Implementation and Optimization of Core Computer Vision Function in Raspberry Pi

Nirmal T M
M.Tech Scholar
Dept. of ECE
Sahrdaya College of Engineering & Technology, Kodakara, Kerala
nirmaltm@ieee.org

Rajeev K
Principal Research Engineer
Technology Specialists Group
QuEST Global, Technopark, Trivandrum, Kerala
rajeev.k@quest-global.com

K R Joy
Head of Department
Department of ECE
Sahrdaya College of Engineering & Technology, Kodakara, Kerala
joykarotte@gmail.com

***Abstract*— Computer vision is the technology that for acquiring, processing, analyzing and understanding high-dimensional data images in order to produce numerical or symbolic information. Nowadays computer vision concepts are being pervasively applied in all fields of life, but due to the requirements of high processing power and high amount of parallelism, real-time computer vision applications are restricted to field-programmable gate array and high end processing graphics processing unit. The Raspberry Pi is an easily available, low cost and medium processing power embedded development board. Raspberry's capability in performing computer vision functions has already been demonstrated with OpenCV support. The optimization of OpenCV functions in Raspberry Pi is still considered a challenge. This work aims to implement and optimize core computer vision algorithms on Raspberry Pi. Apart from algorithm level adaptations for better hardware utilization, the further optimization is done by kernel recompilation of Raspberry Pi has also been considered. The counting of objects from an image has a great application in the present market. This paper deals with finding the number of object from an image in an optimized manner.**

***Index Terms*—Connected Component Labeling, OpenCV, Raspberry Pi, ARM, Visual Studio, Rosenfeld, Contour labelling.**

I. INTRODUCTION

The design use and advancement of Real Time computer vision algorithms are developing rapidly in the present scenario. Most of the current real time image processing algorithms requires resources of a high end desktop computer to process, And may not capable to keep up with the increased scanning speed in case of real time applications. The embedded real time platform is a good solution for real time computer vision. But due to the lack of processing power and critical memory, the advancement in real time computer vision on embedded are challenging. The applications of embedded boards are robotic, military, remote sensing, and vehicle automation etc… for all these application the computer vision is performed in a high performance graphical board or FPGA boards. The GPU supports of embedded boards are not used for this type of applications. The adaption of real time computer vision to these embedded boards in an efficient manner it will be very effective and low cost for many of the real time applications. It is very easy to reconfigure and remote reconfiguration can also possible. Many of the external interfaces can be easily done with embedded boards.

The computer vision results are in the forms of decisions and data that human can understand. The main idea behind this computer vision application is to explode the human vision by the utilization of computer resources. This electronic vision help us to get data from an un-human environment also more precise and faster that can be used decision making with the help of electronic devices. The image analysis includes the geometric, structural, physics and color analysis and the resulting data or the decision can be used for automation. The automation having a lot of application in industrial and unmanned vehicles etc…

The embedded board can be described as a single-board computer (SBC) is a complete computer built on a single circuit board, with microprocessor(s), memory, input/output (I/O) and other features required for a computer. It described as a system On Chip (SOC), means the whole system is enclosed in a chip, that having memory, processor and I/O controller circuits. Single-board computers were made as demonstration or development systems, for educational systems, or for use as embedded computer controllers.

The paper aims to implement and optimize computer vision on an embedded platform. The level of optimization can be done in the case of number of frames per second improvement and processor utilization. The selection of embedded board for this process requires lot of comparisons. The selection was based up on the relevance of that board in the present market and resource specification. From that the work select Raspberry Pi as the embedded platform for the implementation and optimization of computer vision. The raspberry Pi board having very limited resources, so we need to obtain maximum possible real-time result in a limited environment.

For this an object counting application is taken. This is having huge application in medical as well as industrial. In medical domain it can be used for detection of tissues or samples or cancer particles form our body part also the result will shows the density of that. So the doctor need not take so much time for calculation of the decisis. Also in industrial domain it can be used for counting the product from the production belt. To count the number of vehicle etc… can be done with the help of this.

The image processing algorithm can be taken as Connected component Labelling. Which is the basic step used for all the computer vison application such as feature extraction, object detection, face detection etc… the CCL (Connected component Labelling) can be implement in different algorithms in a different manner. One pass based or two pass based and contour based. Each method is having its own advantages and disadvantages.

The paper discusses two approaches one is OpenCV basis and the other one without using the OpenCV. Although the OpenCV is an optimized library for the computer vision it has having limitations that the OpenCV need some high memory for loading the complete header libraries even though we are not using whole functions. But the inbuilt functions in OpenCV are optimized better than the normal written code. The object counting code is not present in the OpenCV. In OpenCV the CCL is done by contour based method and it is more faster one. The limitations are this can't able to give the number of objects. The normal coding approach is described more in this paper and we focused on implementation of CCL with Rosenfeld algorithm.

The remaining section of this paper is arranged in this manner, second section contains literature survey based on computer vision application implemented on the Raspberry Pi board. The third section is the brief idea regarding the OpenCV based Contour labelling. The forth section is the description of Rosenfeld algorithm. And the fifth part is the simulation result and implementation. Sixth section includes the results and future works.

II. Literature Review

A. Related Works Done on Raspberry Pi

The real time embedded processing boards are very common in the industrial and military application. But due to fewer constraints the complex computer vision is not usually implemented on this type of boards. There is a large amount of work done in the area of raspberry pi robotic applicator. Referring to computer vision field the number of works are less. The raspberry Pi is enriched with the support of OpenCV and OpenGL.

The Raspberry PI Based Stereo Vision for Small Size ASVs[1] is among one. In this work an approach to stereovision applied to small water vehicles. By using PC and inexpensive off-the-shelf components, we were able to develop an autonomous driving system capable of following other vehicle and moving along paths delimited by colored marks. A pair of webcams was used and, with an ultrasound sensor, that is able to implement a basic frontal obstacle avoidance system. With the help of the stereoscopic system, we inferred the position of specific objects that serve as references to the ASV guidance. The final system is capable of identifying and following targets in a distance of over 5 meters. As vehicles tend to accomplish larger and more complex missions, energetic autonomy poses a problem to the use of powerful computational systems; on the other hand, the cost of special-purpose hardware, though having dropped over the years, is still a limitation to the dissemination of robotic applications. Recently a range of ARM architecture computational devices such as the Raspberry PI or the even more powerful Quad-Core ODROID-U2, devices under USD 90, allowing the off-the-shelf robotics era to begin. Systems like the one described in use computer vision to detect the horizon line and specific objects in the scene as an aid to small sailboat guidance. Others use visual information to classify terrain and distinguish the water areas. Some applications have also been developed that are capable of avoiding obstacles in specific water and scenery conditions using stereovision

The study of how to effectively capture the image and video can capture using Raspberry Pi is described in embedded image capturing system using raspberry pi system [2]. An image capture system with embedded computing can extract information from images without an external processing unit, and interface devices used to make results available to other devices. The choosing of an embedded platform is very unique. The paper proposes an image capturing technique in an embedded system based on Raspberry Pi board. Considering the requirements of image capturing and recognition algorithm, Raspberry Pi processing module and its peripherals, implementing based on this platform, finally actualized Embedded Image Capturing using Raspberry Pi system (EICSRS). Experimental results show that the designed system is fast enough to run the image capturing, recognition algorithm, and the data stream can flow smoothly between the camera and the Raspberry Pi board. The implementation lead to effective utilization of Raspberry pi camera and an efficient image capturing system.

The raspberry Pi based color identification is done in Raspberry Pi Based Color Speaker [3]. The work lead to the design and implementation of the image processing based color speaking system using Raspberry Pi and USB Camera. This design is a minimized electronic gadget which identifies the color of an object image and speaks out the corresponding color. The model uses hardware components such as Raspberry Pi (Model B) and USB webcam. Matlab Simulink tool boxes were used to implement the project. The proposed gadget works in standalone mode without the necessity of PC once programmed. We used rapid prototype technique approach of image processing for real-time application using Matlab Simulink support package meant for Raspberry PI. The real time color identification is done using the help of thresholding and identification of major color component in an image using OpenCV support.

Many of the OpenCV functions are explained in Real-Time Carbonization Using Raspberry Pi [4]. Carbonization is a method of image stylization in which input moreover looks like a sketch or cartoon like image. It includes edges bolder, colors brighter and lively, accompanied with smoothening operation so as to filter out the high frequency details. Furthermore, reduction in the luminance contents is carried out, these are used to quantize the image for obtaining cartoon like effect. Different image processing techniques such as contrast stretching, bilateral filtering, luminance quantization, and edge detection are being implemented to achieve the desired outcome. The work have proposed and implemented a device

using Raspberry pi that could create carbonized images on the fly from a camera feed. OpenCV programming is used for implementing image processing techniques. The OpenCV real time functions are used for getting this cartoonized image.

The above mentioned are the main works done in the Raspberry Pi on computer vision application. The optimization of the resource utilization is not yet done in an efficient manner. The optimization of application working on Raspberry Pi is not performed. Some of the optimization work done in similar platforms can be discussed in the preceding section.

"The optimization of face detection in an embedded processor is described as Implementation and Optimization of Embedded Face Detection System" [5]. To attain real time performance particularly on mobile device platforms it is necessary to apply optimized algorithms. This paper presents work on performance optimization of general computer vision algorithms such as Viola Jones Face Detection on embedded systems with limited resources. The Viola Jones algorithm which is popular for face detection can be implemented on mobile platforms. The algorithms are benchmarked on the Intel processor and Beagle Board xM, which is a new low-cost low-power platform, based on the Texas Instruments (TI) DM 3730 processor architecture. The DM 3730 processor is characterized by the presence of an asymmetric dual-core architecture that include an ARM and a DSP along with a shared memory between them. OpenCV, which is a open source computer vision library developed by Intel corporation was utilized for some of the algorithms. Comparing the results for the different platforms are introduced and analyzed with an emphasis on real-time Application.

The optimization of computer vision can be done in such a way that Optimization of Computer Vision Algorithms for Real Time Platforms [6]. Real time computer vision applications like video streaming on cell phones, remote surveillance and virtual reality have high performance requirements but can be severely restrained by limited resources. The use of optimized algorithms is vital to meet real-time requirements especially on popular mobile platforms. This paper presents work on performance optimization of common computer vision algorithms such as correlation on such embedded systems. The correlation algorithm for face recognition can be implemented using convolution or the Discrete Fourier Transform (DFT). The algorithms are benchmarked on the Intel Pentium processor and Beagle board, which is a new low-cost low power platform based on the Texas Instruments (TI) OMAP 3530 processor architecture. The OMAP processor consists of an asymmetric dual-core architecture, including an ARM and a DSP supported by shared memory. OpenCV, which is a computer vision library developed by Intel corporation was utilized for some of the algorithms. Comparative results for the various approaches are presented and discussed with an emphasis on real-time implementation. These are the current works done in the area of optimization of computer vision.

B. About Raspberry Pi

The Raspberry Pi is a miniature single-board computer developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools. Raspberry Pi Model B into module for use as a part of embedded systems, to encourage their use [7]. The Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor, VideoCore IV GPU, formally it developed with 256 megabytes of RAM, later upgraded (Model B & Model B+) to 512 MB. The system has Secure Digital (SD) or MicroSD (Model A+ and B+) sockets for boot media and persistent storage. The figure 3.1 shows the figure of Raspberry Pi B+ board.

The Foundation provides Debian and Arch Linux ARM distributions for download. Tools are available for Python as the main programming language, with support for BBC BASIC (via the RISC OS image or the Brandy Basic clone for Linux), C, C++, Java, Perl and Ruby.

The Processor of a Raspberry Pi B+ board consist of level 2 cache is 128 KB, used primarily by the GPU, not the CPU. The Broadcom SoC used in the Raspberry Pi is equivalent to a chip used in an old smartphone (Android or iPhone). While operating at 700 MHz by default, the Raspberry Pi provides a real world performance roughly equivalent to 0.041 GFLOPS. On the CPU level the performance is similar to a 300 MHz Pentium II of 1997-1999. The GPU provides 1 Gpixel/s or 1.5 Gtexel/s of graphics processing or 24 GFLOPS of general purpose computing performance. The graphics capabilities of the Raspberry Pi are roughly equivalent to the level of performance of the Xbox of 2001. The Raspberry Pi chip, operating at 700 MHz by default, will not become hot enough to need a heatsink or special cooling. The SoC is stacked underneath the RAM chip, so only its edge is visible.

Most Raspberry Pi devices can be overclocked to 800 MHz and some even higher to 1000 MHz. In the Raspbian Linux distro the overclocking options on boot can be done by a software command running "sudo raspi-config" without voiding the warranty, see note 9 below. In those cases the Pi automatically shuts the overclocking down in case the chip reaches 85 °C (185 °F), but it is possible to overrule automatic over voltage and overclocking settings (voiding the warranty). In that case, one can try putting an appropriately sized heatsink on it to keep the chip from heating up far above 85 °C. In our development we are using the Raspberry Pi B+ model with Raspberrian (Debian flowered) as an OS for the system

C. Why Raspberry Pi

For the implementation of computer vision we gone through different real time embedded boards and come up with some conclusions such that the present world is focussing on a low cost application oriented embedded boards and in that feels the major embedded platforms are Raspberry Pi by Raspberry foundation, Beagle Bone from the family of Texas instruments and Panda Board. The comparison is listed in Table I.

The TI's Beagle bone black have More Fast Processor, But lack of Video decoding and encoding capabilities, it have not a Dedicated camera Interface, 3D graphics accelerator (OpenGL® ES 1.1 or 2.0 for 3D graphics and OpenVG™ for 2D scalable vector graphics), NEON floating-point accelerator,

2x PRU 32-bit microcontrollers, and Simply We can say that a Light version of Panda Board

TABLE I. COMPARISON OF EMBEDDED BOARDS

	BeagleBone Black [Rs : 4300/-]	**Raspberry Pi B+ (New June 2014)** [Rs : 2800/-]	**PandaBoard** [Rs : 18,000/-]
Processor	AM335x 1GHz ARM® Cortex-A8 with 2000 MIPS	ARM1176JZF-S core (ARM11 family, ARMv 6 instruction set)	ARM Cortex-A9 MPCore
Clock Speed	1GHz	700MHz (Most Raspberry Pi devices can be overclocked to 800 MHz and some even higher to 1000 MHz)	1 GHz (PandaBoard) 1.2 GHz (PandaBoard ES)
RAM	512 DDR3L @ 400 Mhz	512 MB SDRAM @ 400 MHz (256 shared with GPU)	1 GB low power DDR2 RAM
On Board Memory	4GB 8-bit eMMC on-board flash storage (eMMC so it's performance is same as external SD card)	-NA- We can add SD card, desirably class 10	-NA- We can add SD card, desirably class 10
GPU	PowerVR SGX530 @ 200 MHz (Little Old also not as good as VideoCore IV)	Broadcom VideoCore IV @ 700 MHz [Broadcom BCM2835 SoC full HD multimedia applications processor]	SGX540 graphics processor @ 200MHz
Video Decoder	-NA-	yes	yes
Ethernet	10/100M (Supported by SoC) So Better Performance	10/100M (USB to Ethernet converter) So average Performance	10/100M On board
USB	1 client / 1 host port	4 Host	2x USB 2.0 (Expansion)
AV	micro HDMI	HDMI & Composite	HDMI A Type
GPIO	65 Pins	40 pins	Expansion Connector Can be used
Power	USB 5V DC Jack 210 to 460mA @ 5V	600mA up to 1 A @ 5V	500mmA-1A @ 5V

*Made by comparing the official website of these board

The panda board have a Dedicated Camera connector As like RBPi, GPU supports OpenGL ES 2.0, OpenGL ES 1.1, OpenVG 1.1 and EGL 1.3, and anOnboard Wi-Fi & Bluetooth connectivity.

The Raspberry Pi have the advantages of More used in Robotics & Robot Vision Application, 15-pin MIPI camera interface (CSI) connector, 3D support with Pi3D - OpenGL ES 2.0 (24 GFLOPS), and High Community support and Ease of use.

From the above comparison we can say that the Raspberry Pi B+ is more useful for our work. The clock speed of Raspberry Pi B+ is little less, but compared to other it's negligible and if necessary we can over clock the device.

The GPU is better for Raspberry Pi B+ and it works high clock speed. The inbuilt memory is not available but it can overcome using a micro SD card. For image processing application we can prefer a class 10 SD card. Raspberry Pi B+ contains a video core so it can able to process real-time image processing. The Ethernet is little slower but it will only effect when we use a multiple network cameras GPIO can easily usable and user friendly. The power conception is little higher but it can negligible. Coming to cost of the product the effective cost of the board is very less and affordable for common persons. So as a conclusion we can say that the Raspberry Pi B+ can be taken as a final board for our real-time image algorithm application.

III. OPENCV BASED CONTOUR LABELLING

The OpenCV having an in built function for Contour based labelling, when we give an input image to that function and the threshold limit according to that the image is threshold and the contour is detected and labelled [8]. In this manner this algorithm is works the result is shown in Fig. 3.1. The various color represent for the different object detected using the function. In this method the number of labels will not get, also the pixel wise scanning is not performing, the detection of smaller particles sometime it may take as a salt and pepper noise so it can't able to get that.so this is only used for an shot team application which need lesser precision and for some experimental cases.

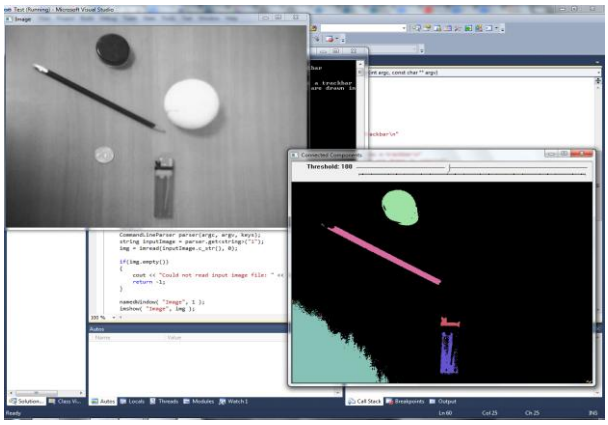


Figure 3.1: Contour based Labelling

IV. ROSENFELD CONNECTED COMPONENT LABELLING

The Rosenfeld is a historical based approach algorithms. It consists of two pass for making the final labelled image. The first scan create the pre labelled image which consist of one left and three upper connected pixel labels also it create the equivalence table during the first scan. And the next step is to resolve the equivalence table [9]. After resolving the equivalence table the result will be an idea regarding the number of labels in the image. This is same as the number of component or element in the image. All these operations are performed on the threshold binary image. The second pass is performed on the labelled image, during the second scan the

labelled image entry is replaced according to the equivalence table entry.

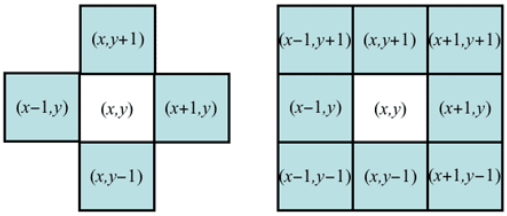


Figure 4.1: (a) 4-Connected (b) 8-Connected Matrix

The result of this algorithm is 8-connected component labelled image as well as the number of element in that image. The sample 4 and 8-connected matrix is shown in Fig-4.1. The parameters and the Rosenfeld algorithm are as follows [10]. Before performing this algorithm the image is padded with zeros in four edge to avoid the programming complexity.

- p_x, e_x, the current pixel and its label
- p_1, p_2, p_3, p_4, the neighbor pixels
- e_1, e_2, e_3, e_4, the associated labels
- n_e, the number of labels
- T, the equivalence table
- a, the ancestor (the primal equivalence label) of e: a = T[e]
- n_a, the number of labels (ancestors) after packing T
- *run*: a set of pixels, on a line, with same label.

A. Algorithm 1: Find algorithm

Input: e a label, T an equivalence table
Result: a, the ancestor of e

1. $a = e$
2. while T[a] = a do
3. a = T[a]
4. return a

B. Algorithm 2: Union algorithm

Input: e_1, e_2 two labels, T an equivalence table
Result: T an equivalence table is edited with least common ancestor a.

1. a_1 find(e1; T)
2. a_2 find(e2; T)
3. if $a_1 < a_2$ then
4. T[a_2] = a_1
5. else
6. T[a_1] = a_2

C. Algorithm 3: positive min value min+

Input: e_1; e_2; e_3; e_4 four labels with at least one non-zero label
Result: m, the smallest non zero value

1. m = INT_MAX
2. for each $e_k \in \{e_1; e_2; e_3; e_4\}$ do step 3
3. if ($e_k \neq 0$ and $e_k < m$) then $m = e_k$
4. return m

D. Algorithm 4: First scan of Rosenfeld's algorithm

Input: e_1; e_2; e_3; e_4, four labels

1. For each *pixel* p_x do step 2
2. if $p_x \neq 0$ then do step 3 else step 14
3. if ($e_1 = e_2 = e_3 = e_4 = 0$) then step 4 else step 7
4. $n_e = n_e + 1$
5. $e_x = n_e$
6. T[n_e] = n_e
7. For each $e_k \in \{e_1; e_2; e_3; e_4\}$ do step 8
8. a_k = Find(e_k,T)
9. $e_x = min+ (a_1; a_2; a_3; a_4)$
10. for each $e_k \in \{e_1; e_2; e_3; e_4\}$ do step 11
11. if ($a_k \neq 0$ and $a_k \neq e_x$) then step 12 else step 14
12. *Union*(e_x; a_k; T)
13. $e_x = a_k$
14. $e_x = 0$

E. Algorithm 5: Equivalence resolution & pack

1. for $e \in [1 : n_e]$ do
2. if T[e] $\neq e$ then
3. T[e] = T[T[e]]
4. else
5. $n_a = n_a + 1$
6. T[e] = n_a

F. Algorithm 6: Second scan of Rosenfeld's algorithm

Input: E, image of labels

1. For each label $e_x \in$ E do
2. e_x = T[e_x]

the above mentioned algorithm 1-6 is the Rosenfeld algorithm used for finding the CCL of an image[11]. For our application we need to perform up to Algorithm 5, there is no need for the second scan. The second scan is used only for displaying the final component detected label image. From algorithm-5 itself we can be able to get the number of components. This requires only the limited amount of memory and less number of calculations. Only the image storage and the label image is need more memory other equivalence table and loop variables need very few amount of storage space.

V. Simulation and Implementation

The code is written in C++ and for reading and displaying the image we interface the OpenCV functions. The simulation is done in Visual Studio 2010. After simulation the code is ported to the Raspberry Pi board. The GCC compiler in the Raspberry pi is used for running the code. The simulation results are shown in bellow. The Fig 5.1 is the input image of coins and it was threshold with the threshold limit of 90. The threshold image is shown in Fig 5.2.

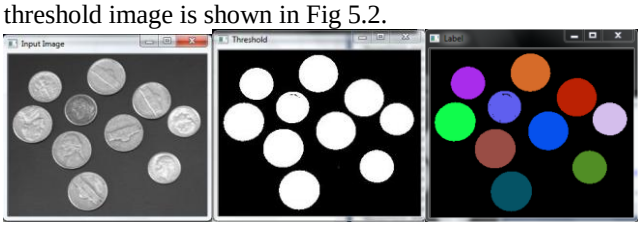
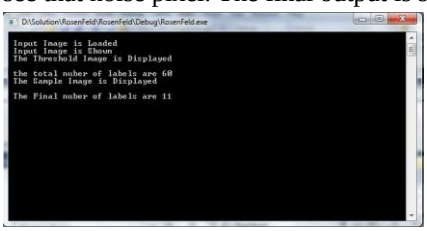


Fig 5.1 Input Image Fig 5.2 Threshold Image Fig 5.3 Label Image

After first scan we will get a pre labelled image that is displayed in Figure 5.3. The color is given is for just an illustration purpose. In time the total number of labels is 60. And after the equivalence resolving we will be getting it as 11 (10 Coin image + one noise dot) that noise dot can be avoided by proper fixing of threshold limit. In threshold image we can see that noise pixel. The final output is shown in Fig 5.4.



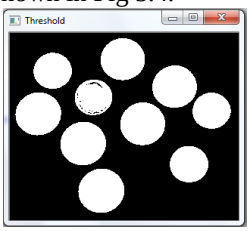


Figure 5.4 Output of threshold 90 Fig 5.5 Threshold 95 image

The dot can be avoided by proper thresholding, the new thresholding limit is changed to 95 and the result is shown in the Fig 5.5 in this case the first pass will produce the total number of component as 57, and in second pass we will get the actual result, that is 10 components (coins). The result is shown in Fig 5.6

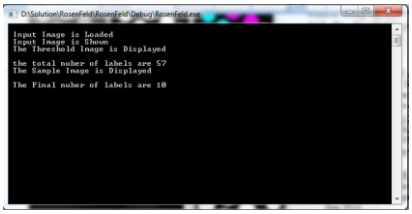


Figure 5.6 Final Result of Detected coins

VI. Result and Future Work

The simulation and implementations done in 246x300 image of coins the threshold need to be more improved to avoid the unwanted dots. This algorithm is almost faster to capable the real time situations. For higher performance and memory optimization we need to use the modern segment based algorithm. The Rosenfeld is a pixel based so it contains lot of conditional statements. But in the segment based algorithm this will have lesser. The main problems with the segment based algorithm are it requires more processing power and memory. The Light speed algorithm is an optimized modern segment based algorithm. This can be used for implementation in Raspberry Pi, with lesser number of conditional statements.

VII. Conclusion

The Connected Component labeling, a core computer vision that is used in many of the image processing application is implemented using the historical pixel based Rosenfeld algorithm and the result are verified using a 246 x 300 image. And after zero padding the image size will become 248x302 and performed the operation. The Rosenfeld Connected Component labelling is completely written in C++ coding without using OpenCV so the memory optimization is done. And the Raspberry pi GCC compiler is used for the implementation.

References

[1] Ricardo Neves and Anibal C. Matos, "Raspberry PI Based Stereo Vision For Small Size ASVs" Proceedings of 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies (ICSCCN 2011)

[2] G.Senthilkumar, K.Gopalakrishnan , V. Sathish Kumar "EMBEDDED IMAGE CAPTURING SYSTEM USING RASPBERRY PI SYSTEM" International Journal of Emerging Trends & Technology in Computer Science, Volume 3, Issue 2, March – April 2014

[3] Dr.S.A.K.Jilani, G.R.S.Manasa "Raspberry Pi Based Color Speaker" SSRG International Journal of Electronics and Communication Engineering (SSRG-IJECE) – volume1 issue7 Sep 2014

[4] Parimal Sikchi, Neha Beknalkar, Swapnil Rane 'Real-Time Cartoonization Using Raspberry Pi' IJCAT International Journal of Computing and Technology, Volume 1, Issue 6, July 2014

[5] Aby P.K, Anumol Jose, Bibin Jose, Dinu L.D, Jomon John, Sabarinath G "Implementation and Optimization of Embedded Face Detection System" Proceedings of 2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies (ICSCCN 2011)

[6] Pramod Poudel and Mukul Shirvaikar "Optimization of Computer Vision Algorithms for Real Time Platforms" 42nd South Eastern Symposium on System Theory University of Texas at Tyler Tyler, TX, USA, March 7-9, 2010

[7] "The Raspberry Pi single-board computer will revolutionise computer science teaching [For & Against]" Engineering & Technology (Volume:7 , Issue: 3), 26

[8] Giuseppe Papari, Nicolai Petkov 'Edge and line oriented contour detection: State of the art' ScienceDirect, Image and Vision Computing 29 (2011) 79–103

[9] Akmal Rakhmadi, Nur Zuraifah Syazrah Othman, Abdullah Bade 'Connected Component Labeling Using Components Neighbors-Scan Labeling Approach' Journal of Computer Science 6 (10): 1096-1104, 2010

[10] Luigi Di Stefano, Andrea Bulgarelli 'A Simple and Efficient Connected Components Labeling Algorithm' Image Analysis and Processing, 1999. Proceedings. International Conference on Image Analysis and Processing, 322 – 327, 1999

[11] Lionel Lacassagne, Bertrand Zavidovique 'Light Speed Labeling Efficient Connected Component Labeling on RISC Architectures' Image Processing (ICIP), 2009 16th IEEE International Conference on Image Processing, 3245 – 3248, Nov. 2009.