DESIGN DOCUMENT

# AMPOULDE DEFECT DETECTION

Table of Contents

[AMPOULDE DEFECT DETECTION 1](#_Toc98776474)

[User Interface Design (AI) 3](#_Toc98776475)

[Labels 3](#_Toc98776476)

[Buttons 3](#_Toc98776477)

[Options/Settings 3](#_Toc98776478)

[Setting Configuration 4](#_Toc98776479)

[Setting 4](#_Toc98776480)

[Working 4](#_Toc98776481)

[Steps to Activate 4](#_Toc98776482)

[Connecting and Running 5](#_Toc98776483)

[Action 5](#_Toc98776484)

[Event 5](#_Toc98776485)

[Next Event 5](#_Toc98776486)

[Program Design 6](#_Toc98776487)

[Connection 6](#_Toc98776488)

[Interface Specification 6](#_Toc98776489)

[Architecture of AI Software 7](#_Toc98776490)

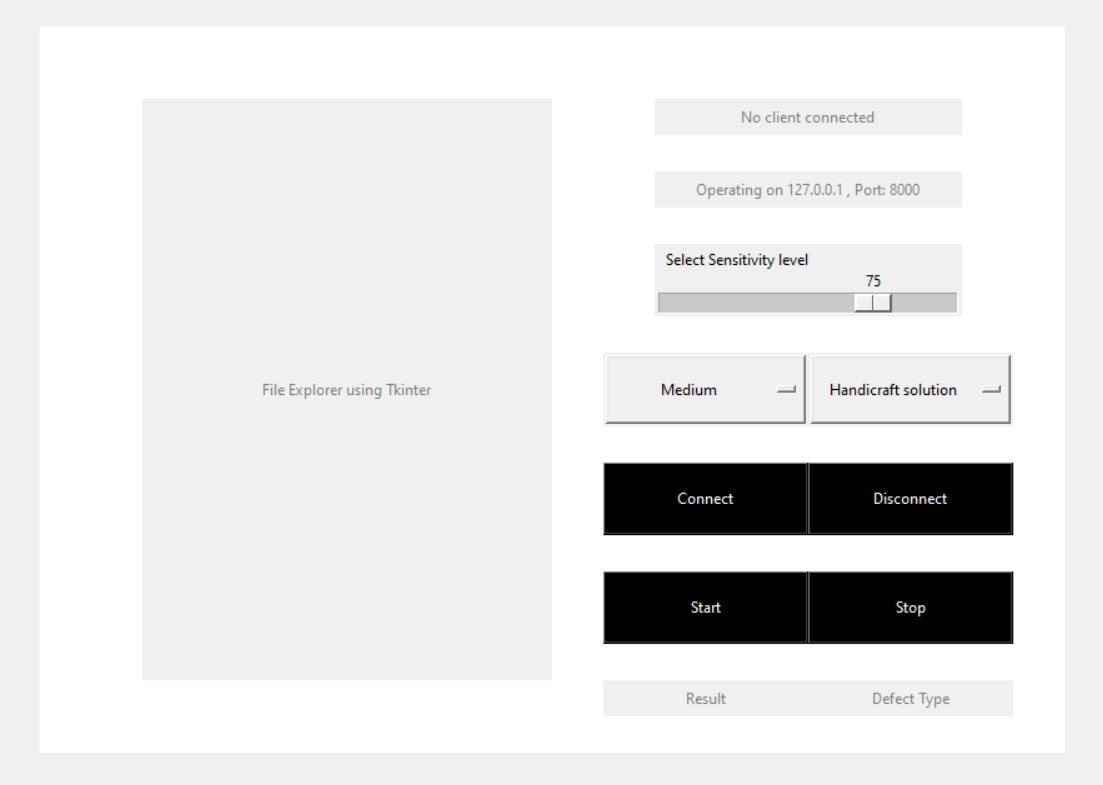
[Database Design 11](#_Toc98776492)

[Flow of Source Code 12](#_Toc98776493)

[Python code: 12](#_Toc98776494)

[.SAV files 14](#_Toc98776495)

# User Interface Design (AI)



**O3**

**O2**

**O1**

**B4**

**L4**

**L5**

**B3**

**B2**

**B1**

**L3**

**L1**

**L2**

### Labels

1. Connection status
2. Server operating Address and Port
3. Image display
4. Result
5. Defect type

### Buttons

1. Connect
2. Disconnect
3. Start
4. Stop

### Options/Settings

1. Method to use (Handicraft/SVM solution)
2. Sensitivity option: Low, Med and High. (only for SVM solution)
3. Sensitivity level 0 (Lowest) to 100 (Highest) to defects. (Only for Handicraft solution)

## Setting Configuration

|  |  |  |
| --- | --- | --- |
| Setting | Working | Steps to Activate |
| Handicraft Solution (Default setting) | * This setting can be found on O1. This is the solution specially designed for detecting defects in ampoule bottles. * This setting comes with the sensitivity scale in O3. The higher the sensitivity on this scale the smaller the defects our solution is able to detect. At low sensitivities only big defects will be detected. | 1. Go to O3 and choose how sensitive to defects do you want the AI solution to be.      1. Go to O1, select Handicraft solution.   Now you are ready to start detection in this mode. |
| SVM solution | * This setting can also be found on O1. This is the second solution we provide which detects defects on the bottles using state of the arts method being used today. * This setting comes with sensitivity menu in O2. This menu has three levels of sensitivity: Low, Medium and High. This decided how sensitive to defects you want the AI tool to be. | 1. Go to O2 and select your sensitivity option from the drop down menu. You can choose Low, Medium or High.  1. Go to O1 and select SVM solution from the drop down menu.   Now you are ready to start detection in this mode. |

## Connecting and Running

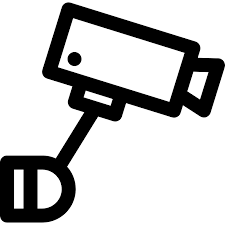
|  |  |  |
| --- | --- | --- |
| Action | Event | Next Event |
| Connect | AI software will start listening if there is a software/Client ready to connect on the server address and port specified on Label 2. | If a software is available, the AI software will connect to it and show the Client address and connection status on Label 1.  The AI software will now block any further connections. |
| Start | If connected to a client, the AI software will start receiving any data transferred from the client to the server. The data comes in two parts   * Size of the incoming image in bytes * The incoming image itself.   The app is now in detection mode. | The AI software will perform the selected mode of detection from O1, O2 and O3 (see setting configurations table above) and determine the result of the image. The result status: Clean/Defected will be displayed on Label 4 and the defect type, if any, will be displayed on Label 5.  The resulting image with defects highlighted, if any, will be displayed on Label 3.   * Blue highlight for Line defects * Red highlight for Scratch defects * Purple highlight for Imprint defect.   Now the app will be ready to receive more data and perform the processing over again. |
| Disconnect | AI software will close all connections from the current client/software. | AI software is now ready to make a new connection. |
| Stop | The AI software will break out of detection mode. It will stop receiving data from client. | AI software will wait for “Start” to continue receiving data and performing detection. |

# Program Design

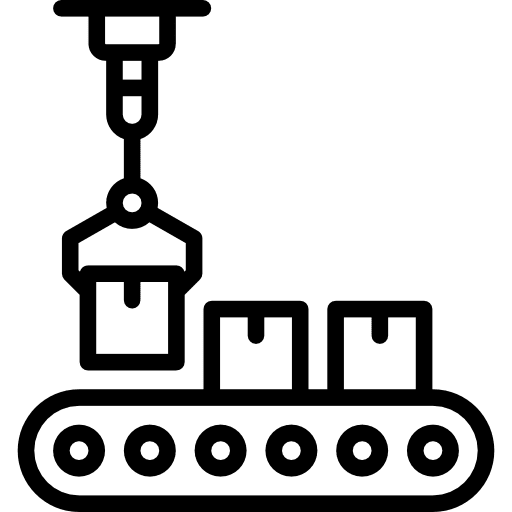
## Connection

The AI software operates on address 127.0.0.1, Port 8000. This is also displayed on the App on Label 2. After pressing the “Connect” button on the AI app, the “connect” button on the vision software is also clicked so that it can connect to our AI software.

## Interface Specification



**Vision system**



**Eject module**

(1) acquisition

(2) vision analysis

(6) eject signal

**Analysis system**

**Machine Controller**

(5) eject signal

Vision Software

(3) Image Transfer

(4) Result

AI Software

After a connection is established between the Vision software and the AI software, the following steps are carried out:

1. The Vision software acquires data from the vision system.
2. It analyses the image and turns it into the required form for our AI software.
3. It transfers the image to our AI software by first transmitting the size of the incoming image in bytes, and then transferring the image itself in bytes. After the AI software receives the amount of bytes equivalent to the size of the image, it will stop receiving and will process the image and generate a result for the image as Clean (0) or Defected (1). Now it will start listening for a new image to process.
4. The result signal is sent back to the Vision software through the already established TCP socket connection and the Vision software interprets the signals as 0 for Okay and 1 for No Go.
5. In case of NG (No Go) signal, the vision software sends an eject signal to the machine controller
6. The machine controller in turn sends an eject signal which causes the eject module to eject the specific ampoule from the production line.

## Architecture of AI Software

Once the software receives the image, following things are checked based on which we apply our different algorithms:

1. Method to apply (Option 1 on App)
   1. Handicraft solution (By default)
   2. SVM solution
2. Sensitivity
   1. In case of Handicraft solution, the sensitivity scale of 0-100 (Option 3) lets the user choose the desired sensitivity to defects. Default is 75.
   2. In case of SVM solution, the drop down menu of sensitivities (Option 2) lets the user choose the desired sensitivity to defects. Default is Medium.

These parameters are pertinent to the way the image is processed and analysed to find defects.

#### **Method 1: Handicraft Solution**

#### Detecting Defect:

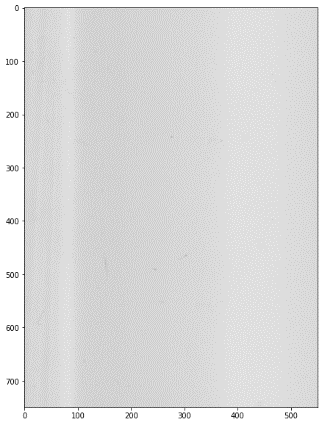
Result

Noise Cancellation

Edge Detection and Refinement

Image Pre-processing

1. Firstly as part of pre-processing all images are cropped and resized to remove the bottle edges from the images.
2. Next we take the sensitivity is taken into account.
   1. Sensitivity < 15: no defect is detected
   2. 15 < Sensitivity < 90: Defects like lines or scratches are detected and the size of the defect detected depends on where the sensitivity lies in between these points.
   3. Sensitivity > 90: At this high sensitivity small defects like Imprints, are also checked for in the image.
   4. Sensitivity > 95: All defects, even the most minor ones, are detected.
3. For line and scratch defects, we detect all the edges present in the image. (using Canny Edge Detector)
4. On the resulting binary image we perform different morphological operations such as dilation, erosion to make the edges more defined and remove any small inconsequential edges, to ensure only the meaningful edges remain.
5. For noise cancellation we perform thresholding by comparing the intensity levels in the original image of the edges found. This way any edges which had intensities closer to the normal intensities of the bottle are removed.
6. Then we perform noise cancellation by removing defects that have a size, smaller than a certain threshold. This threshold is decided by the sensitivity level we mentioned above.
7. In case of sensitivity > 90, we perform imprint detection, for which threshold the original image. We use two distinct properties of imprints to detect them:
   1. Imprints are at a very low intensity than the rest of the image and defects
   2. They have a specific size range
8. By checking these two parameters we detect any imprints present in the image.



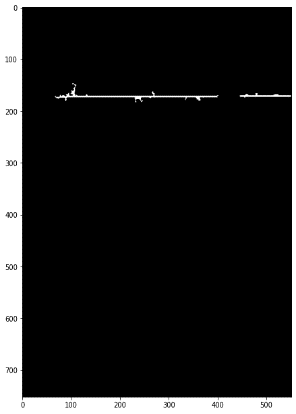
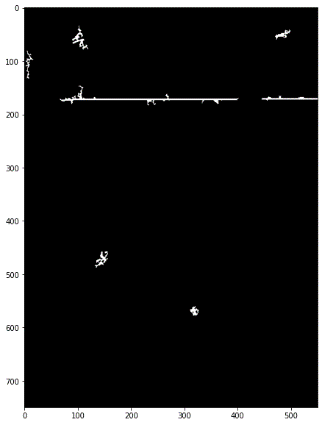


Figure 3 After Edge detection & Refinement

Figure 4 After noise cancellation

Figure 2 After pre-processing

Figure Original Image

#### Finding Defect type and highlighting it in Image:

Scratch: Highlighted in Red

Line: Highlighted in Blue

Checking Angle of defect

Checking Pixel Connectivity in defect

Imprint: Highlighted in Purple

The line defects usually have a lot more pixels connected together as compared to scratch defects. However the rare scratch can also have a lot of pixels. Along with this the line defects are almost always horizontal or vertical and not at an angle. Therefore two conditions are applied to each edge remaining in the binary mask

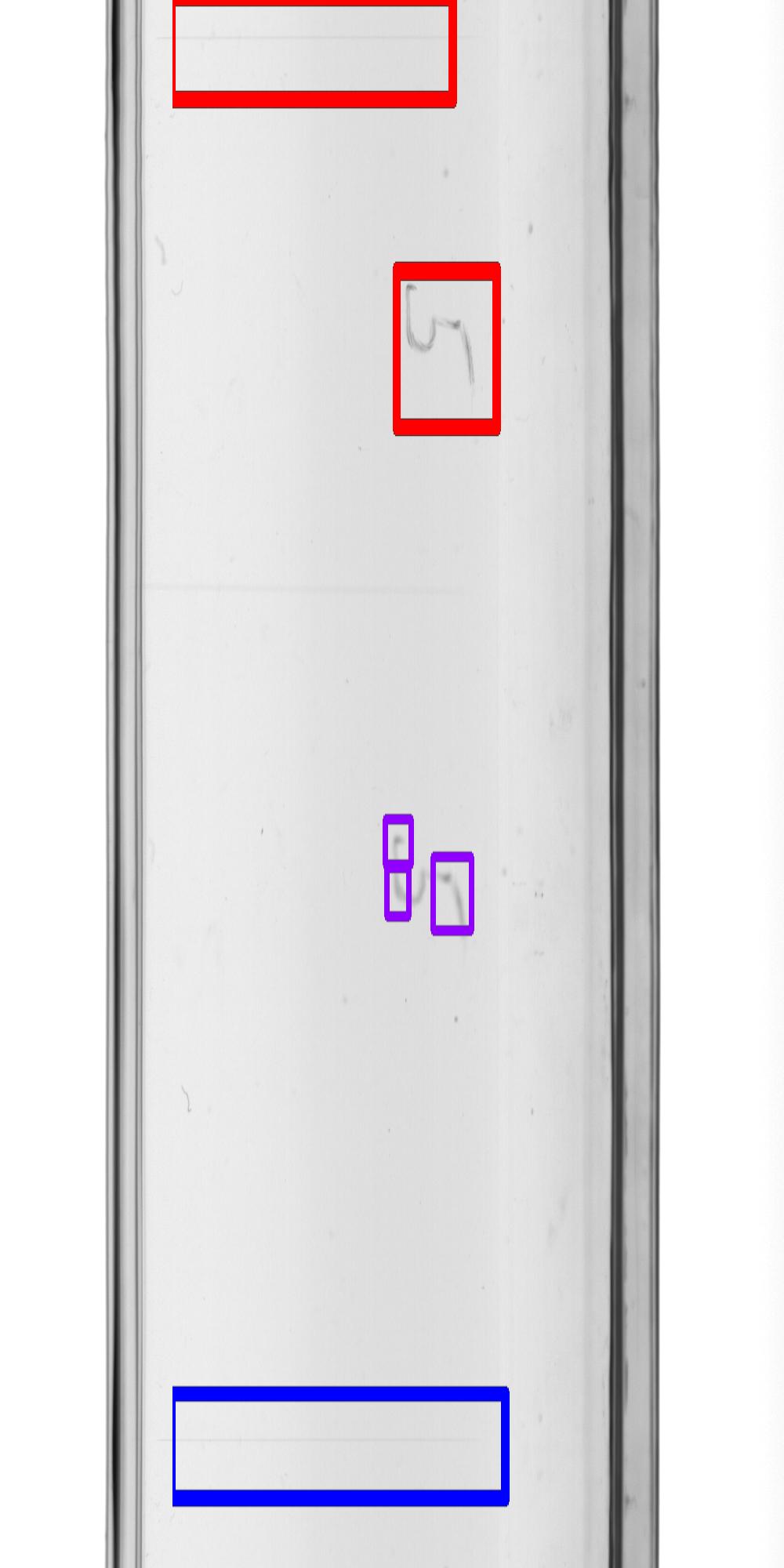
1. The number of pixels connected in a contour and the angle of the contour is calculated and if the number of pixels pass a certain threshold then
2. The angle of contour is checked and if it shows a horizontal or vertical edge, the edge is classified as a line defect, otherwise a scratch defect.

Since Imprints are detected separately from other defects there is no need for any check.

Each defect is highlighted in the original image with a bounding box, the colour of which depends on the type of defect: Red for scratch defect, Purple for Imprints and Blue for line defect.

At sensitivity = 97:

### 



#### **Method 2: SVM Solution**

We used machine learning to train 3 different sets of classifiers for different sensitivities: Low, Medium and High.  
For each of these sensitivity specific set we have 3 Classifiers:

1. Clean vs. Defected SVM classifier
2. Line vs. other defects SVM classifier
3. Scratch vs. Imprint defect Binary classifier

Pre-processing

Patch extraction (128x128)

Sensitivity: LOW, MEDIUM, HIGH

HOG features

SVM classifier 2: Line or other defects

Binary classifier: Scratch or Imprint

SVM classifier 1: Clean or Defected

Result Highlighted

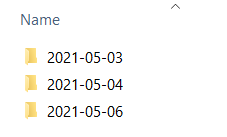
The Image obtained goes through the following steps to be correctly classified:

1. After pre-processing of the bottle image, it is divided into several 128x128 patches with 1/4 overlap between each patch.
2. A HOG descriptor of each patch is extracted and passed through the first classifier to classify between clean and defected. This classifier model is selected based on the sensitivity level that the user desires i.e. Low, Medium or High. In case of clean patch, further classification is stopped and the features of next patch are classified.
3. In case of defected classification, the patch is then passed onto the next classifier with classes: Line and other defects. The classifier is selected based on the sensitivity level again. For Line defects further classification terminated and the next patch is accessed the same way.
4. In case of other defects, the patch is passed through our binary classifier to detect either scratch or imprint.
5. If even one patch in the entire image is detected as defected, then the bottle is defected.
6. A bounding box is drawn around each defected patch with colours Blue for Line, Red for scratches and Purple for imprints for visualization.

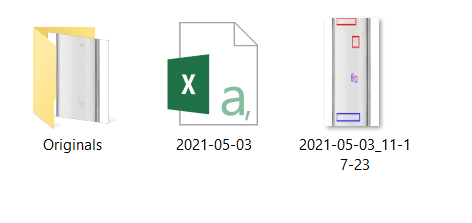
# Database Design

The AI software can be accessed through this path: Ampoule\_Final/dist/TCP\_server\_GUI.exe   
This is an executable file which will show the User Interface on start up.

The AI software also has a results directory at: Ampoule\_Final/dist/Results

This Results directory will have multiple sub-directories for the date on which the software was run and the results were stored, like this:  
  
Each date has results stored from that date so the user can easily view and move between the results.

There are three types of files in each date directory:



#### Originals

These are the original images that are passed to the AI software on that date. It keeps a track of them by storing them with time stamps of when the detection on that image started.

#### Excel

An excel file is maintained for each day. It contains the names of all the results produced along with their status (Clean or Defected) and their defect type if any. A new file is created for each new day of detection.

#### Results

The resulting images after detection, with defects highlighted in them are stored as well. They are also saved with the same time stamp as the original image had, e.g. the result image has the name “2021-05-03\_11-17-23” showing that this image was detected on 3rd May, 2021 at 11:17:23 hours.

# Flow of Source Code

### Python code:

**Class : App**

**Initialize function:**

* Sets up socket connection at local server 127.0.0.1, port 8000
* Starts listening at above port
* Sets up GUI interface
* Loads .sav weight files

**Connect function:**

* Accepts new connection
* Prints IP address of new connection
* If already connected to a client, displays an error message

**Disconnect function:**

* Checks if the program is in the middle of defect detection
* If so, displays an error message
* Otherwise closes existing connection

**Start function:**

* Checks if client is connected. If not, displays an error message
* Creates a new directory in the ‘Results’ folder and assigns paths to save images, and csv files.
* Calls **main\_func** function

**Main\_func function:**

* Calls **recvall** function to accept the length information of the incoming image
* Uses the length to then accept the coded incoming image from socket connection.
* Decodes the image into readable format
* Checks which solution is selected: **Handicraft** or **SVM Solution.**
* For SVM solution, calls **defect** function to find if the image is defected, the class of defect, and the marked image, respectively.
* For Handicraft solution, calls **defect\_bin** function to find if the image is defected, the class of defect, and the marked image, respectively.
* Displays the result on the GUI.
* Sends the result (0 for clean and 1 for defected) back to machine.
* Saves the marked image and the result in a csv file by calling **save\_img** function

**Detect function:**

* Checks sensitivity level chosen
* Breaks down the images into ROIs.
* For each ROI, calls the **classify** function to get the result and class of defect for that patch.
* If the result for any ROI is defected, classifies the bottle as defected.
* The class of defect most seen in the image patches is returned as the defect class.
* Contours are drawn around the defected patches with different colours representing different defects.
* This marked image is then displayed over GUI and returned to the main function along with the result.

**Classify function:**

* Retrieves HOG descriptor of the image patch by calling **HOG** function
* SVM classifier for the selected sensitivity levels then predicts the label for the patch
* If defected, it takes prediction from the line\_vs\_otherDefects SVM classifier to detect between line defect and scratch defect.
* If classified as otherDefects, it takes prediction from the scratch\_vs\_imprint SVM classifier.
* Returns the results.

**Detect\_bin function:**

* Checks sensitivity level
* If sensitivity level is above 15, does the following:
  + Calls **line\_scratch** function to detect line defects and scratch defects and collect marked image.
  + If sensitivity level is above 90, calls the **imprint\_detect** function to detect imprints and collect marked image.
  + The marked image (containing marked patch around the defect) is then assigned different colours based on the defect class.
  + If any line is detected in the bottle, the defect class is automatically assigned line defect (since line is the biggest defect).
  + If no line, then scratch is given priority over imprint.
  + This marked image is then displayed over GUI and returned to the main function along with the result.

**Stop function:**

* Stops accepting bottle images to detect.

### .SAV files

There are 4 .sav files present in the ‘dist’ folder. These are the different SVM classifiers for classification of bottles. These will be used by the program if the user selects SVM solution from method of classifying effects.

Three of these files correspond to SVM classifiers for different sensitivity levels, Low, Medium and High.

Line\_vs\_imp.sav is used by our program to further classify between Line and imprints.