International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 07 | July -2017 www.irjet.net p-ISSN: 2395-0072

Video and Image Processing for Finding Paint Defects using BeagleBone Black

Mr. Sohan Lokhande[1], Mr. P. T. Sasidharan[2].

[1]Student, Electronics Design and Technology, NIELIT, Aurangabad, Maharashtra, India.
[2]Professor, Dept. of Instrumentation, NIELIT, Aurangabad, Maharashtra, India.

Abstract - *Nowadays Automatic inspection plays an important role in Industrial Quality Management. This paper proposes a new computer vision system for automatic painted car body inspection using Beaglebone Black for quality control in industrial manufacturing. In most worldwide automotive industries, the inspection process is still mainly performed by human vision, and thus, is insufficient and costly. Therefore, automatic paint defect inspection is required to reduce the cost and time waste caused by defects. This new system i.e. BeagleBone Black (BBB) analyzes the images acquired from car body to detect different kinds of defects. OpenCV library is used for performing image processing operations. Initially, defects are detected using a camera connected to the beaglebone black and filtered using OpenCV libraries and then localized by using local binary pattern (LBP) and next, detected defects are classified into different defect types by using kNN classifier. A user friendly GUI is made in Qt designer where the entire video stream, captured images, defect area highlighted with name of that defect, etc will be displayed on a remote device. The results show that this method could detect defects and classify them with high accuracy*.

***Key Words*: BeagleBone Black (BBB), Defect Detection, kNN classifier, Local binary pattern (LBP), Qt.**

1. INTRODUCTION

Quality inspection is an important aspect of modern industrial manufacturing. In automotive industry, automatic paint inspection is of crucial importance to maintain the paint quality. In most worldwide automotive manufacturers the quality inspection process is still mainly performed by human vision. However, manual visual inspection is a qualitative and subjective process with often unreliable results due to its reliance on inspectors' own criteria and experience. Also, it is labor intensive and time-consuming. Therefore, automatic paint defect inspection is required to reduce the cost and time waste caused by defects. BeagleBone Black is a board specifically developed for image processing. BeagleBone Black is a minicomputer consisting of an ARM Cortex A8 processor with inbuilt 2GB eMMC, 512MB DDR3, micro HDMI, USB slot, etc. OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. This library is cross-platform and free for use under the open-source BSD license. In this paper a novel approach for automatic car body paint defect detection and classification using BeagleBone Black is presented with the use of OpenCV library. The process is organized into two stages:

1. Image capturing and Defect detection
2. Defect classification

First, a camera is connected to the beaglebone black to its USB slot for capturing video stream of the car. The video feed is given to the beaglebone black for further processing. Images are acquired from the video and then on the images filtering is done using OpenCV libraries. We have to import the OpenCV libraries into the python and change its path. Then joint distribution of local binary pattern (LBP) operators are used for detecting the defected areas and in the second stage kNN classifier are used for predicting type of the detected defect. GUI is created using Qt creator where all of the above processing of images, defected images with its classification will be displayed. Graphical view will make the system more users friendly.

2. Block Diagram

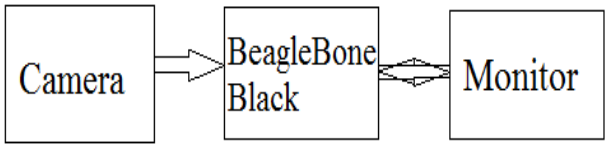

Fig-1:- Block Diagram

Above shown is a basic block diagram where a camera collects the images and gives further to the Beaglebone black for Image Processing. The beaglebone black will receive the image and first it will perform filtering and then defect detection over that image and classification of that defect and all this information will be forwarded to the monitor where a GUI will display all the information.

2.1. Camera

USB Web camera with any specifications.

2.2. BeagleBone Black

The BeagleBoard is a low-power open-source hardware single-board computer produced by Texas Instruments. BBB includes an ARM Cortex-A8 CPU, 512 MB RAM, 2 GB eMMC flash memory, etc. BBB comes with Angstrom operating system. The USB camera is connected to the BBB to the USB slot. Video stream is captured of the car through the camera and processed by BBB to find the paint defects if any. Processing involves Image capturing, defect detection, defect classification. We can change the operating system of the BBB through booting it with OS images but we have to download and install every package we require into it. The OpenCV, python, etc packages must be installed into the BBB operating system.

2.3. Monitor

The monitor will contain a Graphical User Interface (GUI) for the end user to operate and find the paint defect on the images acquired through BBB. GUI will be created using Qt creator. The GUI will consist of video feed, captured images, defected images, paint faults classification with its defect name and a database of the previous images.

3. Image Processing

Image Processing is the analysis and manipulation of a digitized image, especially in order to improve its quality. It is a processing of images using mathematical operations by using any form of the signal processing for which the input is an image, a series of images, or a video, such as a photograph or video frame; the output of image processing may be either an image or a set of characteristics or parameters related to the image.

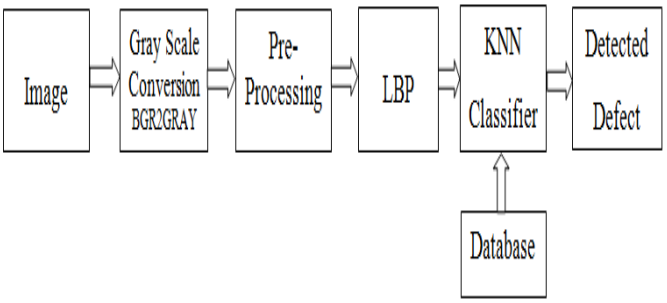

Fig-2 Image processing

The Images which are captured from the video will be provided for further processing. The process for testing the image will be the same but just a classifier will be used to detect that particular defect in that image. The classifier will compare the values of the test image with the values of the trained dataset. It will take top five matching values and it will compare its append values. The more Label values match that defect will be shown.

3.1 Image

When we click the "Start Video" button on the GUI the video will start and when we press the "Space" button on the keyboard the image frame will be captured from that video feed. The captured image will be displayed inside the "Label 6" of the GUI created in Qt designer. So again when we start the video and capture the image the latest captured image will be displayed in Label 6 and the previous captured image will be displayed inside the "Label".

3.2 Gray scale Conversion

The captured image which we be selected will be further be converted into gray scale image for. Converting the image makes it easy to find the defect areas due to uniformity in the image. In beaglebone Black as we are using the openCV library it contains a function for converting the image into gray scale. BBB takes the image into BGR format so the function for converting the image into gray scale is BGR2GRAY.

3.3 Pre-processing

Pre-processing is a common name for operations with images at the lowest level of abstraction -- both input and output are intensity images. Pre-processing is an improvement of the image data that suppresses unwanted distortions or enhances some image features important for further processing. Here we are using the GaussianBlur function provided in the openCV library for pre-processing. The Gaussian blur is a type of image-blurring filters that uses a Gaussian function (which also expresses the normal distribution in statistics) for calculating the transformation to apply to each pixel in the image.

3.4 LBP

Local binary patterns (LBP) are a type of visual descriptor used for classification in computer vision.

The LBP feature vector, in its simplest form, is created in the following manner:

- Divide the examined window into cells (e.g. 16x16 pixels for each cell).
- For each pixel in a cell, compare the pixel to each of its 8 neighbors (on its left-top, left-middle, left-bottom, right-top, etc.). Follow the pixels along a circle, i.e. clockwise or counter-clockwise.
- Where the center pixel's value is greater than the neighbor's value, write "0". Otherwise, write "1". This gives an 8-digit binary number (which is usually converted to decimal for convenience).
- Compute the histogram, over the cell, of the frequency of each "number" occurring (i.e., each combination of

which pixels are smaller and which are greater than the center). This histogram can be seen as a 256-dimensional feature vector.

- Optionally normalize the histogram.
- Concatenate (normalized) histograms of all cells. This gives a feature vector for the entire window.

In our project we are using 3x3 matrixes so it will compare the pixel to each of its 8 neighbors (on its left-top, left-middle, left-bottom, right-top, etc.)

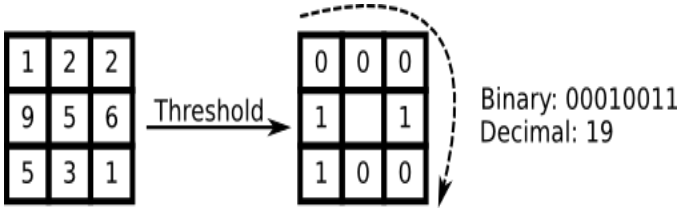

Fig 3- Thresholding using LBP matrix

The output of the LBP will be in a straight form so we have to transpose the output to make it appear in a straight form. So we have to define two variables as for **Thresholding** and for **pixel**. Thresholding will consist of simple loop that if the centre pixel value is greater than or equal to the side pixel value then just append (1) or if the value is smaller than the centre pixel then append (0).

3.5 KNN Classifier

In pattern recognition, the ***k*-nearest neighbor algorithm** (***k*-NN**) is a non-parametric method used for classification and regression. In both cases, the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression:

- In *k-NN classification*, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.

- In *k-NN regression*, the output is the property value for the object. This value is the average of the values of its *k* nearest neighbors.

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all machine learning algorithms. The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning). The distance can, in general, be any metric measure: standard Euclidean distance is the most common choice. The Euclidean distance is calculated by taking the distance of the test point from the distance of the measured point

3.6 Detected Defect

The kNN classifier will identify the defect on the basis of its algorithm the defect name will be displayed on to the GUI. Text box is present in the GUI named as Detected Defect where the defect name will be displayed and the defect will be highlighted inside a bounding box so that we can identify and find a solution to that defect.

4. GUI

Fig 4- Qt GUI

The GUI is created using PyQt which we have to install into our BBB. The above GUI has several text box and labels each have its individual significant importance for displaying the images and text. When we start the GUI press the Start Video button so it will activate the camera connected on the beaglebone black and we will get the video feed on the screen. Then we press the "Spacebar" button to capture an image frame from it. Then we can press the Testing button to start the testing of the captured image frame. The Image button on the GUI is given to browse through our previously captured images and we can select them to solidify our defect detection. The test results will be displayed inside the text box at the bottom right corner with the name of the defect displayed and detected using the design methodology. Again when we start the video and capture the image frame the

previously captured image will be replaced with the new image and the previous image will be displayed inside the label on the top right corner and this process will continue.

5. Results

The defect is detected and displayed inside the beaglebone black but we can view it using Qt Designer. The GUI results displayed are:

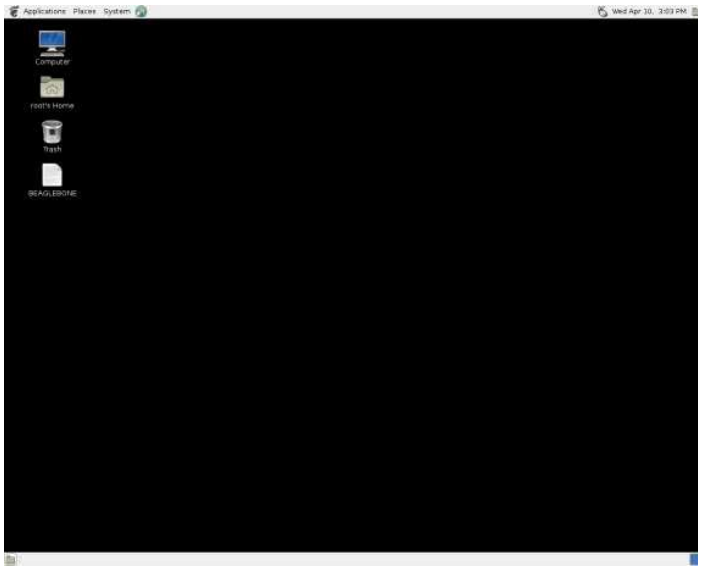

Fig 5- BBB Desktop screen

This the Desktop screen of the BeagleBone Black which will be displayed on the remote computer.

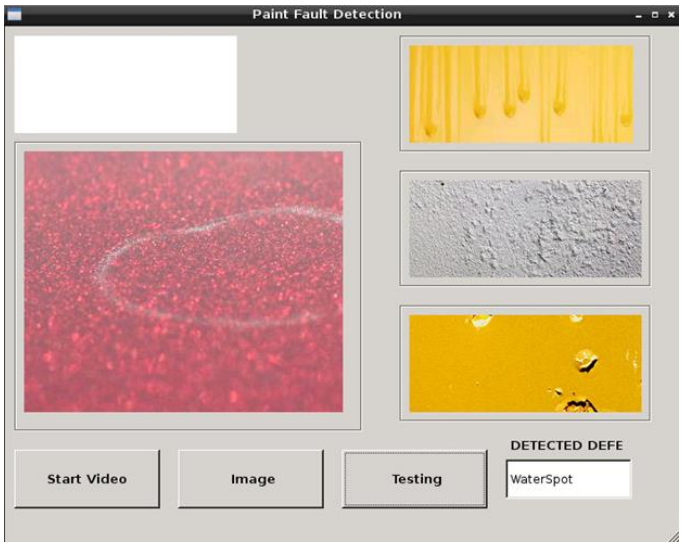

Fig 6- Waterspot defect detected and displayed

The image displayed in the big Label box is the test image on which we are applying our methodology to find the Paint defects. The paint defect is detected and displayed in the detection text box as "Waterspot"

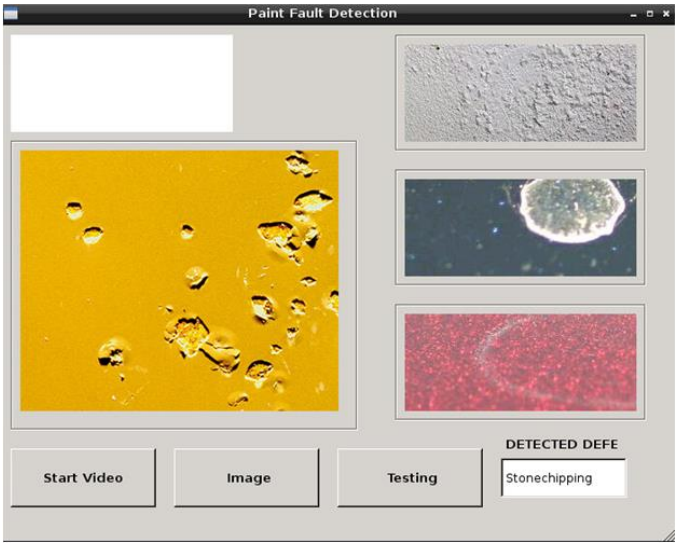

Fig 7-Stonechipping Defect detected and displayed

6. Conclusion

Two-stage method for detection and classification of painted car body defects has been presented. The proposed algorithm was effective in detecting various defects such as Stonechipping, Peeling, Waterspot and Run, etc. This method is a useful tool for automobile industries to inspect, localize and classify different defects in their products. The results presented in section 5, show that a reliable yet cost-efficient inspection of painted surfaces is attained matching the needs of industry. All of the detection is automated using the beaglebone Black. And the Graphical view makes it easy for the end user to operate and its own potential

REFERENCES

[1] Parisa Kamani, Elaheh Noursadeghi, Ahmad Afshar, Farzad Towhidkhah, "Automatic paint detection and classification"
[2] "Getting started – USB Network and Adapter" Chapter 1. Exploring Beaglebone: Tools and Techniques for Building with Embedded Linux, Wiley Publications by Derek Molloy on June 18th 2014.
[3] "Qt on Beaglebone" Chapter 6 Exploring Beaglebone: Tools and Techniques for Building with Embedded Linux, Wiley Publications by Derek Molloy on June 18th 2014.
[4] Anand Nayyar and Vikram Puri, "A Comprehensive Review of the BeagleBone Technology Smart Board Powered by ARM", International Journel of Smart Home, Vol 10, No. 4 (2016) pp. 95-108.
[5] "Programming Qt" by Matthias Kalle Dalheimer published in 1998.

[6] C. Doring, A. Eichhorn, X. Wang, and R. Kruse, "Improved Classification of Surface Defects for Quality Control of Car Body Panels, " in Proc. IEEE International Conference on Fuzzy Systems, Canada, 2006, pp. 1476-1481

[7] "Image processing and openCV" Chapter 11. Exploring Beaglebone: Tools and Techniques for Building with Embedded Linux, Wiley Publications by Derek Molloy on June 18th 2014.

[8] "Learning openCV" a book by Adrian Kaehler and Gary Rost Bradski, O'reilly publications, published on September 2008.

[9] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 2nd ed , New Jersey , Prentice Hall , 2002.

[10] G. M. A. Rahaman and M. M. Hossain, "Automatic defect detection and classification technique from image: a special case using ceramic tiles", IJCSIS, Vol. 1, No. 1, pp. 22-30, May. 2009.