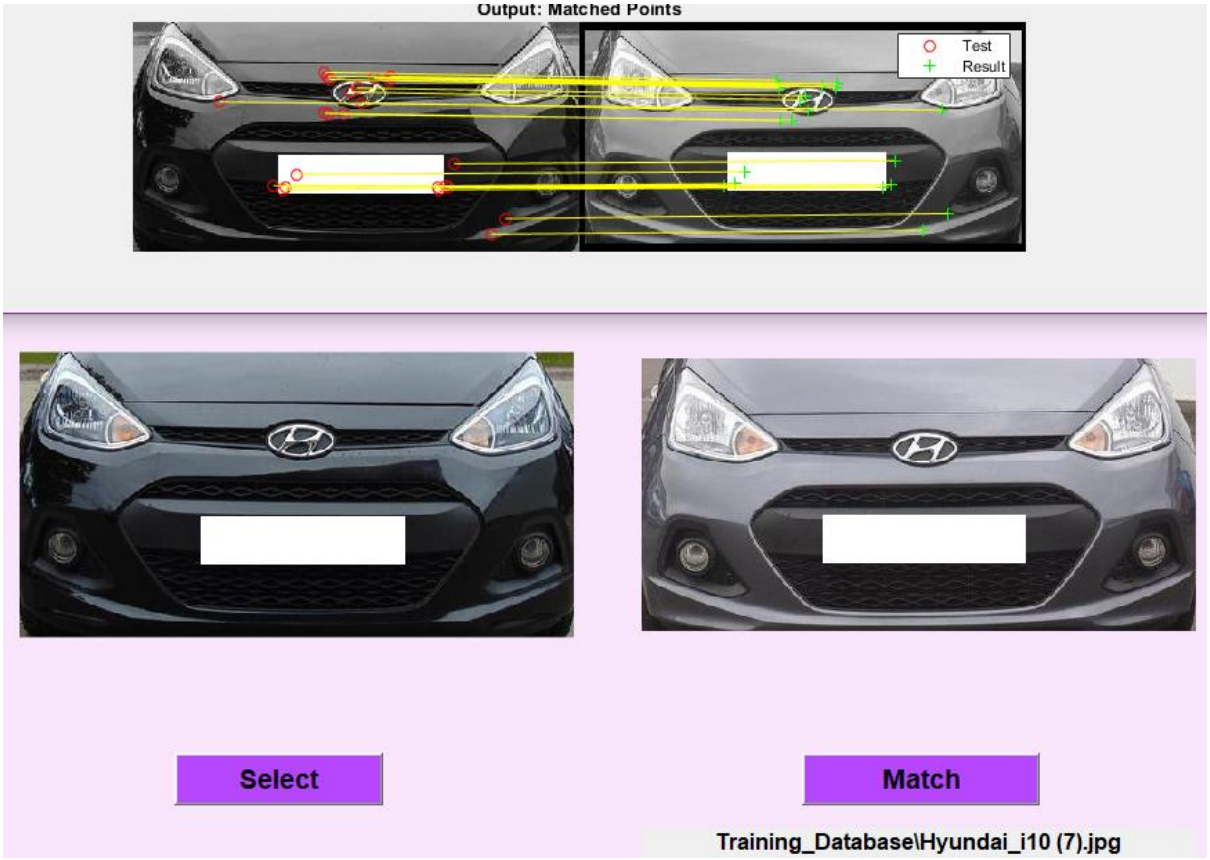


Fig 11.1 Output: Hyundai i10

12. Hyundai i30

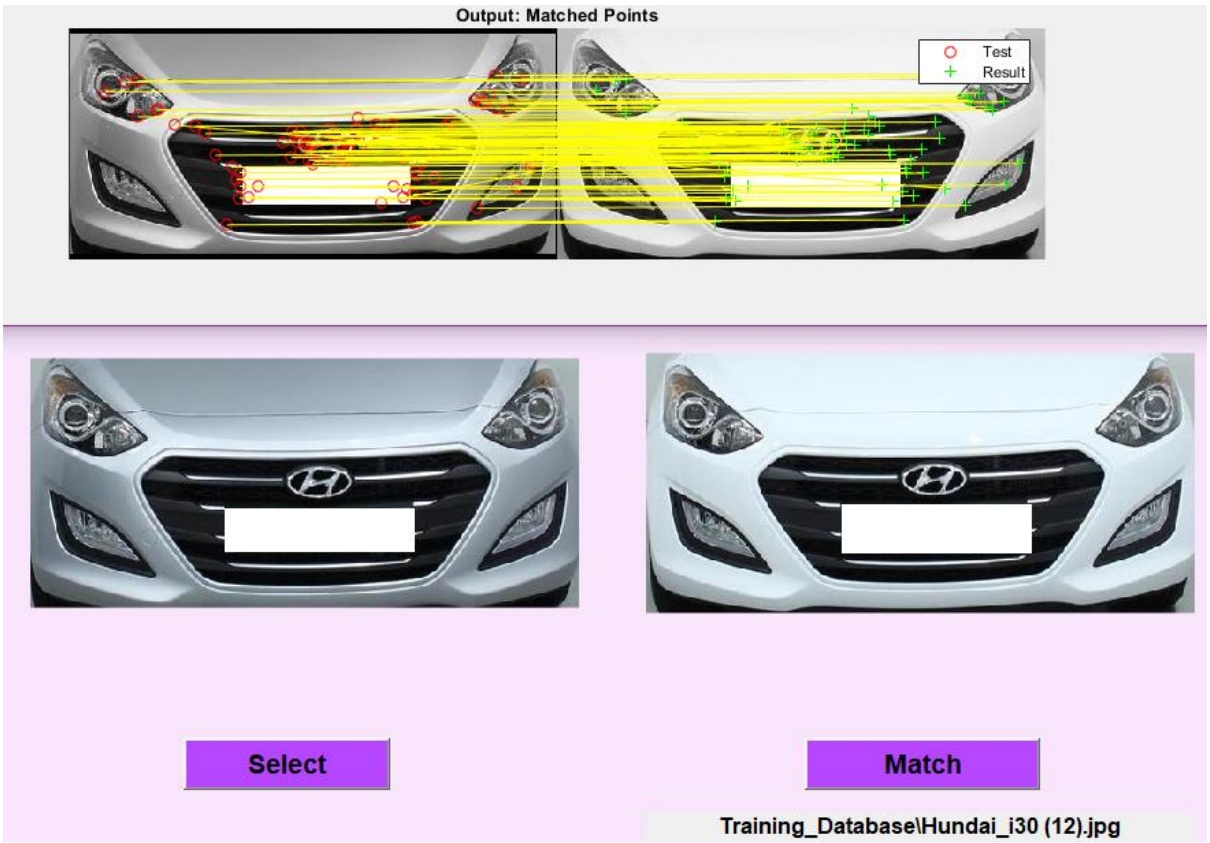


Fig 12.1 Output: Hyundai i30

13. Kia Ceed

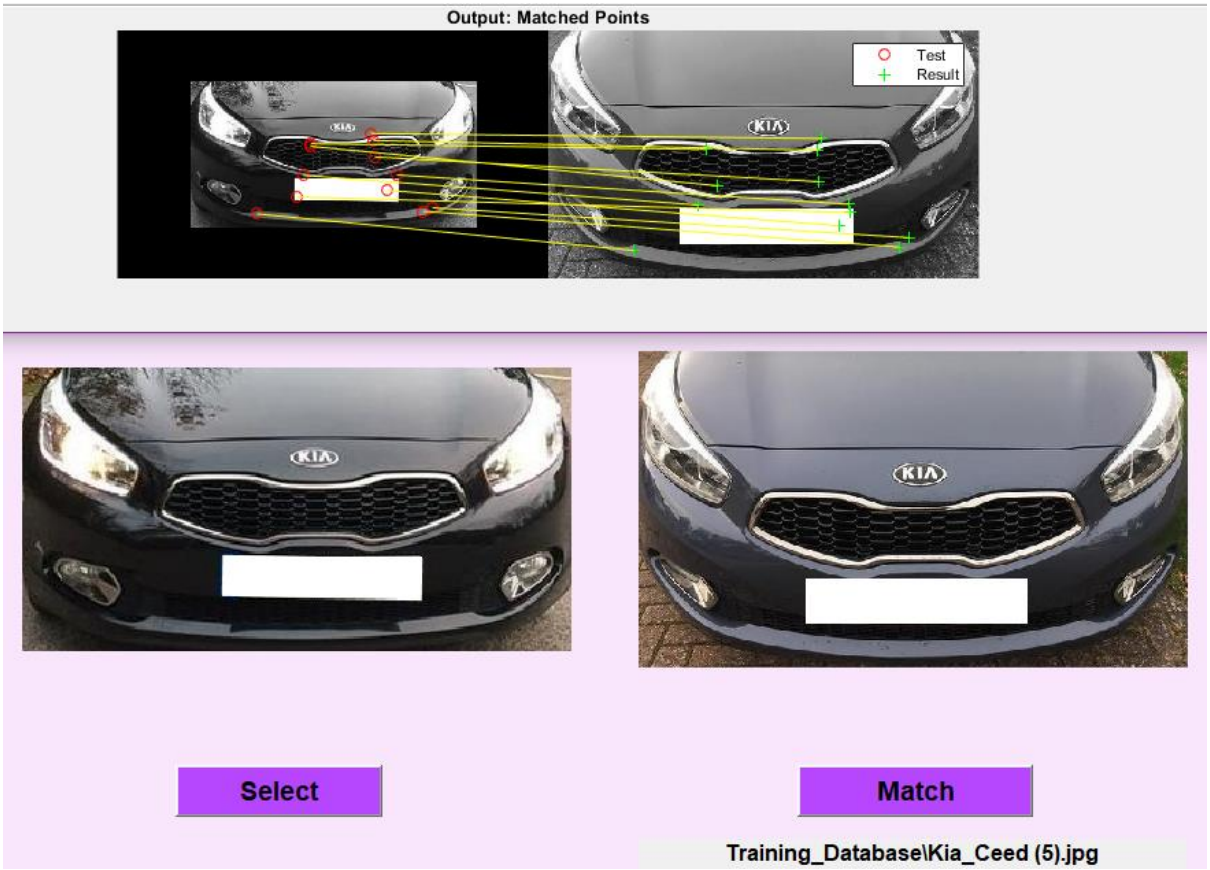


Fig 13.1 Output: Kia Ceed

14. Kia Rio

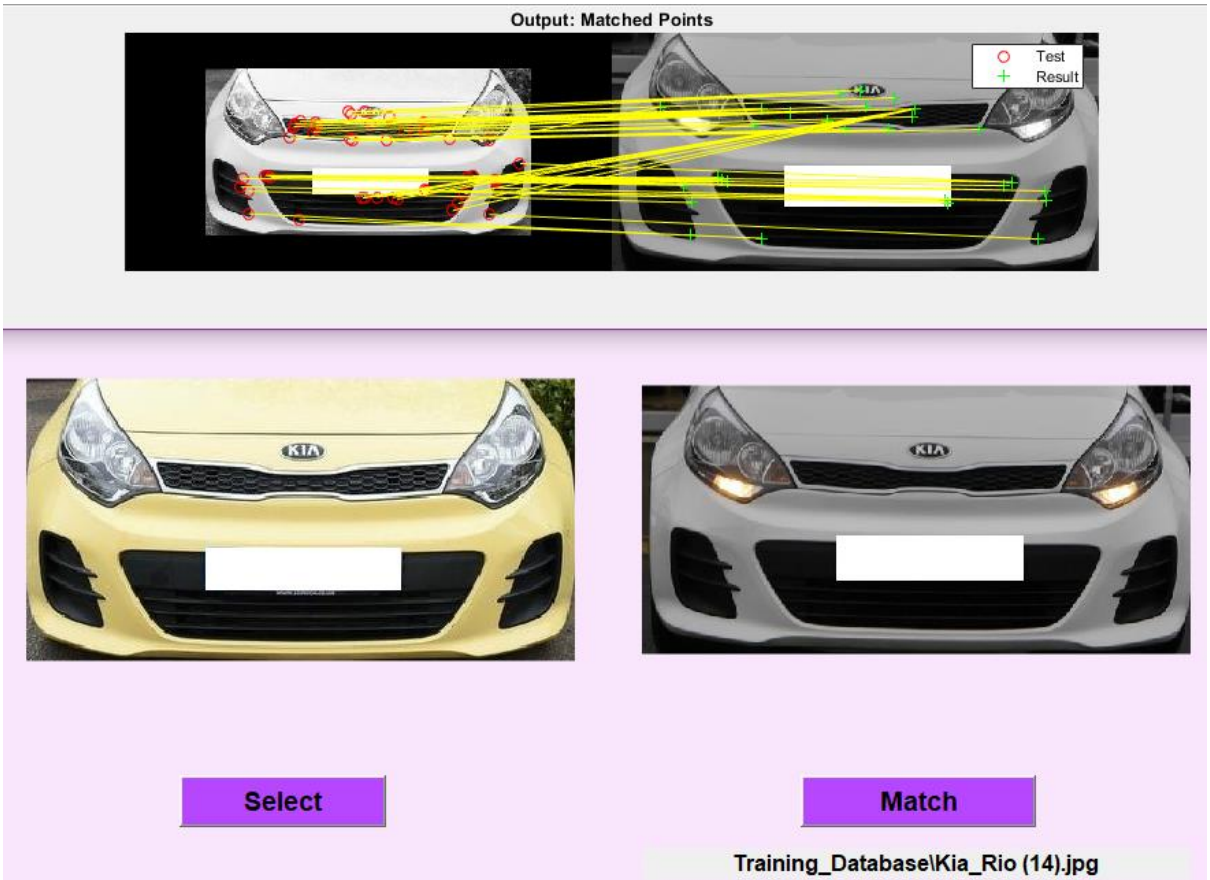


Fig 14.1 Output: Kia Rio

15. Kia Sportage

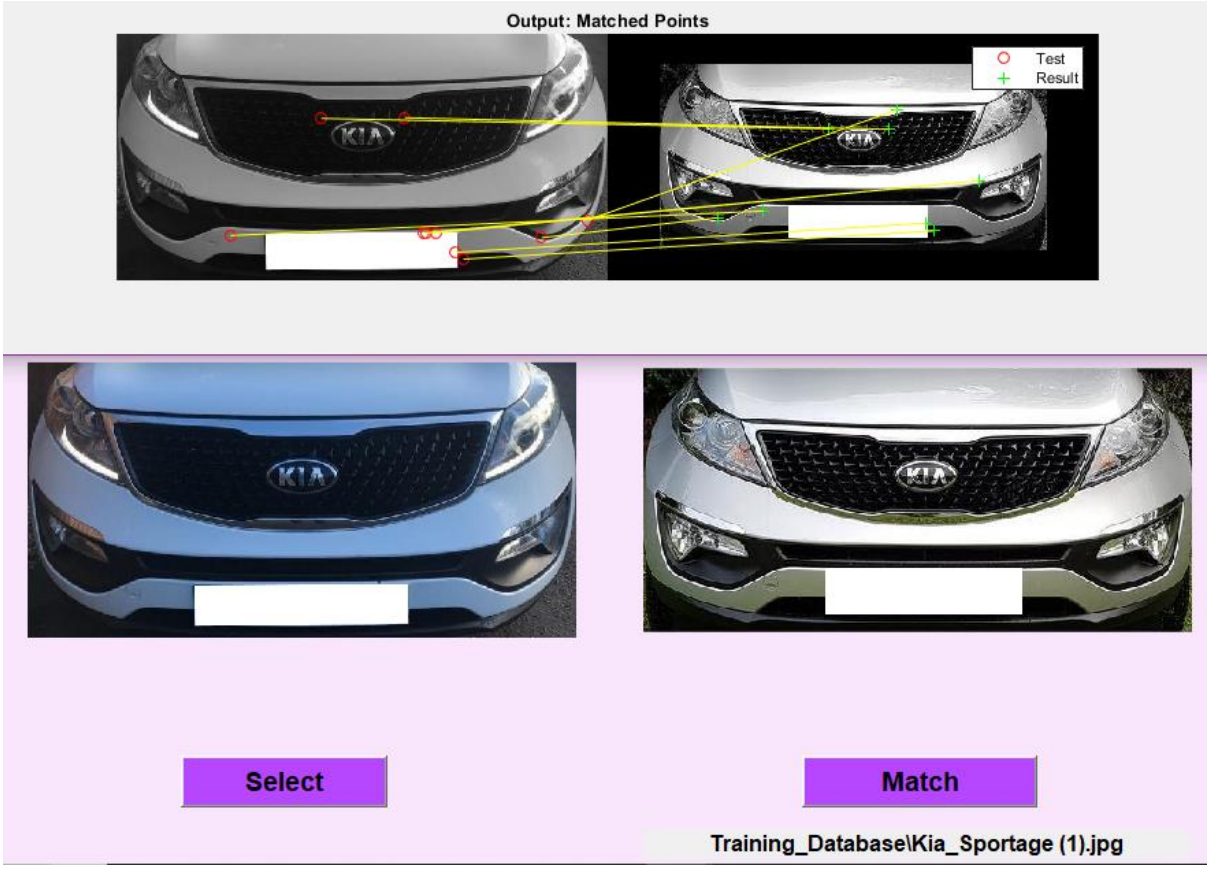
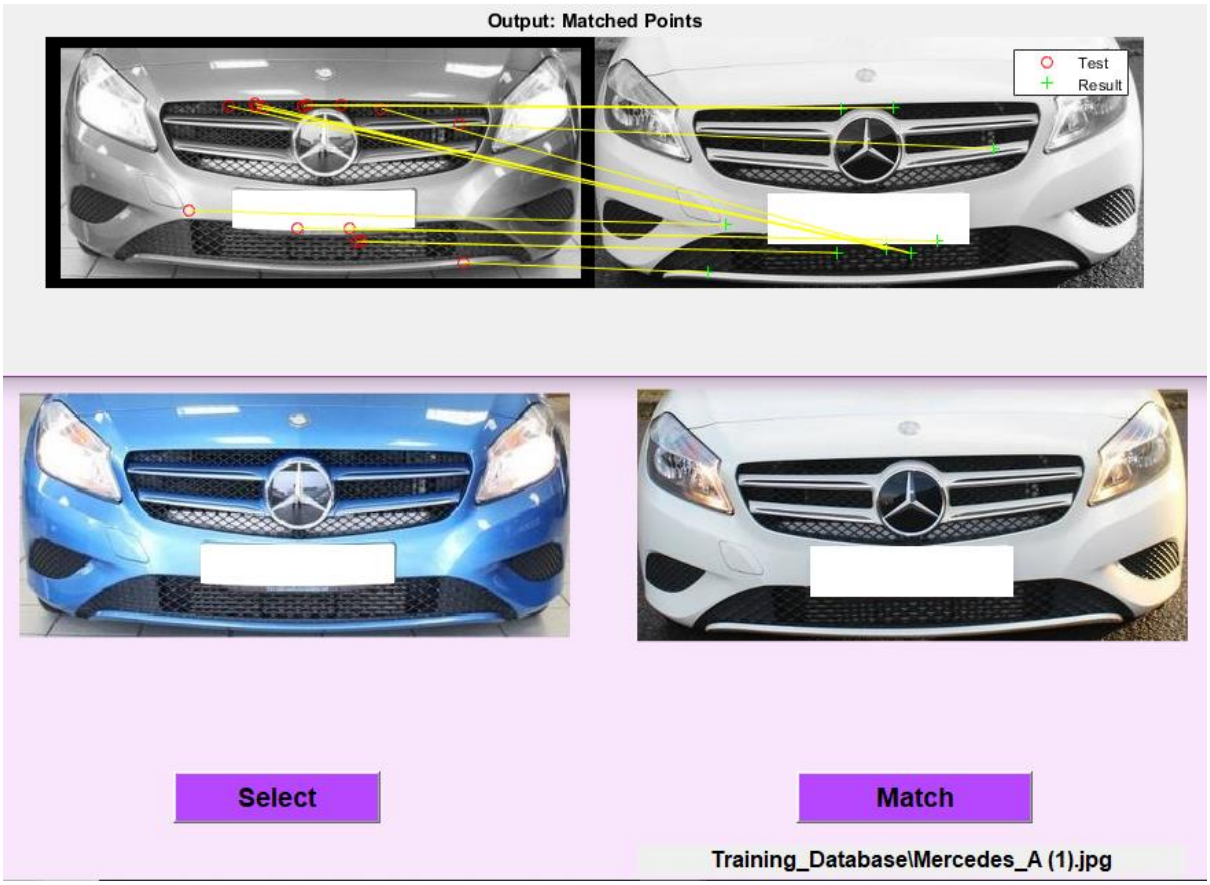
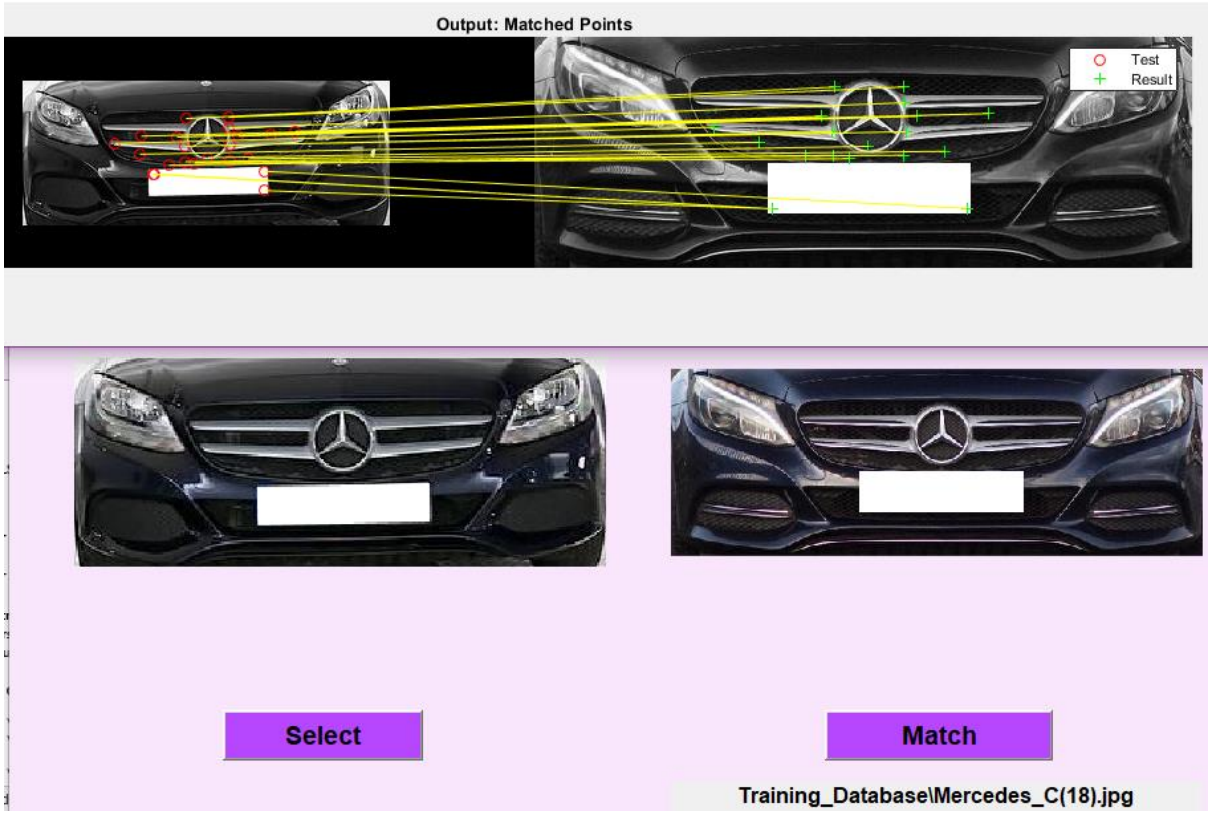


Fig 15.1 Output: Kia Sportage

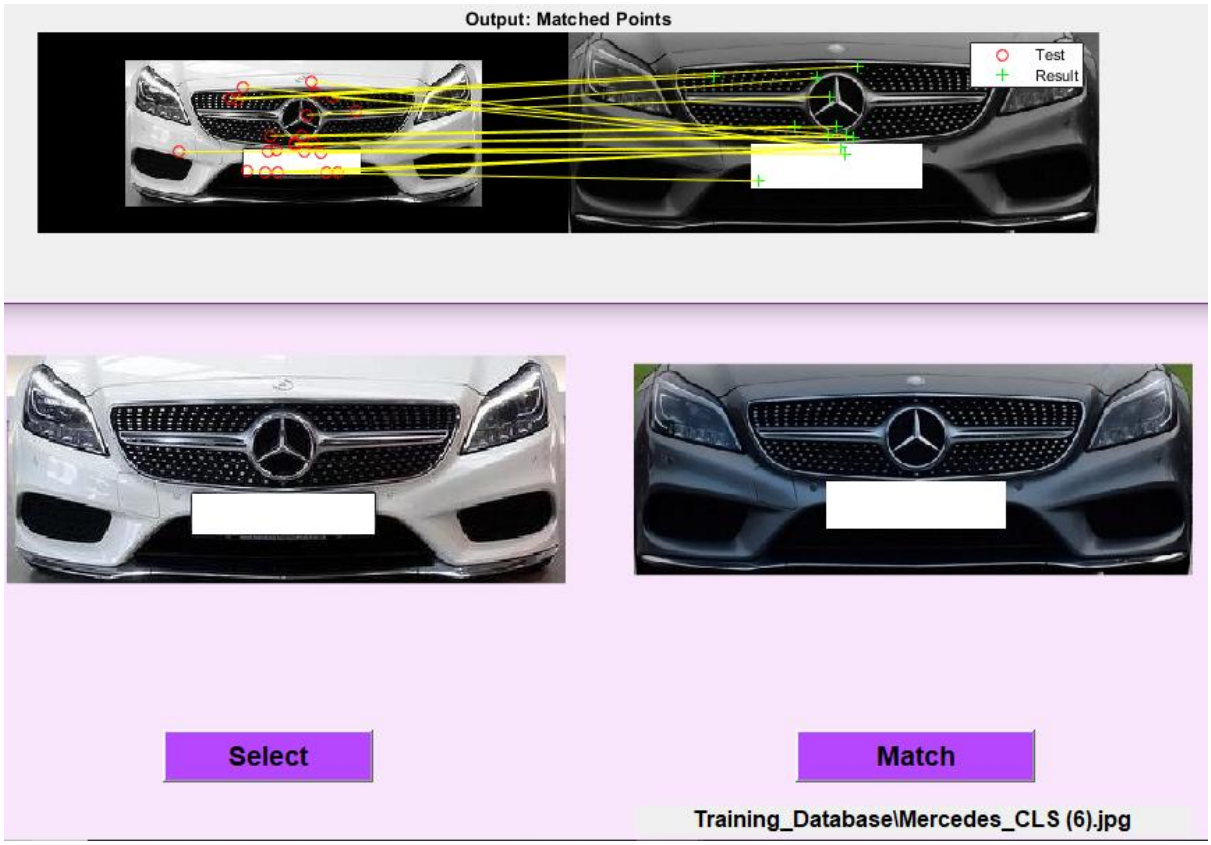
16. Mercedes A



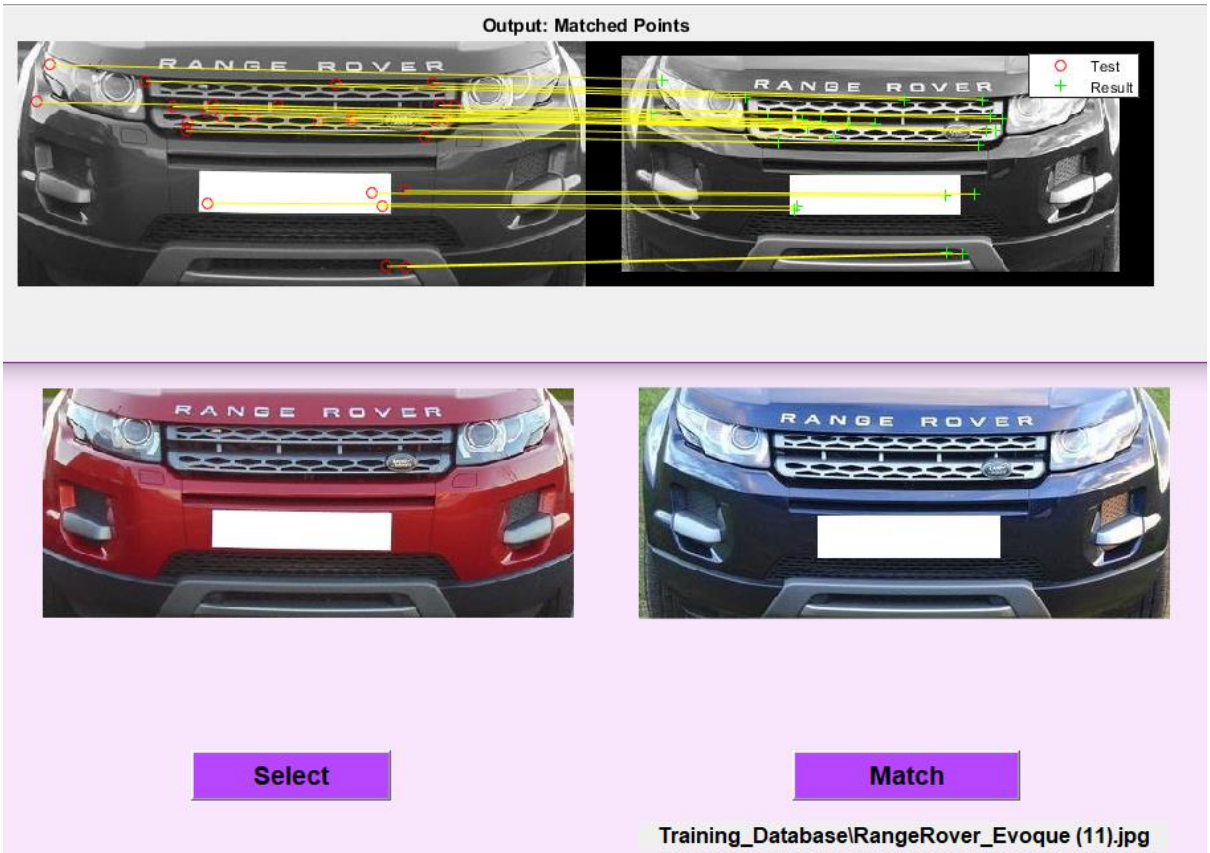
17. Mercedes C



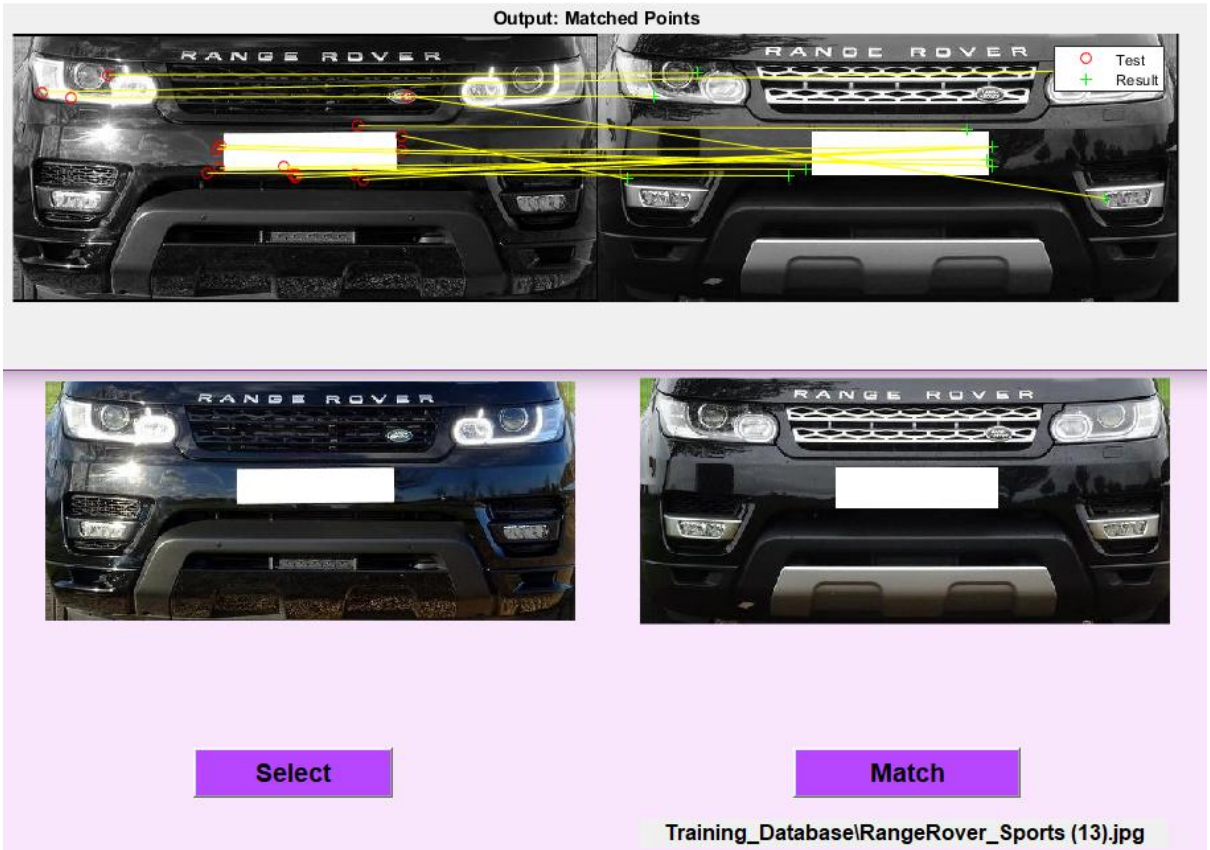
18. Mercedes CLS



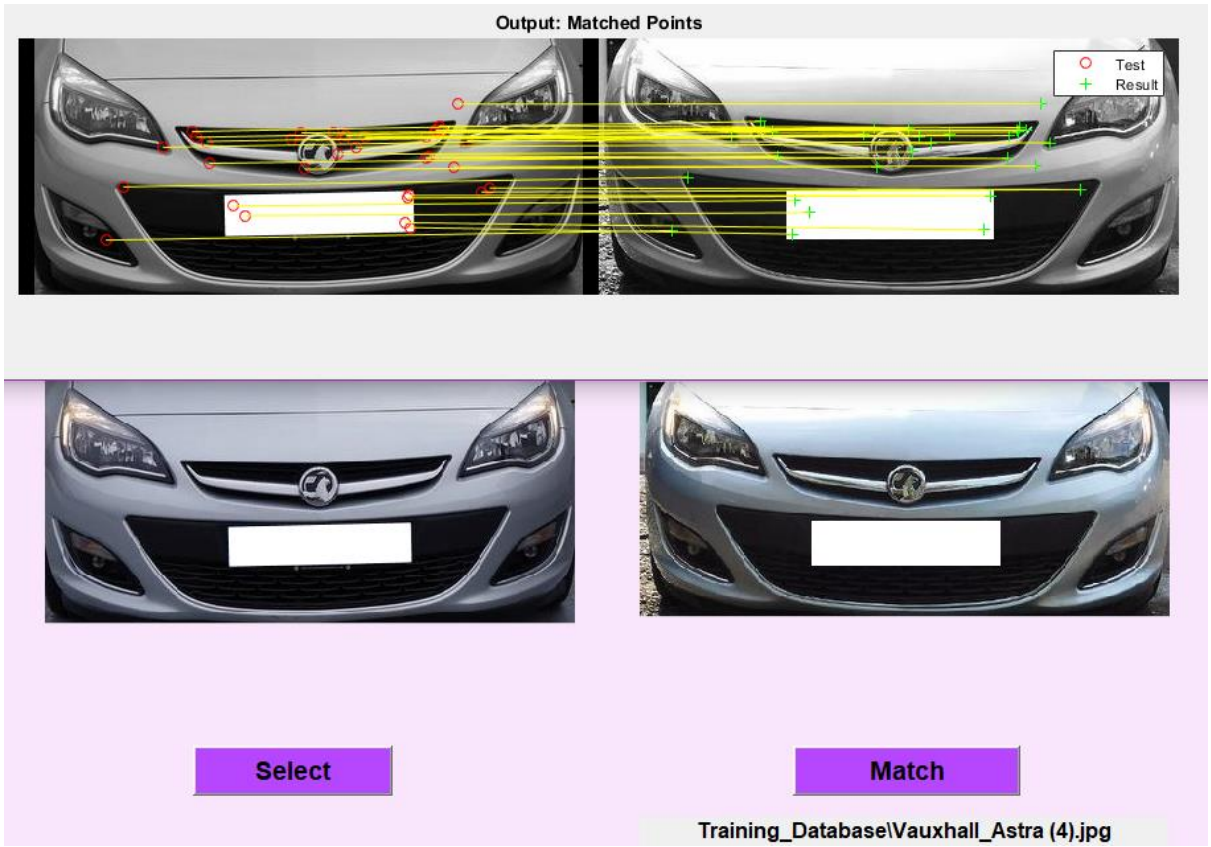
19. Range Rover Evoque



20. Range Rover Sport



21. Vauxhall Astra



22. Vauxhall Corsa

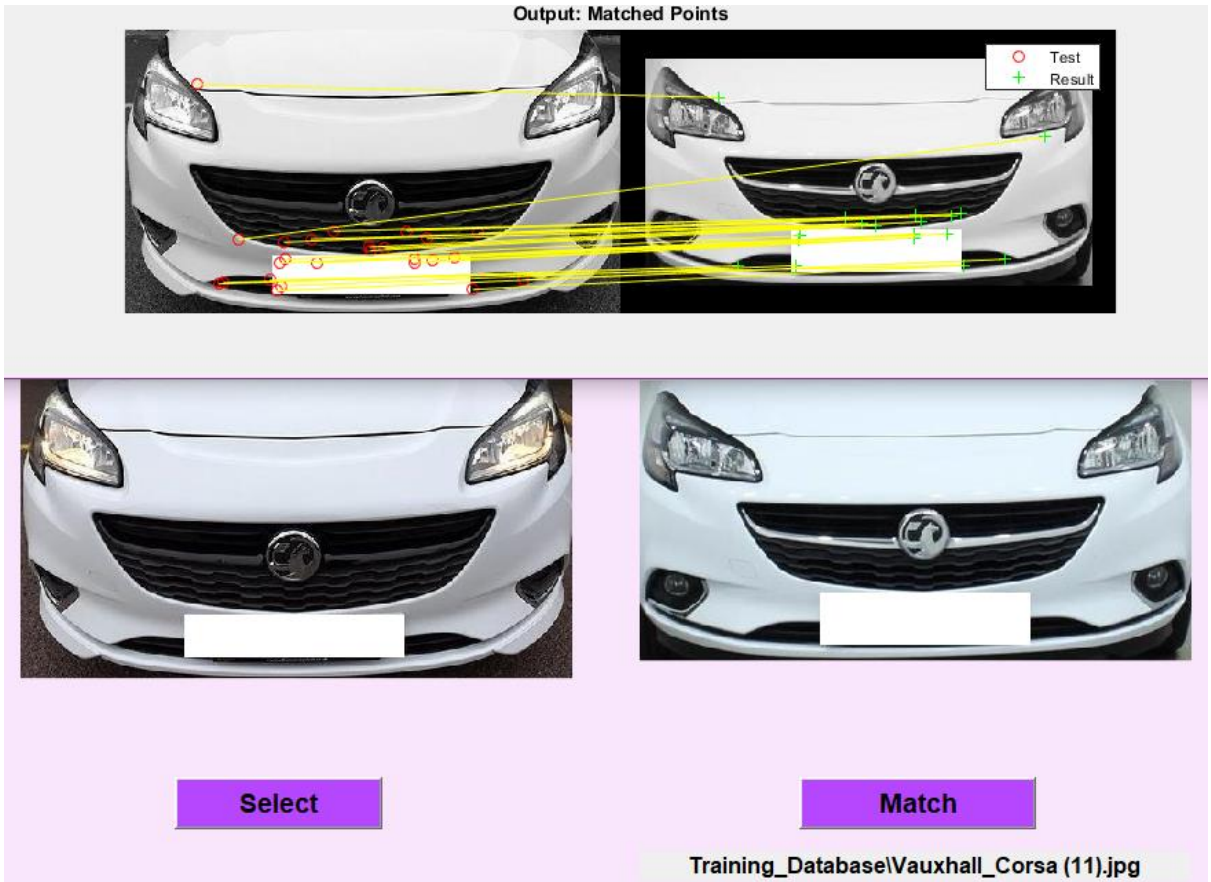


Fig 22.1 Output: Vauxxhall Corsa

23. Volkswagen Golf

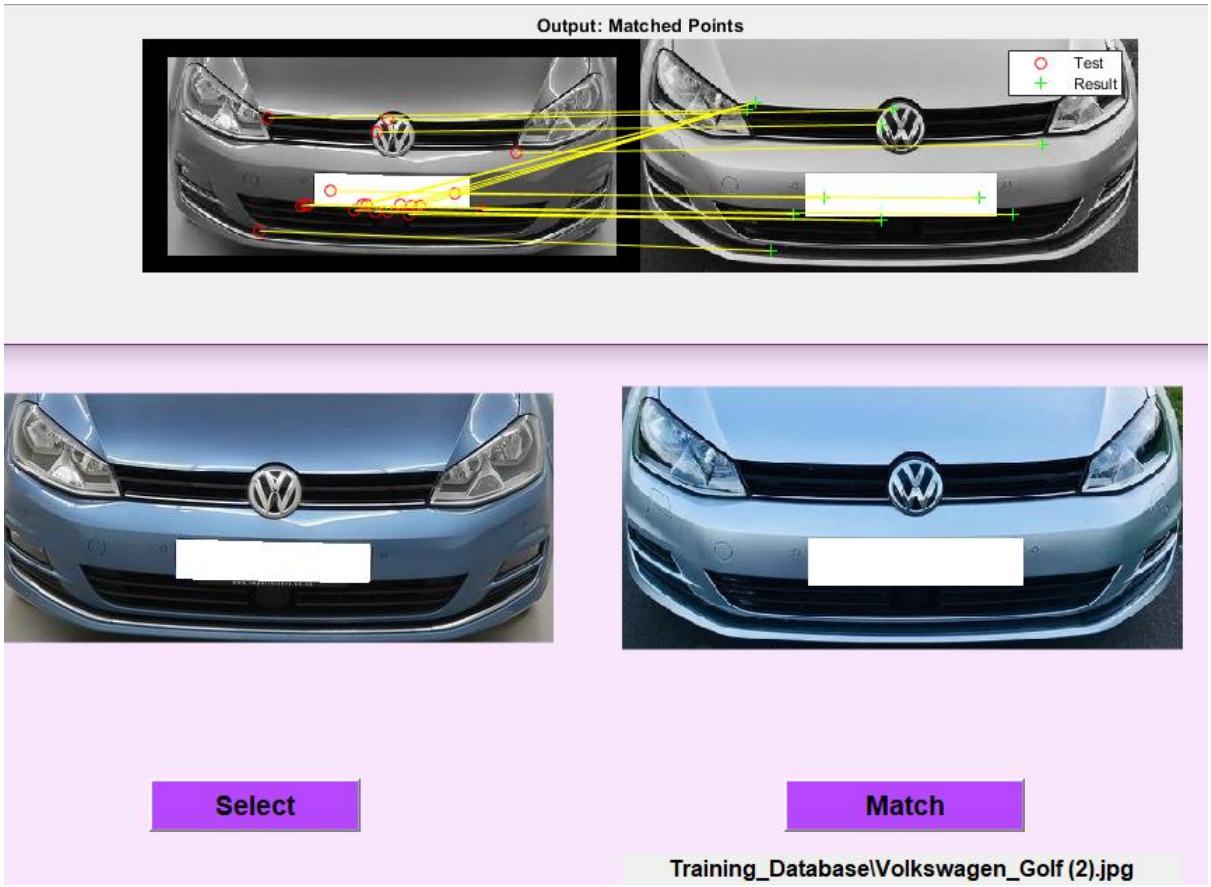


Fig 23.1 Output: Volkswagen Golf

24. Volkswagen Polo

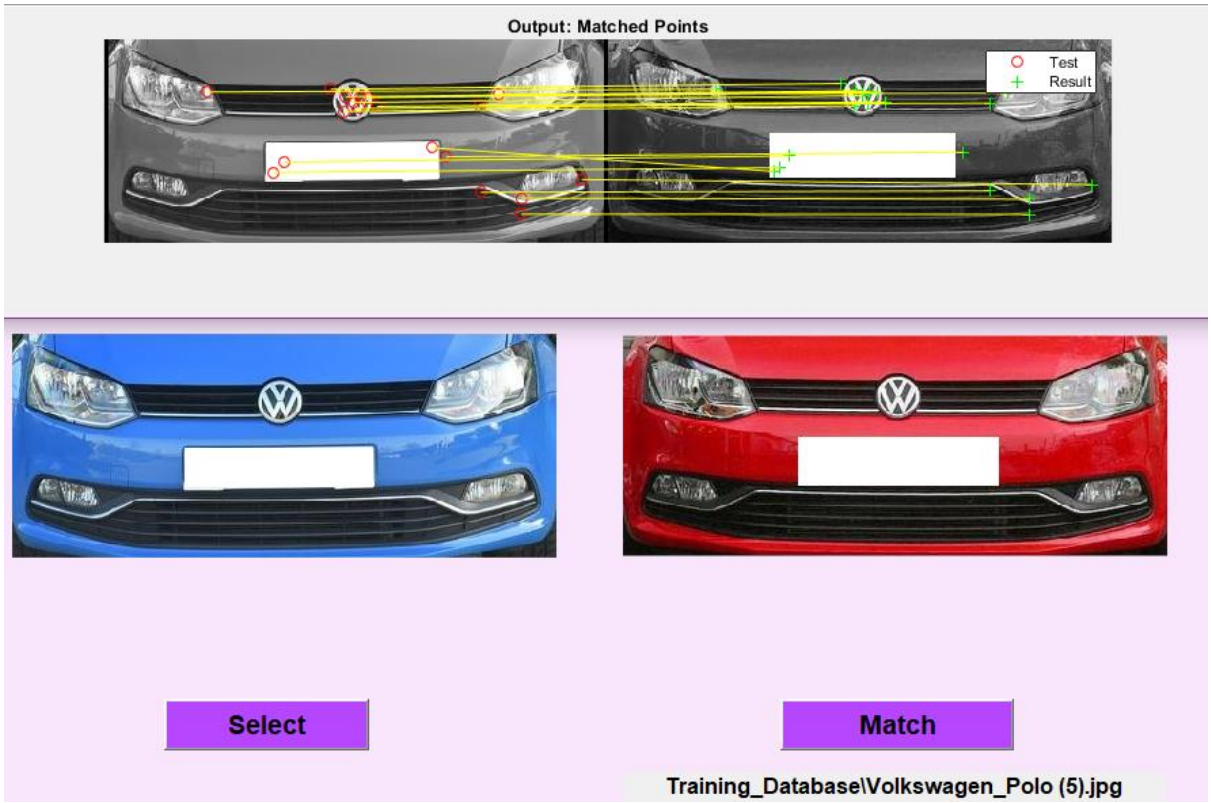


Fig 24.1 Output: Volkswagen Tiguan

25. Volkswagen Tiguan

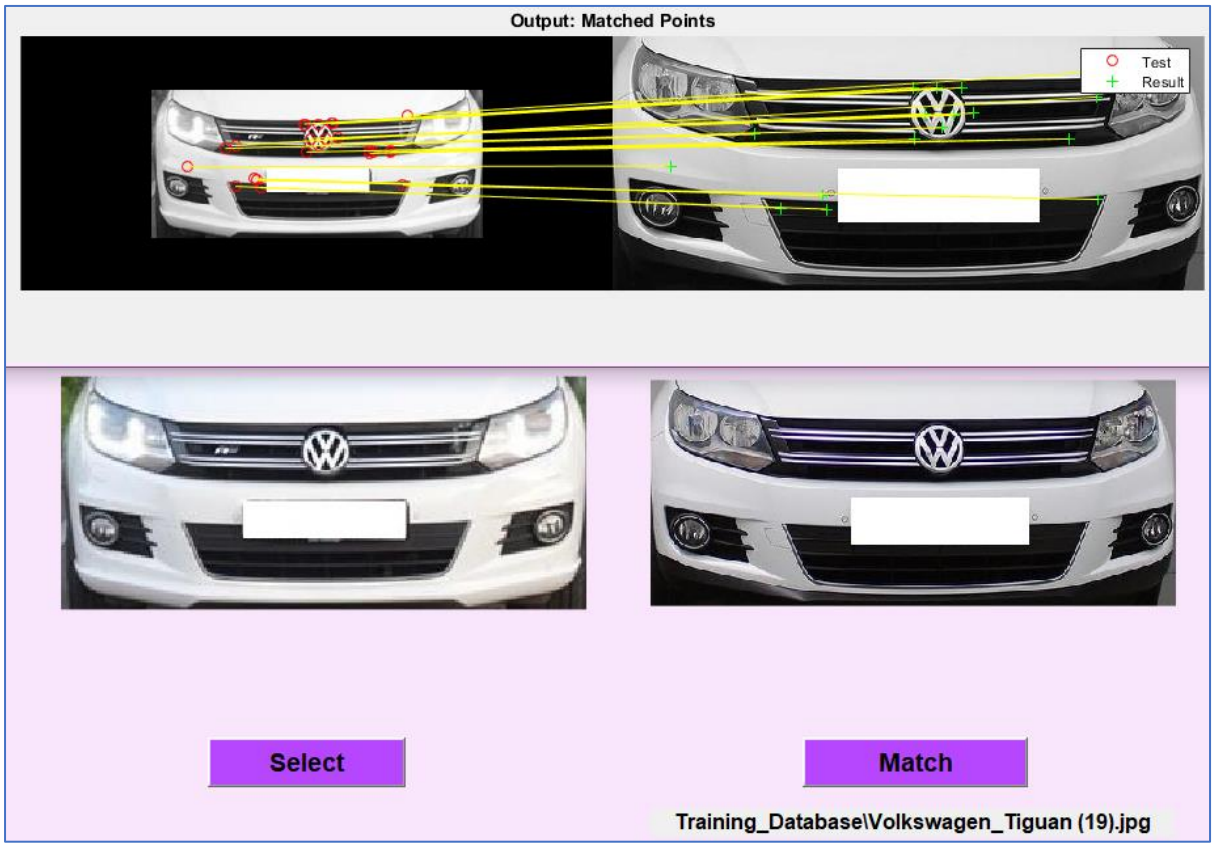


Fig 25.1 Output: Volkswagen Tiguan

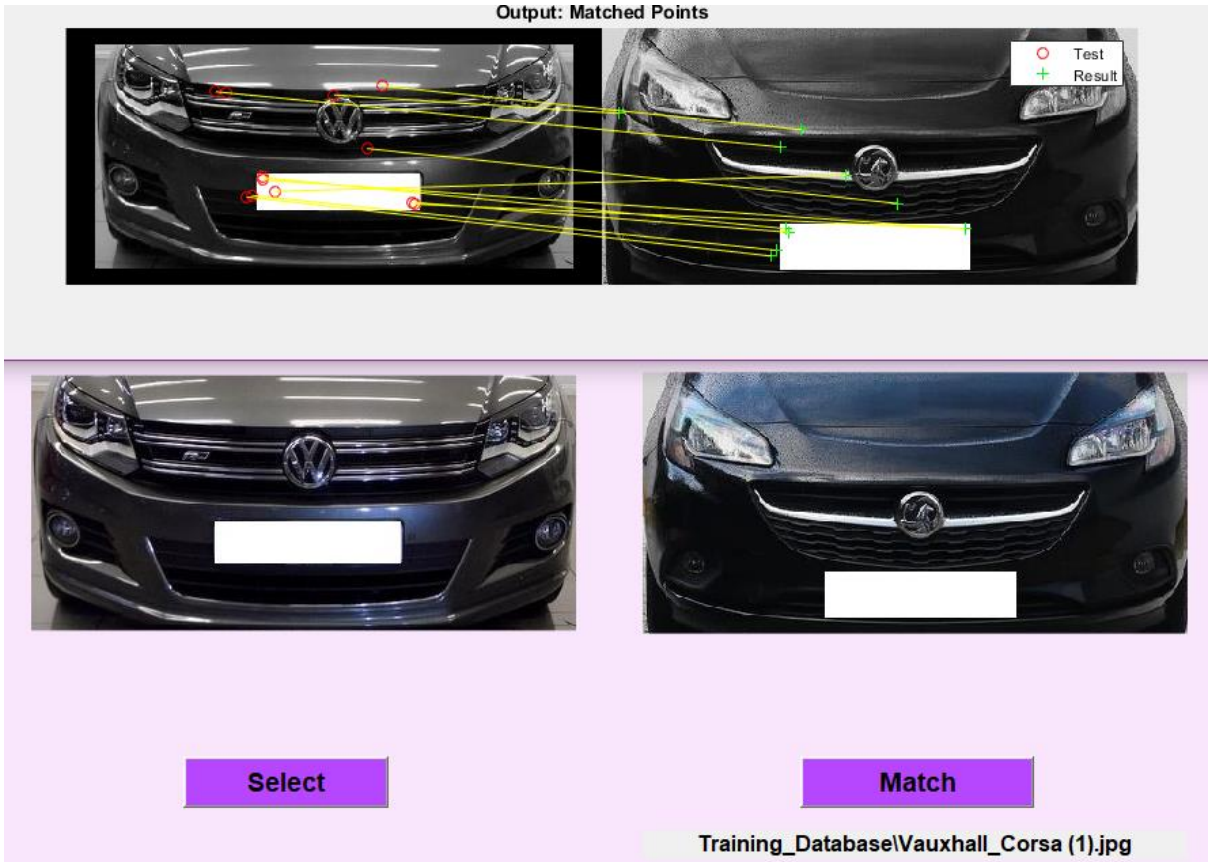


Fig 25.2 Output: **Vauxhall Corsa**: Wrong Result

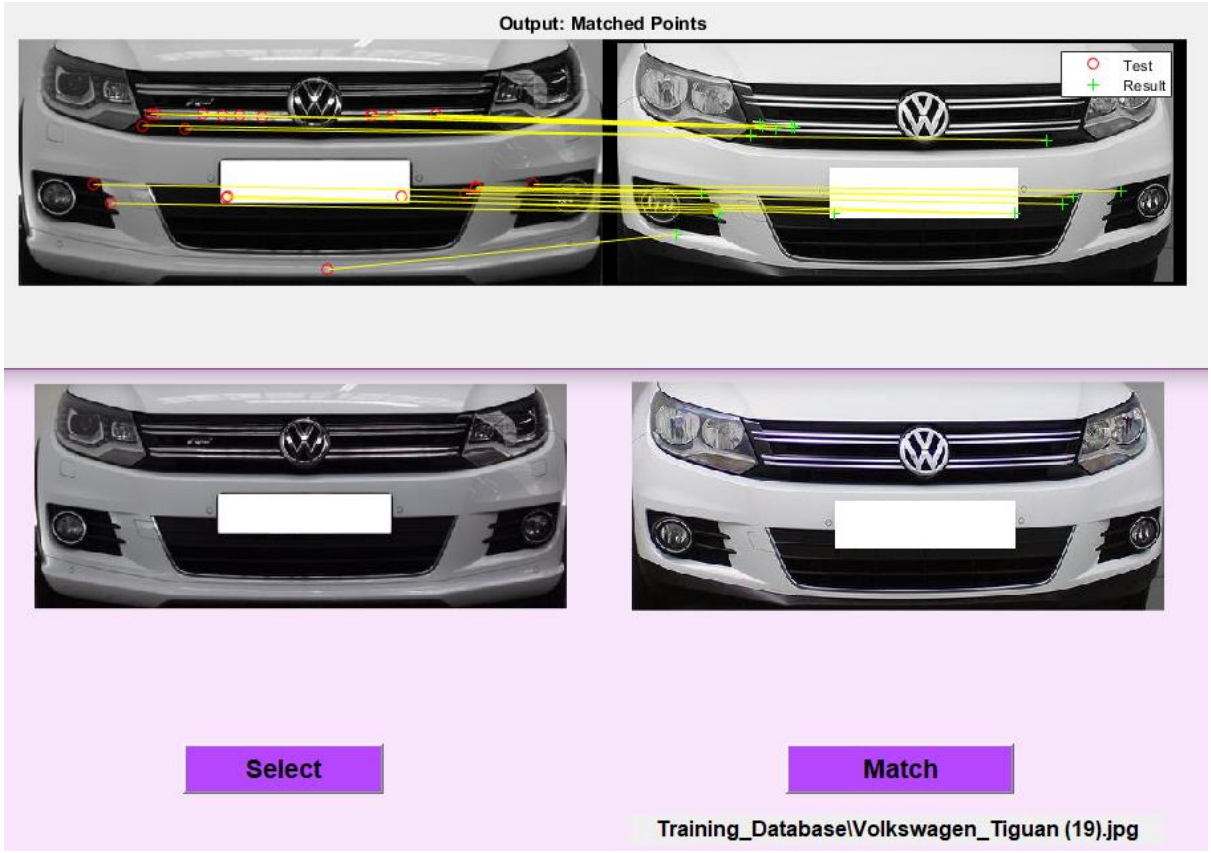


Fig 25.3 Output: Volkswagen Tiguan

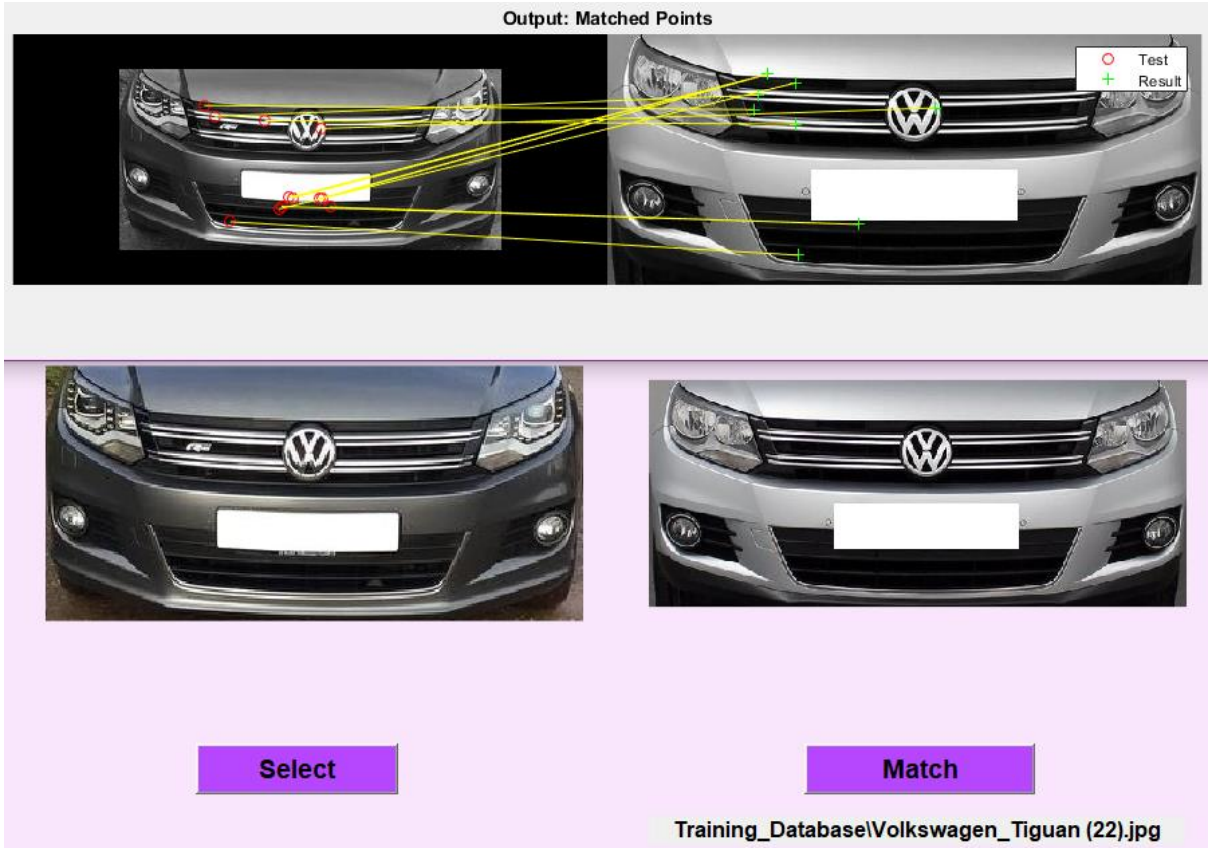


Fig 25.4 Output: Volkswagen Tiguan

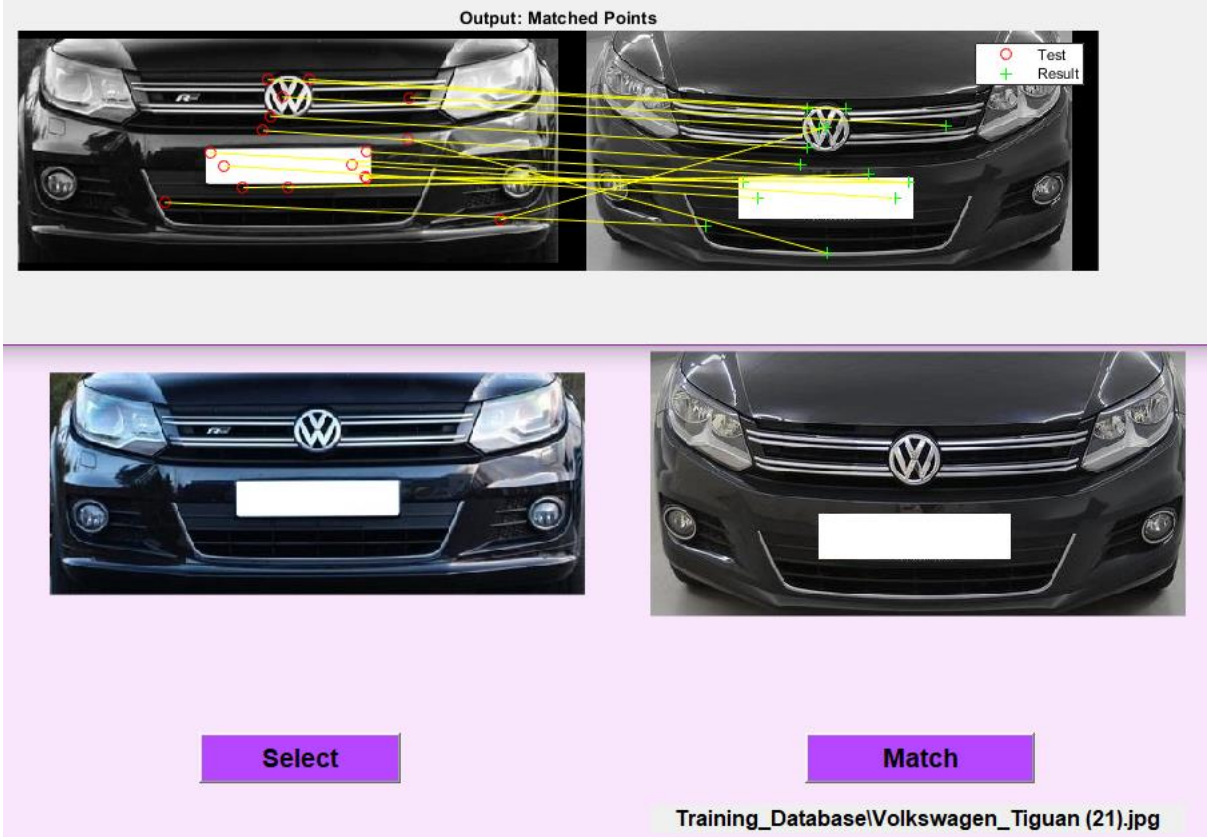


Fig 25.5 Output: Volkswagen Tiguan

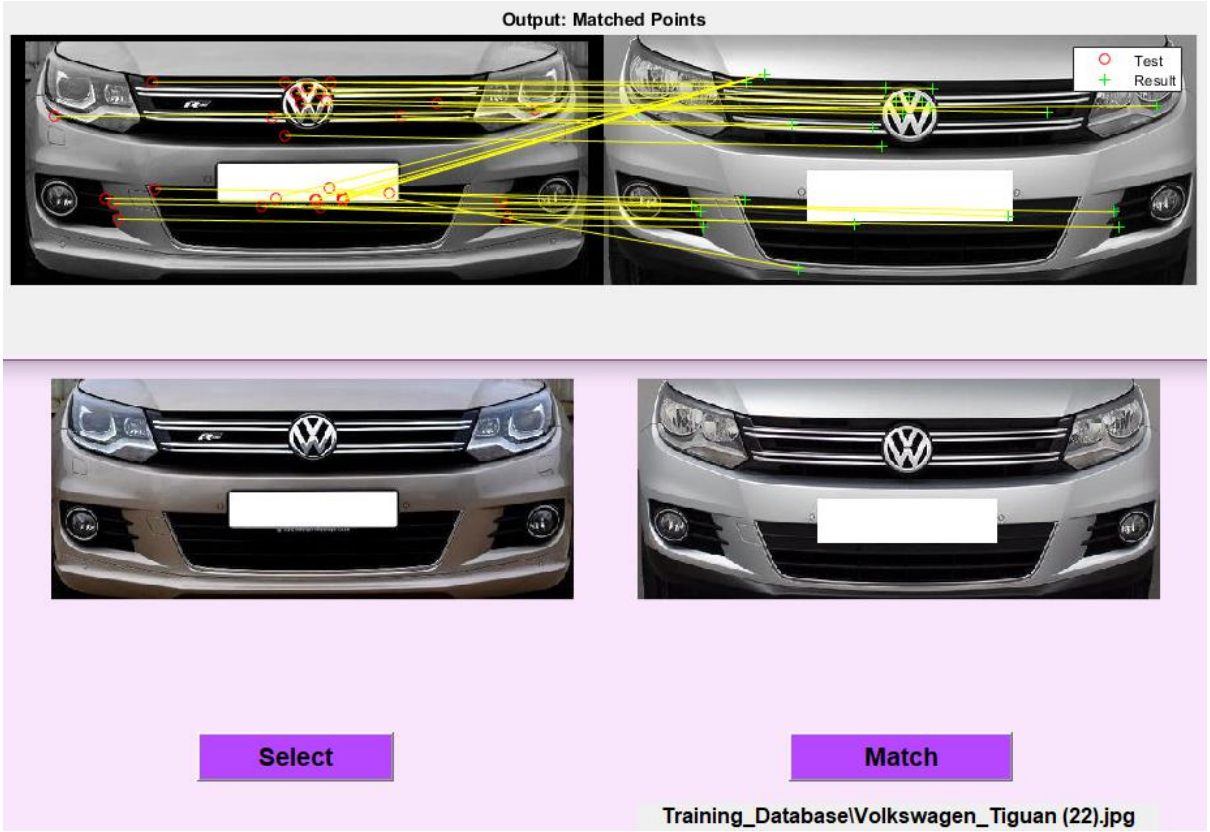


Fig 25.6 Output: Volkswagen Tiguan

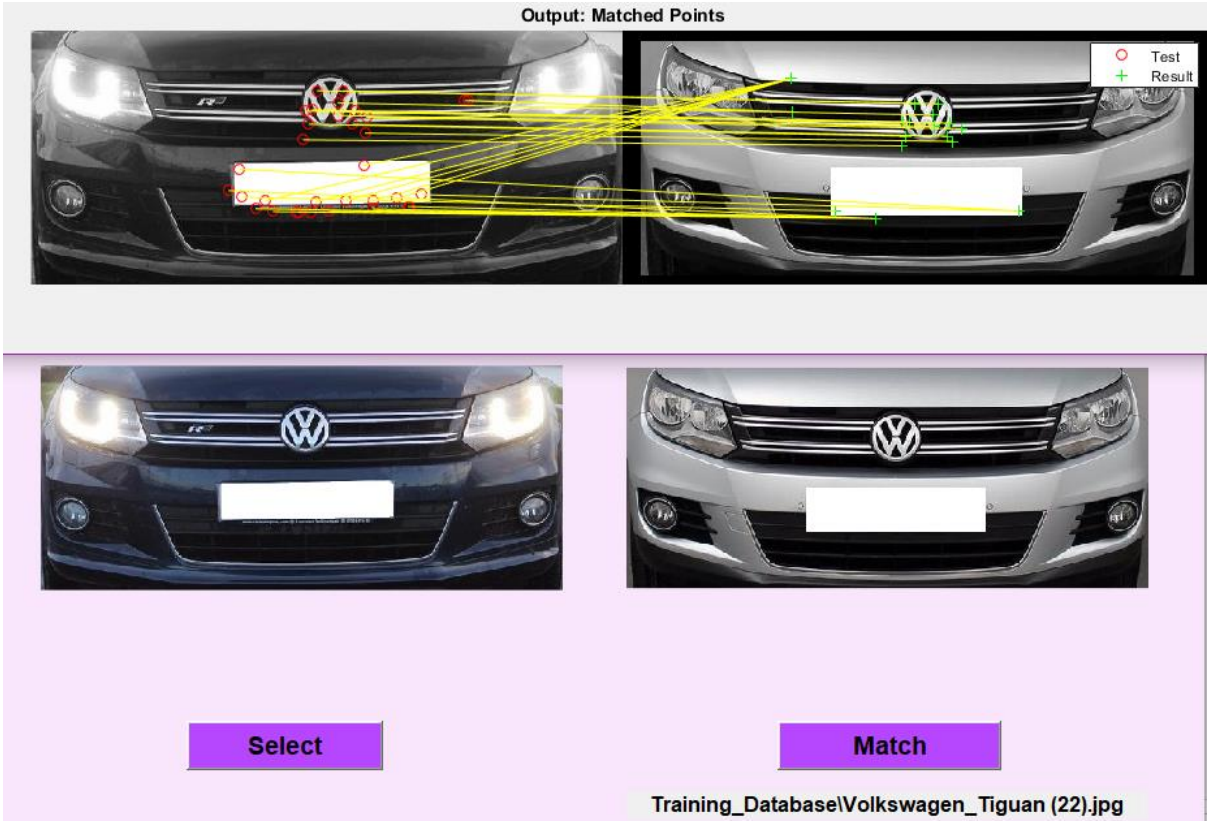


Fig 25.7 Output: Volkswagen Tiguan

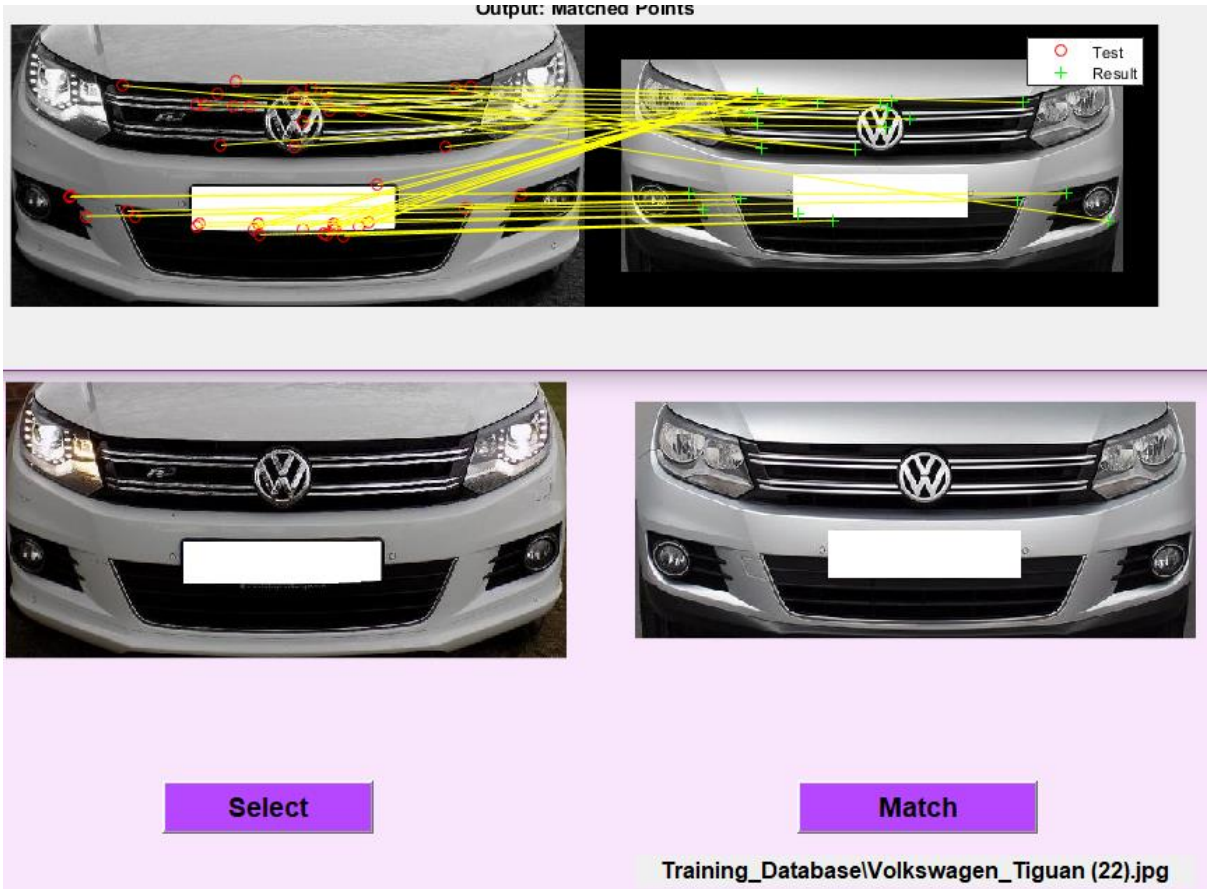


Fig 25.8 Output: Volkswagen Tiguan

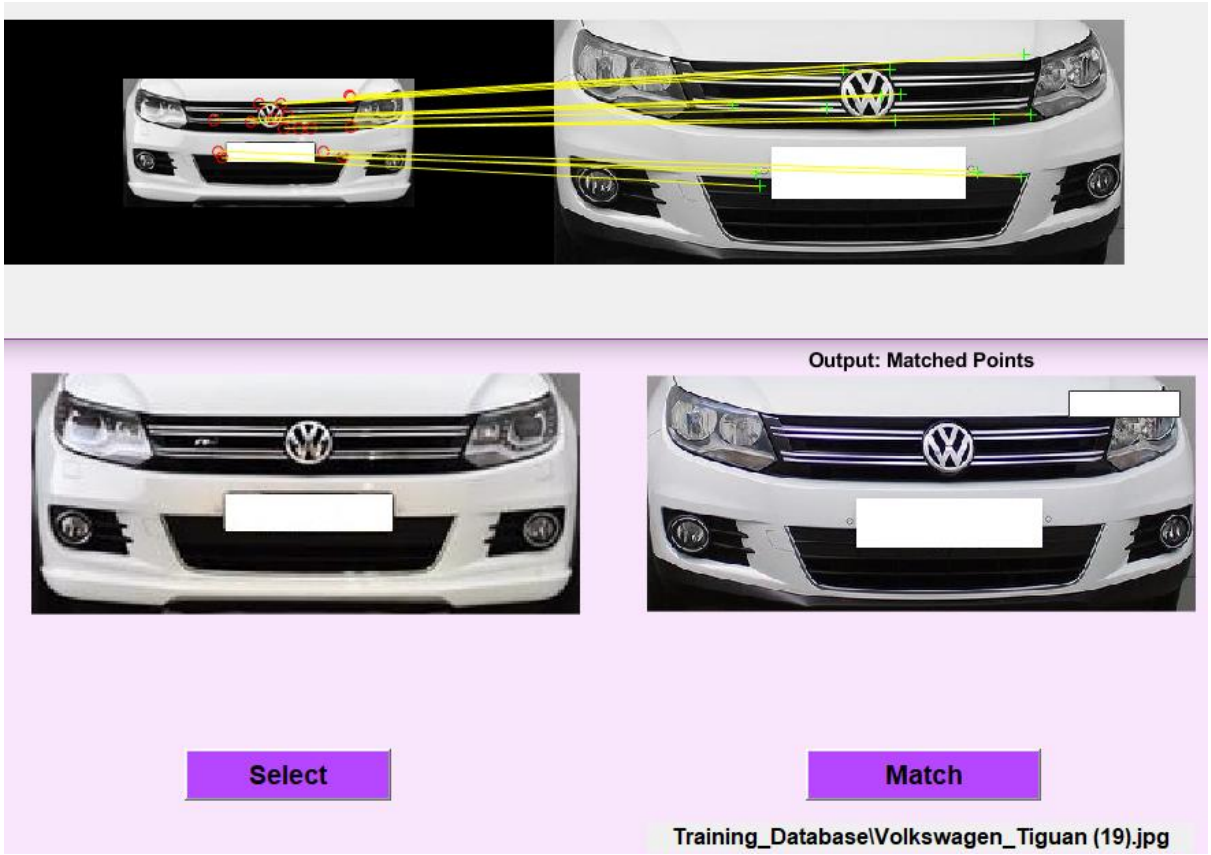


Fig 25.9 Output: Volkswagen Tiguan

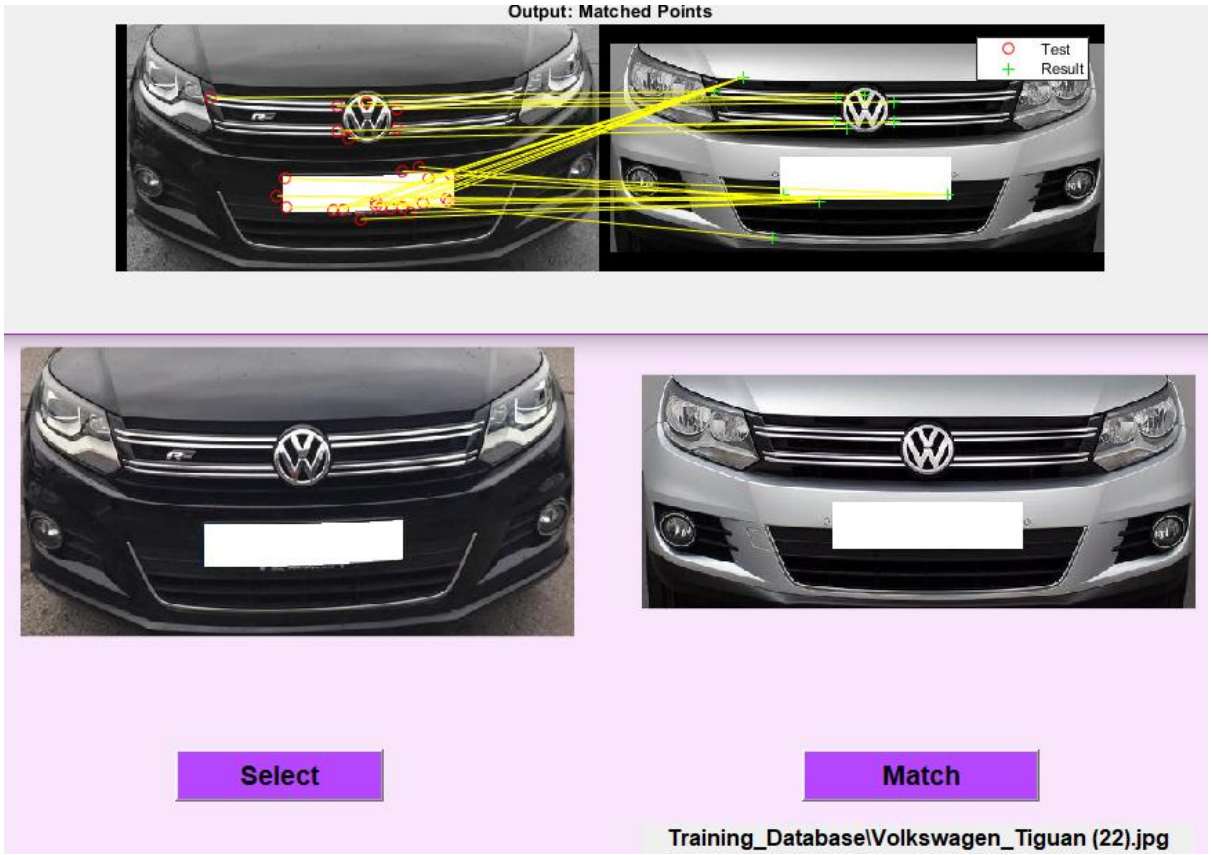


Fig 25.10 Output: Volkswagen Tiguan

References:

1. (En.wikipedia.org, 2019)
   En.wikipedia.org. (2019). Speeded up robust features. [online] Available at: https://en.wikipedia.org/wiki/Speeded_up_robust_features [Accessed 10 Jan. 2019].

2. (Uk.mathworks.com, 2019)
   Uk.mathworks.com. (2019). Find matching features - MATLAB matchFeatures- MathWorks United Kingdom. [online] Available at: https://uk.mathworks.com/help/vision/ref/matchfeatures.html [Accessed 10 Jan. 2019].

3. (Ars.els-cdn.com, 2019)
   Ars.els-cdn.com. (2019). [online] Available at: https://ars.els-cdn.com/content/image/1-s2.0-S0031320314002441-gr3.jpg [Accessed 10 Jan. 2019].

4. (Uk.mathworks.com, 2019)
   Uk.mathworks.com. (2019). Display corresponding feature points - MATLAB showMatchedFeatures- MathWorks United Kingdom. [online] Available at: https://uk.mathworks.com/help/vision/ref/showmatchedfeatures.html [Accessed 11 Jan. 2019].

5. (Nayak, 2019)
   Nayak, S. (2019). Understanding AlexNet | Learn OpenCV. [online] Learnopencv.com. Available at: https://www.learnopencv.com/understanding-alexnet/ [Accessed 11 Jan. 2019].