An Efficient Method for Detecting Impurity from Rice Mixture Using Digital Embedded System

Bharat K. Patel[1] R. C. Patel[2]
[1,2]Instrumentation & Control deptartment
[1,2] LD College of Engineering, Ahmedabad

ABSTRACT: A faster, automatic, precise, economic and real time prototype system is developed for detecting impurity from rice using ARM controller. The system uses hue value of the color impurity to detect them real time from the mixture of rice. With the help of OpenCV (Open computer vision) library an algorithm is developed to track impurity from an embedded device to track impurity from rice mixture in real time. It is useful in quality inspection of rice grain in agriculture industry.

I. INTRODUCTION

A. *An Introduction to Digital Image Processing*

The Digital image processing technology has got great attention and considerable development in recent years. However the present image processing systems are not satisfactory with respect to real time applications and constraints. The traditional image processing techniques rely on large quantities of electronic computing devices, which results in enormous maintenance, transportation and other expenditures. These systems are based on PC and are therefore too bulky to have portability; the desk-top operating system does not have real-time character. Embedded platform is excellent due to its features of smaller size, low-cost, low power consumption and low maintenance. Therefore, developing an image processing system on the embedded platform can reduce the production and maintenance costs, improves the reliability and controllability. In this regard, this project proposes an image processing system based on embedded platform using very low cost ARM controller Raspberry Pi board (700Mhz.).

Image Processing has becomes more and more important in the areas of quality inspection of food. But most of the Image processing applications are used as a desktop application. The image processing system based on PC is too bulky to have portability; these systems will need large amount of components and size. Further the desk-top operating system does not have real-time character. The continuous development of modern industrial technology has put forward higher requirements for the acquisition, processing and transmission of image information. Embedded platform has features like smaller size, low-cost, low power consumption and low maintenance. Therefore developing an image processing system on the embedded platform can improve these systems in case of reliability and controllability. Comparing with the traditional image processing system, this system will have small size, low cost, good stability and real-time character

B. *An Introduction to Embedded systems*

Embedded system is a special computer system which centers on application, Takes computer technology as basis and adapts to the requirements of application system for function, reliability, cost, volume and power consumption and whose software and hardware can be tailored. Embedded technology has a natural connection with real-time property: an embedded system is designed for a specific purpose and it is usually restricted by space, cost, storage and bandwidth, etc., so it must be made to measure on hardware and software to the utmost extent to improve efficiency and such results finally increase its real-time property.

C. *An Introduction to quality inspection of rice*

Rice is one of the most consumed foods in the world. With the improvement of standard of living of human being, higher quality of food is required than before. The quality of rice is based on variety of properties such as texture, color, size shape, number of broken rice kernel, stones mixed, yellow particle mixed with it etc. For on line measurement a transport system is used to position the rice kernel below the camera and for sorting a discharging device is used to collect the classified kernels in different containers. Machine Vision System is relative expensive, influenced by external light conditions and needs an experienced person to setup the system. In the conventional rice sorter, image of fallen rice from a shoot is captured, and then rice color-stone is compared with some threshold value. If the rice color stone is below the threshold value the rice is rejected as damaged rice. For this conventional rice sorter, when percentage of paddy in polished rice exceeds 1%, it is difficult to exclude all paddies. Furthermore if rice flow exceeds a few thousands (kgh), the recognition percentage is below 90% and recognition ability is not always guaranteed. While small flow of rice guarantees recognition ability, this matter leads a low efficiency for rice sorting. So there is a need to develop a low cost and reliable system.

D. *An Introduction to OpenCV (Open Computer Vision)*

Originally an Intel research initiative, OpenCV is a cross-platform open source computer vision library, mostly employed for its real time image processing performance. It aims to provide well tested, optimized, open source implementations of, state of the art image processing and computer vision algorithms. OpenCV is an open source computer vision library available from http://SourceForge.net/projects/opencvlibrary. The library is written in C and C++ and runs under Linux, Windows and Mac OS X. There is active development on interfaces for Python, Ruby, Matlab, and other languages. Multiple languages bindings are available for OpenCV such as OpenCV Python has been created for Python as well as the bindings automatically built with swig. OpenCV was designed for computational efficiency and with a strong focus on real time applications. OpenCV is written in Optimized C and can take advantage of multicore processors. One of the OpenCV's goal is to provide a simple-to-use computer vision infrastructure that helps people build fairly sophisticated vision applications quickly. The OpenCV library contains over 500 functions that span many areas in vision, including factory product inspection, medical imaging, security, user interface, camera calibration, stereo vision, and robotics.

E. *Installation Procedure For OpenCV On Raspbian Operating System (Raspberry Pi BOARD)*

In the Raspberry Pi board Raspbian operating system a Linux version for Arm Controller is installing.
The following are the steps to install & make configuration for OpenCV on Raspberry Pi having Raspbian as an operating system installed on it.

STEP_1:

Installing the dependencies packages for OpenCV.

```
sudo apt-get  update
sudo apt-get install build-essential
sudo apt-get install cmake
sudo apt-get install pkg-config
sudo apt-get install libpng12-0 libpng12-dev libpng++-dev libpng3
sudo apt-get install libpnglite-dev libpngwriter0-dev libpngwriter0c2
sudo apt-get install zlib1g-dbg zlib1g zlib1g-dev
sudo apt-get install pngtools libtiff4-dev libtiff4 libtiffxx0c2 libtiff-tools
sudo apt-get install libjpeg8 libjpeg8-dev libjpeg8-dbg libjpeg-progs
sudo apt-get install ffmpeg libavcodec-dev libavcodec53 libavformat53 libavformat-dev
sudo apt-get install libgstreamer0.10-0-dbg libgstreamer0.10 libgstreamer0.10-dev
sudo apt-get install libxine1-ffmpeg libxine-dev libxine1-bin
sudo apt-get install libunicap2 libunicap2-dev
sudo apt-get install libc1394-22-dev libdc1394-22 libdc1394-utils
sudo apt-get install swig
sudo apt-get install libv41-0 libv41-dev
sudo apt-get install python-numpy
sudo apt-get install libpython2.6 python-dev python2.6-dev
sudo apt-get install libgtk2.0-dev pkg-config
```

STEP_2:

Now download the source file for OpenCV using wget

```
wget
http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.3.1/OpenCV-2.3.1a.tar.bz2
```

STEP_3:

Once finished downloading extracting the archive and then remove the archive (to save space), then change directory to the top of the source tree, now make a directory for the build and change into it.

```
tar –xvjpf OpenCV-2.3.1a.tar.bz2
rm OpenCV-2.3.1a.tar.bz2
cd OpenCV-2.3.1
mkdir build
cd build
```

STEP_4:

Now we need to make a standard configuration for building using cmake

```
cmake  -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D BUILD_PYTHON_NEW_SUPPORT=ON -D BUILD_EXAMPLES=ON ..
```

STEP_5:

In order to compile run make

```
sudo  make
```

This will take nearly 6-7 hours to compile on Raspberry Pi.

STEP_6:

And then install with make install

```
sudo make install
```

STEP_7:

We need to do some configurations for OpenCV. First open the opencv.config file with the following code:

```
sudo nano/etc/ld.so.conf.d/opencv.conf
```

Add the following line at the end of the file and then save it.

```
/usr/local/lib
```

Then edit the system-wide bashrc file:

```
sudo nano/etc/bash.bashrc
```

Add the following new lines to the end of the file:

```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig export PKG_CONFIG_PATH
```

Now everything is installed and configured and we can make use of OpenCV to build project for Image Processing in Raspberry Pi.

F. *Compiler for Raspberry Pi*

Among the different compiler available for Raspberry pi Geany editor is used widely because of its lightweight, easy to set up and compiling capability of different programming languages. In LXTerminal by typing the command

```
sudo apt-get install Geany
```

Now we can build and compile any of the vision program written in C, C++ or Python using Geany to make a real time prototype system for machine vision.

II. COLOR FORMAT IN IMAGE PROCESSING

There are many format of color are available among which RGB color format is most widely used. The RGB color format can represent any standard color or brightness using a combination of Red, Green and Blue components. For efficiency, this is typically stored as 24-bit number using 8-bits for each color component (0 to 255) so that for example, White is made of 255 Red + 255 Green + 255 Blue. This is the technique that nearly all computer screens have used for decades, and so it is the standard color format used in computer software. But unfortunately when it comes to computer vision, RGB value will vary a lot depending on strong or dim lighting conditions and shadows, etc. In comparison, HSV is much better at handling lighting differences, and it gives us an easy way to use color value. In OpenCV, H can be visualize as an angle which ranges from 0° to 180° while S ranges from 0 to 1 & V ranges from

0 to 1. Saturation is the amount of grey in color in terms of %. Value describes the brightness or intensities in terms of %. In OpenCV, the range for 'hue' is ‘0-179’, 'saturation' is ‘0-255’ and 'value' is ‘0-255’ respectively. 'Hue' represents the color, 'saturation' represents the amount to which that respective color is mixed with white and 'value' represents the amount to which that respective color is mixed with black. The 'hue' is unique for a specific color distribution of an object. But 'saturation' and 'value' may vary according to the lighting condition of that environment. Hue values of basic colors

- Orange 0-22
- Yellow 22- 38
- Green 38-75
- Blue 75-130
- Violet 130-160
- Red 160-179.

$H \in [0°, 360°]$ $S \in [0,1]$ $V \in [0,1]$

$$H = \arccos \frac{0.5(2R-G-B)}{\sqrt{(R-G)^2-(R-G)(G-B)}}$$

$$S = \frac{\max(R,G,B)-\min(R,G,B)}{\max(R,G,B)},\ S = 0 \text{ if } C=0, \text{ else } C=\frac{C}{V}$$

$$V = \max(R, G, B)$$

III. SYSTEM DESIGN SCHEME

The proposed system is composed of three parts : image acquisition module, image processing module and output module. The hardware platform used here is Raspberry Pi board having BCM2835 Arm controller which is a 32-bit processor. The image acquisition module consists of USB camera of Logitech of 0.3 Megapixels. It first captures the raw image by CMOS camera in real time and sends the acquired image via USB port in the board. Once the captured data is sent to the Arm controller it processes the image via the used algorithm and at last displays expected results on any display monitor. After processing the data any required signal can be send to the outside world. It is easy to create session of Raspberry Pi board with computer (Windows XP) using Xming and putty software and connecting the board with Ethernet cable. An algorithm made using OpenCV is successfully compiled and run in Geany editor in Python language to do require image pre and post processing.

A. *Design of hardware platform*

Block diagram of the prototype system

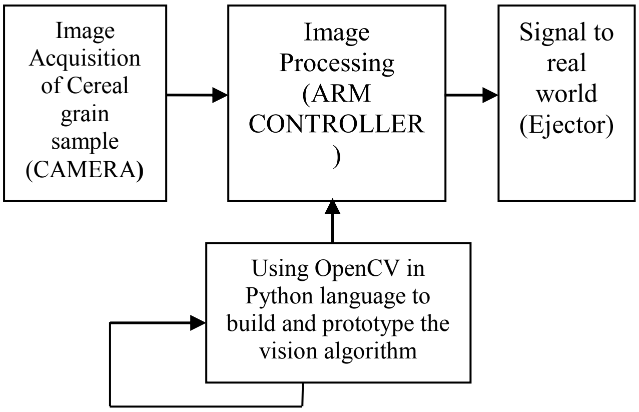
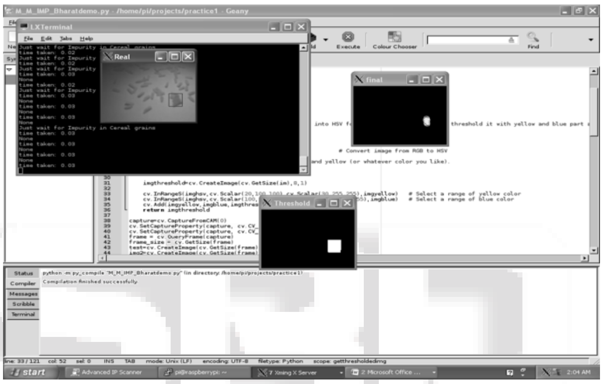


Image Acquisition of Cereal grain sample (CAMERA)
Image Processing (ARM CONTROLLER)
Signal to real world (Ejector)
Using OpenCV in Python language to build and prototype the vision algorithm

Fig. 1: prototype system

B. *Algorithm steps*

1. Capture image from camera (width 320, height 240).
2. Do Gaussian filter of the capture image to smoothen the colour image (Filtering).
3. Convert RGB to HSV (Colour conversion).
4. Threshold the hue value for required impurity to track from rice mixture (thresholding).
5. Do erosion and then dilation to remove unwanted pixel on the edges of the track object (Morphological operation).
6. Draw rectangle on the track area (Blob Analysis).
7. Show the track area using white colour of the contour found
8. Calculate the centroid of the track area in the image.
9. Store the value of the track co-ordinate in an array
10. Give the co-ordinate value of the track portion to the output pin continuously. (Signal to the real world).



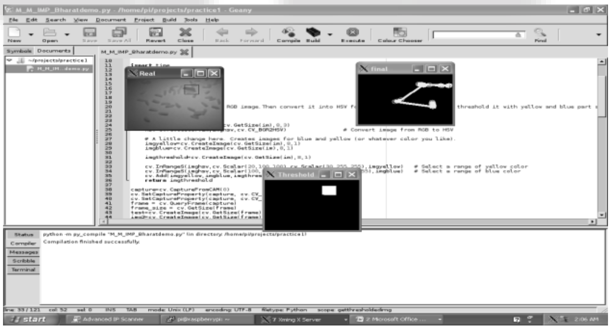
```
Just wait for Impurity in Cereal grains
time taken: 0.02
Just wait for Impurity
time taken: 0.02
Just wait for Impurity
time taken: 0.03
None
time taken: 0.02
Just wait for Impurity
time taken: 0.03
None
time taken: 0.03
None
time taken: 0.03
None
Just wait for Impurity in Cereal grains
time taken: 0.03
None
time taken: 0.03
None
time taken: 0.03
None
time taken: 0.03
```

```
    imgthreshold=cv.CreateImage(cv.GetSize(im),8,1)

    cv.InRangeS(imghsv,cv.Scalar(20,100,100),cv.Scalar(30,255,255),imgyellow)   # Select a range of yellow color
    cv.InRangeS(imghsv,cv.Scalar(100,                   55),imgblue)   # Select a range of blue color
    cv.Add(imgyellow,imgblue,imgthres
    return imgthreshold

capture=cv.CaptureFromCAM(0)
cv.SetCaptureProperty(capture, cv.CV_
cv.SetCaptureProperty(capture, cv.CV_
frame = cv.QueryFrame(capture)
frame_size = cv.GetSize(frame)
test=cv.CreateImage(cv.GetSize(frame)
img2=cv.CreateImage(cv.GetSize(frame)
```

```
python -m py_compile "M_M_IMP_Bharatdemo.py" (in directory: /home/pi/projects/practice1)
Compilation finished successfully.
```

```
line: 33 / 121   col: 52   sel: 0   INS   TAB   mode: Unix (LF)   encoding: UTF-8   filetype: Python   scope: getthresholdedimg
```

Fig. 2: Tracking of Impurity from rice mixture



```
import time

    # A little change here. Creates images for blue and yellow (or whatever color you like).
    imgyellow=cv.CreateImage(cv.GetSize(im),8,1)
    imgblue=cv.CreateImage(cv.GetSize(im),8,1)

    imgthreshold=cv.CreateImage(cv.GetSize(im),8,1)

    cv.InRangeS(imghsv,cv.Scalar(20,100,100),cv.Scalar(30,255,255),imgyellow)   # Select a range of yellow color
    cv.InRangeS(imghsv,cv.Scalar(100,                   55),imgblue)   # Select a range of blue color
    cv.Add(imgyellow,imgblue,imgthres
    return imgthreshold

capture=cv.CaptureFromCAM(0)
cv.SetCaptureProperty(capture, cv.CV_
cv.SetCaptureProperty(capture, cv.CV_
frame = cv.QueryFrame(capture)
frame_size = cv.GetSize(frame)
test=cv.CreateImage(cv.GetSize(frame)
img2=cv.CreateImage(cv.GetSize(frame)
```

```
python -m py_compile "M_M_IMP_Bharatdemo.py" (in directory: /home/pi/projects/practice1)
Compilation finished successfully.
```

```
line: 33 / 121   col: 52   sel: 0   INS   TAB   mode: Unix (LF)   encoding: UTF-8   filetype: Python   scope: getthresholdedimg
```

Fig. 3: Continuous tracking of the co-ordinate of the centroid of the impurity particle.

IV. CONCLUSION

Experiments were conducted to check the feasibility of the developed system. Mixture of rice and yellow particle were taken in random orientation and it was found that the system was able to track the co-ordinate of the yellow particle and store it in the memory and send it to the output pin of the board so that it could give signal to the ejector to reject the impurity from the rice mixture. The system is independent of the uniformity of luminous intensity of light. The program is in continuous loop and it takes time of approximately 0.03sec to complete the operation of image scanning and processing. The system can replace the conventional mechanical methods for rice sorting with high speed and reliability. We have considered a prototype system for sorting impurity from rice using an embedded raspberry pi board.

REFERENCES

[1] F. Takeda, H. Uchida, T. Tsuzuki, H. Kadota, S. Shimanouchi Kochi Univ. of Technol. ‘A proposal of grading system for fallen rice using neural network’
02/2002; DOI:10.1109/IJCNN.2002.1005560 ISBN: 0-7803-7278-6 In proceeding of: International Joint Conference on Neural Networks, 2002. IJCNN '02, Volume: 1

[2] Neelamma K. Patil and Ravi M. Yadahalli, ‘Classification of food grains using HSI color model by combining color and texture information without performing pre-processing and segmentation’, World Journal of Science and Technology 2012, 2(5):50-53 ISSN: 2231 – 2587

[3] Kavita Tated Sunil Morade, ‘Application of Image Processing for Automatic Cleaning of Rice’ Special Issue of International Journal of electronics, Communication & Soft Computing Science & Engineering, ISSN: 2277-9477

[4] Liu Guang-rong, “Rice Color Inspection Based on Image Processing Technique", International Conference on Advances in Energy Engineering, 2010.

[5] Bhupinder Verma,“Image Processing Techniques for Grading & Classification of Rice", International conference on computer & communication technology, ICCC 2010.

[6] learn.adafruit.com/category/learn-raspberry-pi

[7] http://sourceforge.net/projects/opencvlibrary/files/ opencv-unix/2.3.1/OpenCV-2.3.1a.tar.bz2

[8] http://SourceForge.net/projects/opencvlibrary.

[9] http://docs.opencv.org