

# **MACHINE VISION BASED BACTERIA-COLONY** **COUNTER**

*Thesis submitted in partial fulfillment of the requirement for the award of  
degree of*

**Master of Engineering  
In  
Electronic Instrumentation and Control**



By:  
**Monita Goyal**  
**(80651012)**


Under the supervision of:  
**Mandeep Singh, Assistant Professor**

ELECTRICAL AND INSTRUMENTATION DEPARTMENT  
THAPAR UNIVERSITY  
PATIALA – 147004  
**JUNE 2008**


## Declaration


The proposed method relates to a colony counter for bacterial specimens on a petri dish. I hereby declare that the report entitled "Machine Vision based Bacteria Colony Counter" is an authentic record of my own work carried out as requirement for the award of degree of M.E. (Electronic Instrumentation & Control) at Thapar University, Patiala, under the guidance of Mr. Mandeep Singh, Assistant Professor, during January to June 2008.

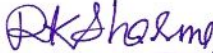
Date: July 4, 2008

  
-----  
(MONITA GOYAL)  
Roll No. 80651012

It is certified that the above statement made by the student is correct to the best of our knowledge and belief.

  
-----  
(Mr. Mandeep Singh)  
Assistant Professor, EIED  
Thapar University, Patiala

  
-----  
(Dr. Smarajit Ghosh)  
Professor & Head, EIED  
Thapar University, Patiala

  
-----  
(Dr. R K Sharma)  
Dean of Academic Affairs  
Thapar University, Patiala

## Acknowledgment

---

The real spirit of achieving a goal is through the way of excellence and austere discipline. I would have never succeeded in completing my task without the cooperation, encouragement and help provided to me by various personalities.

With deep sense of gratitude I express my sincere thanks to my esteemed and worthy supervisor, Mr. Mandeep Singh, Assistant Professor, Department of Electrical & Instrumentation Engineering, Thapar University, Patiala, for his valuable guidance in carrying out this work under his effective supervision, encouragement, enlightenment and cooperation.

I shall be failing in my duties if I do not express my deep sense of gratitude towards Dr. Smarajit Ghosh, Professor and Head of the Department of Electrical & Instrumentation Engineering, Thapar University, Patiala, who has been a constant source of inspiration for me throughout this work.

I am also thankful to all the staff members of the Department for their full co-operation and help.

My greatest thanks to all who wished me success especially my parents. Above all I render my gratitude to the ALMIGHTY who bestowed self-confidence, ability and strength in me to complete this work.

Place: Thapar University, Patiala

Date: July 4, 2008

  
Momita Goyal

## Abstract

---

Machine Vision is an emerging area related to real-time capturing, processing, and analyzing the images for various kinds of scientific and industrial applications. Bacteria counting are required in number of applications in the fields such as Biotechnology, Pathology etc. Manual counting of large number of Bacteria in any of these applications can be a tedious and time consuming process, prone to human errors. This can be automated using Machine Vision concepts.

In my thesis, I propose a technique of “Machine Vision based Bacteria Colony Counter” which is based on particle analysis approach that enables to count bacterial colony on a Petri dish. Machine vision based bacteria colony counter allows each plate to be counted with equal efficiency. In the present work the image is captured by IEEE-1394 Digital Camera Prosilica 2.0.1 and processed by vision workstation Compact Vision System-1450. Configuring the two and optimizing their performance for the above application is one of the major components of this work.

## Organization of Thesis

---

The first chapter briefly introduces bacteria and colony counter. The second chapter provides an overview of vision basic concepts which provide the anatomy of Machine Vision. The third chapter describes the requirements to set up the development computer for image acquisition using IEEE 1394 digital camera. The fourth chapter gives an introduction to Vision Assistant 7.1, on which our research work is focused. The fifth chapter discusses the use of Particle analysis and its extension for counting the number of microorganisms, finally concluding thesis in sixth chapter with future scope.

## Table of Contents

---

Declaration	i
Acknowledgement	ii
Abstract	iii
Organization of Thesis	iv
Table of Contents	v
List of Figures	x
List of Tables	xii
List of Abbreviations	xiii
<b>Chapter 1</b>	
<b>Introduction</b>	<b>1</b>
<b>1.1 Bacteria</b>	<b>2</b>
1.1.1 Characteristics of bacteria	3
1.1.2 Growth and reproduction of Bacteria	4
1.1.2.1 Lag Phase	6
1.1.2.2 Log Phase	6
1.1.2.3 Stationary Phase	7
1.1.2.4 Decline Phase	7
<b>1.2 Agar Plate</b>	<b>7</b>
<b>1.3 Bacterial Colony</b>	<b>8</b>

1.3.1 Features of Bacterial Colony	9
1.3.2 Bacteria in a Colony	11
<b>1.4 Colony Counter</b>	<b>11</b>
1.4.1 Colony Plate Types	12
<b>1.5 Literature Survey</b>	<b>13</b>
<b>Chapter 2</b>	
<b>Machine Vision</b>	<b>16</b>
<b>2.1 Anatomy of Machine Vision</b>	<b>17</b>
<b>2.2 Components of Machine Vision</b>	<b>18</b>
2.2.1 Camera	18
2.2.2 Image processor	19
2.2.3 Display unit	19
<b>2.3 Basics of Digital image</b>	<b>19</b>
2.3.1 Digital image	20
2.3.2 Properties of a digitized image	20
2.3.2.1 Image resolution	20
2.3.2.2 Image definition	20
2.3.2.3 Number of planes	21
2.3.3 Image types	21
2.3.3.1 Grayscale images	22
2.3.3.2 Color images	22
2.3.3.3 Complex images	22
<b>2.4 Functions of Machine Vision</b>	<b>23</b>

<b>2.5 Applications of Machine Vision</b>	<b>24</b>
<b>2.6 Advantages and Limitations of Machine Vision</b>	<b>26</b>
<b>2.7 Comparison between Machine Vision and Human Vision</b>	<b>27</b>
<b>2.8 Comparison between Machine Vision and Computer Vision</b>	<b>28</b>
<b>Chapter 3</b>	
<b>Setup and Configuration</b>	
<b>3.1 Setup Overview</b>	<b>30</b>
3.1.1 Setting Up the Hardware	31
3.1.1.1 Subnet Considerations	32
3.1.1.2 CVS-1450: Hardware	32
3.1.1.3 Connecting the CVS-1450 Device to a Network	32
3.1.1.4 Connecting a Camera and Monitor to the CVS-1450	33
3.1.1.5 Wiring Power to the CVS-1450 Device	34
3.1.1.6 Connecting the CVS-1450 to the Development Computer	35
3.1.2 Setting up the Development Computer	36
3.1.3 Acquiring an Image	37
<b>3.2 IMAQ for IEEE 1394 (Fire wire) compatible cameras</b>	<b>37</b>
3.2.1 High-level functions	38
3.2.2 Low-level functions	38
<b>3.3 Acquisition flow</b>	<b>39</b>
3.3.1 Initialization	39
3.3.2 Configuration	41
3.3.3 Acquisition	42



## **Chapter 4**

### **Vision Assistant**

<b>4.1 Introduction to Vision Assistant 7.1</b>	<b>45</b>
<b>4.2 System requirements and installation</b>	<b>45</b>
4.2.1 Installing Vision Assistant	46
4.2.2 Features	46
<b>4.3 Processing functions of NI Vision Assistant</b>	<b>47</b>
4.3.1 Image functions	47
4.3.2 Color functions	48
4.3.3 Grayscale Functions	48
4.3.4 Binary Functions	49
4.3.5 Machine Vision Functions	49
<b>4.4 Particle Analysis</b>	<b>51</b>
4.4.1 Applications	51
4.4.2 Concepts	52
<b>4.5 Thresholding</b>	<b>54</b>
4.5.1 Applications	54
4.5.2 Intensity Threshold	54
4.5.3 Thresholding Example	55
4.5.4 Automatic Threshold	56

## **Chapter 5**

### **Problem Statement and Solution**

<b>5.1 Problem Statement</b>	<b>58</b>
------------------------------	-----------

<b>5.2 Justification of Problem Statement</b>	<b>58</b>
<b>5.3 Problem solution</b>	<b>59</b>
<b>5.4 A warm-up exercise</b>	<b>59</b>
<b>5.5 Problem Algorithm</b>	<b>61</b>
5.5.1 Acquiring image of Petri dish	61
5.5.2 Opening the stored image	62
5.5.3 Extracting color planes from an image	64
5.5.4 Filtering the Image	65
5.5.5 Separating Particles from the Background with Thresholding	66
5.5.6 Modifying Particles with Morphological Functions	68
5.5.7 Isolating Circular Particles	71
5.5.8 Analyzing Circular Particles	73
5.5.9 The Complete Processing Script	74
5.5.10 Estimating Processing Time	76
<b>5.6 Results and discussion</b>	<b>77</b>
<b>Chapter 6</b>	
<b>Conclusions &amp; Future scope</b>	<b>81</b>
6.1 Conclusions	81
6.2 Future scope	81
<b>References</b>	<b>83</b>

## List of Figures

---

Fig.1.1 Bacteria colonies on a 100mm Petri dish	1
Fig.1.2 Bacteria	3
Fig.1.3 Bacterial growth curve	5
Fig.1.4 Agar plate	8
Fig.1.5 Form, Elevation and Margin	10
Fig.1.6 Colony Plate Types	12
Fig.2.1 Anatomy of a machine vision system	18
Fig.2.2 Effect of image processor	19
Fig.2.3 Spatial reference of the (0, 0) pixel	20
Fig.3.1 CVS-1450 Series Front Panel	31
Fig.3.2 Basic Hardware	33
Fig.3.3 Wiring Power to the CVS-1450 Device	34
Fig.3.4 Ethernet Connection	35
Fig.3.5 Relationships between cameras, interface files, and camera files	40
Fig.4.1 Sample list of parameters	52
Fig. 4.2 Image Histogram	54
Fig.4.3 Example of Thresholding	55
Fig.4.4 Brightest area of Thresholding	55
Fig.5.1 Synthetic Image of particle to be counted	60
Fig.5.2 Result of particle Counting	60
Fig.5.3 Setup of acquiring image	61
Fig.5.4 Hardware Setup	62

Fig.5.5 Acquiring images in Vision Assistant	62
Fig.5.6 Opening an image	63
Fig.5.7 Extracting RGB-Red Plane	64
Fig.5.8 Filtering the Image	66
Fig.5.9 Separating Particles from the Background with Thresholding	67
Fig.5.10 Modifying Particles with Proper Close	69
Fig.5.11 Modifying Particles with fill holes	70
Fig.5.12 Modifying Particles with Remove Border Objects	71
Fig.5.13 Isolating Circular Particles	72
Fig.5.14 Counting Circular Particles	74
Fig. 5.15 Processing Script	75
Fig. 5.16 Processing Script (Continued)	75
Fig.5.17 Estimating Processing Time of an image	76

## List of Tables

---

Table 1.1 Generation time	6
Table 2.1 Bytes per pixel	23
Table 2.2 Functions and Applications of Machine Vision	26
Table 2.3 Comparison between Machine Vision and Human Vision	27
Table 2.4 Comparison between Machine Vision and Computer Vision	28
Table 3.1 Camera naming conventions	39
Table 3.2 Decoder inputs and corresponding outputs	44
Table 4.1 Values of the Particle	53
Table 5.1 Results and discussion	79

## List of Abbreviations

---

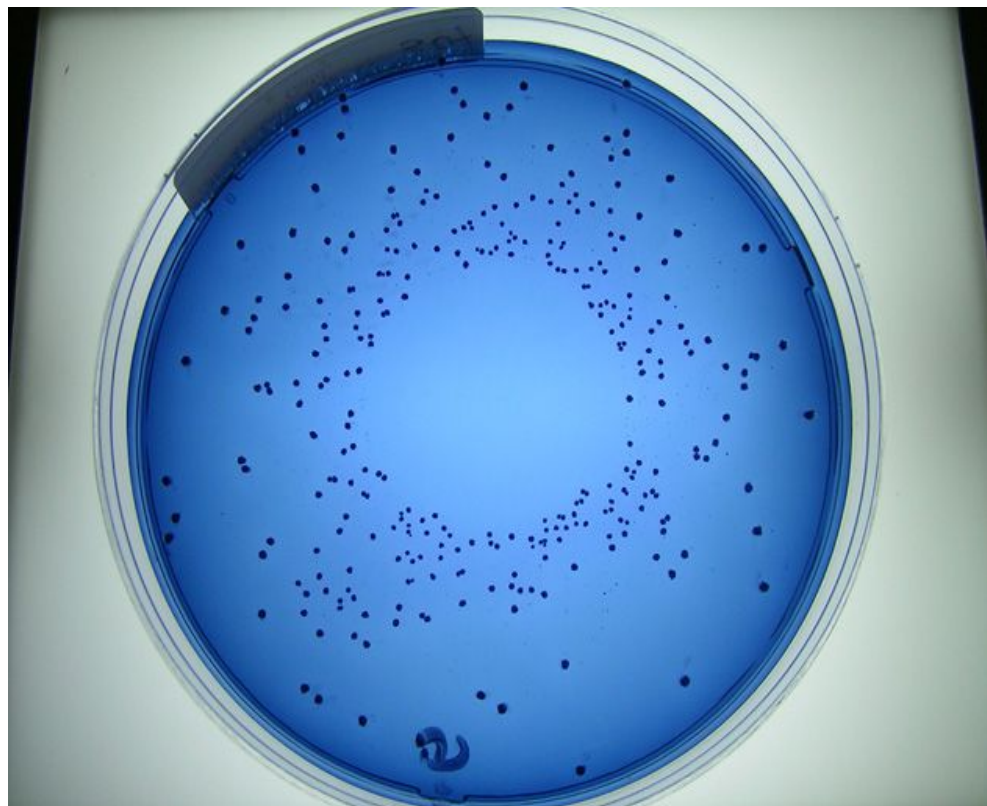
API	Application Programming Interface
CAT 5	Category 5
CCD	Charged Coupled Device
CVS	Compact Vision System
DCAM	Digital Camera
HSL	Hue Saturation Luminescence
I/O	Input/Output
IEEE	Institute for Electrical and Electronic Engineers
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
LAN	Local Area Network
NI	National Instruments
RGB	Red Green Blue
ROI	Region of Interest
RFM	Rumen Fluid Medium
TV	Television
VI	Virtual Instrument

# Chapter 1

## Introduction

---

The present work relates to a colony counter for bacterial specimens on a petri dish using Machine Vision which overcomes all of the disadvantages of the previously known devices. A colony counter is an instrument used to count colonies of bacteria or other microorganisms growing on an agar plate. Bacterial colony is a group of bacteria growing on a plate that is derived from one original starting cell. An agar plate is a sterile Petri dish that contains a growth medium (typically agar plus nutrients) used to culture microorganisms. Bacterial colony counting process is usually performed by well-trained technicians manually. However, there might exist hundreds of colonies in a traditional 100mm Petri dish as shown in figure 1.1:



**Figure 1.1: Bacteria colonies in a Petri dish**

Therefore, this manual enumeration process has a very low throughput and is time consuming and labor intensive in practice. In addition, the manual counting is an error-prone process since the counting results of the same plate obtained from different technicians might vary, especially when a vast number of colonies appear on the plate. Another possible cause of variation is the judgment of the indistinguishable colony overlaps. Thus, it is important to have consistent criteria for measuring overlapped colonies. To produce consistent and accurate results and improve the throughput, the existing colony counter device has been developed.

Machine vision based bacteria colony counter would remove a tedious chore and allow each plate to be counted with equal efficiency. This method is used not only in medical examinations, but also in the food and drug safety evaluations, environmental monitoring and public health. One of the functions of this Machine is to count discrete areas within a selected brightness range.

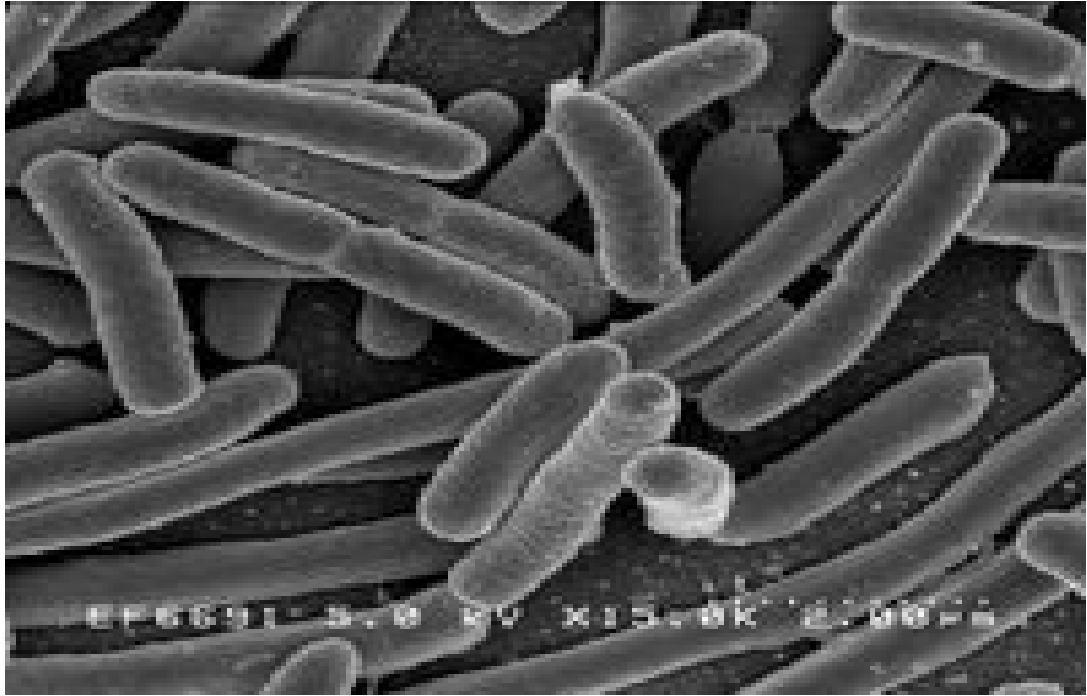
## 1.1 Bacteria

**Bacteria** are unicellular microorganisms. They are typically a few micrometres long and have many shapes including curved rods, spheres, rods, and spirals. The study of bacteria is bacteriology, a branch of microbiology. Bacteria are ubiquitous in every habitat on earth, growing in soil, acidic hot springs, radioactive waste, seawater, and deep in the earth's crust. There are typically 40 million bacterial cells in a gram of soil and a million bacterial cells in a millilitre of fresh water; in all, there are approximately five nonillion ( $5 \times 10^{30}$ ) bacteria in the world. Bacteria are vital in recycling nutrients, and many important steps in nutrient cycles depend on bacteria, such as the fixation of nitrogen from the atmosphere. However, most of these bacteria have not been characterised, and only about half of the phyla of bacteria have species that can be cultured in the laboratory.

There are approximately 10 times as many bacterial cells as human cells in the human body, with large numbers of bacteria on the skin and in the digestive tract. Although the vast majority of these bacteria are rendered harmless or beneficial by the protective effects of the immune system, a few pathogenic bacteria cause infectious diseases, including cholera, syphilis, anthrax, leprosy and bubonic plague. The most common fatal bacterial diseases are respiratory infections, with tuberculosis alone killing about 2 million people a



year, mostly in sub-Saharan Africa. In developed countries, antibiotics are used to treat bacterial infections and in various agricultural processes, so antibiotic resistance is becoming common. Bacteria like *Escherichia coli* cells is shown in figure 1.2:



**Figure 1.2: Bacteria**

In industry, bacteria are important in processes such as wastewater treatment, the production of cheese and yoghurt, and the manufacture of antibiotics and other chemicals.

Bacteria are prokaryotes. Unlike cells of animals and other eukaryotes, bacterial cells do not contain a nucleus and rarely harbour membrane-bound organelles. Although the term *bacteria* traditionally included all prokaryotes, the scientific classification changed after the discovery in the 1990s that prokaryotic life consists of two very different groups of organisms that evolved independently from an ancient common ancestor. These evolutionary domains are called Bacteria and Archaea [http:Wiki-b].

#### **1.1.1 Characteristics of bacteria:**

Bacteria are relatively small. Bacterial cells do not contain a nucleus or other membrane-bound organelles. These bacteria are Single-celled & simple morphologies. These bacteria are like primarily synthesizers or absorbers. Most bacteria do not cause human diseases,

but most infectious diseases are caused by bacteria (and viruses). More typically, bacteria are beneficial, whether to ecosystems or directly to individual organisms.

### 1.1.2 Growth and reproduction of Bacteria

**Growth** is an orderly increase in the quantity of cellular constituents. It depends upon the ability of the cell to form new protoplasm from nutrients available in the environment. In most bacteria, growth involves increase in cell mass and number of ribosome, duplication of the bacterial chromosome, synthesis of new cell wall and plasma membrane, partitioning of the two chromosomes, septum formation, and cell division [http:Growth-1].

Unlike multicellular organisms, increases in the size of bacteria (cell growth) and their reproduction by cell division are tightly linked in unicellular organisms. Bacteria grow to a fixed size and then reproduce through **binary fission**, a form of asexual reproduction. Under optimal conditions, bacteria can grow and divide extremely rapidly, and bacterial populations can double as quickly as every 9.8 minutes. In cell division, two identical clone daughter cells are produced. Some bacteria, while still reproducing asexually, form more complex reproductive structures that facilitate the dispersal of the newly formed daughter cells. Examples include fruiting body formation by Myxobacteria and arial hyphae formation by Streptomyces, or budding. Budding involves a cell forming a protrusion that breaks away and produces a daughter cell.

In the laboratory, bacteria are usually grown using solid or liquid media. Solid growth media such as agar plates are used to isolate pure cultures of a bacterial strain. However, liquid growth media are used when measurement of growth or large volumes of cells are required. Growth in stirred liquid media occurs as an even cell suspension, making the cultures easy to divide and transfer, although isolating single bacteria from liquid media is difficult. Most laboratory techniques for growing bacteria use high levels of nutrients to produce large amounts of cells cheaply and quickly. However, in natural environments nutrients are limited, meaning that bacteria cannot continue to reproduce indefinitely. This nutrient limitation has led the evolution of different growth strategies. Some organisms can grow extremely rapidly when nutrients become available, such as the formation of algal (and cyanobacterial) blooms that often occur in lakes during the summer. Other organisms have adaptations to harsh environments, such as the production of multiple antibiotics by

Streptomyces that inhibit the growth of competing microorganisms. In nature, many organisms live in communities (e.g. biofilms) which may allow for increased supply of nutrients and protection from environmental stresses. These relationships can be essential for growth of a particular organism or group of organisms (syntrophy) [http:Wiki-b].

### Phases of Growth

A typical bacterial growth curve depicts the changes in the number of viable bacteria that occur when a resting bacterium is exposed to fresh growth medium [http:Growth-2]. This growth curve can be divided into four operationally distinct phases:

1. Lag phase
2. Logarithmic (log) phase
3. Stationary phase
4. Decline or death phase

The Phases of growth is shown in figure 1.3:

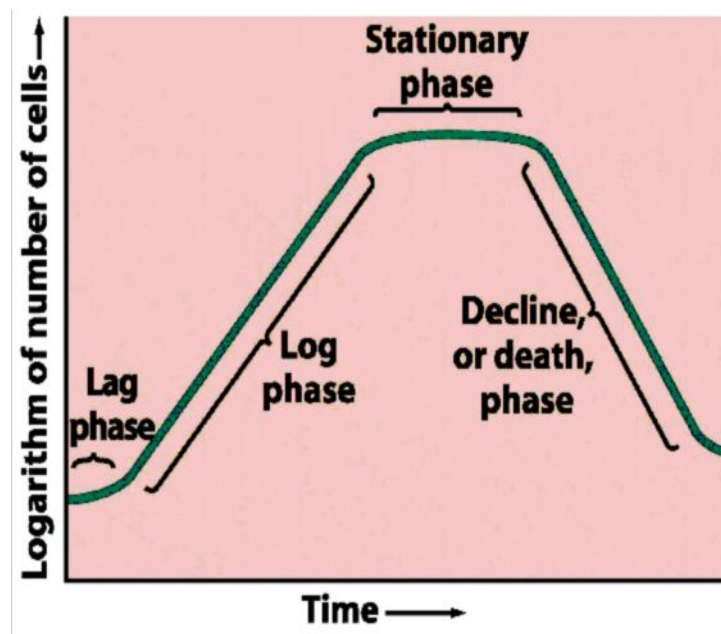


Figure 1.3: Bacterial growth curve

### 1.1.2.1 Lag Phase

When a population of bacteria first enters a high-nutrient environment that allows growth, the cells need to adapt to their new environment. The first phase of growth is the lag phase, a period of slow growth when the cells are adapting to fast growth. The lag phase has high biosynthesis rates, as enzymes and nutrient transporters are produced. Organisms do not increase significantly in number; they are acclimating to the new medium or environment. They are metabolically active. The cells grow in size, synthesize enzymes, and incorporate molecules from the medium. As the individual organisms grow in size, also produce large quantities of energy.

### 1.1.2.2 Log Phase

The second phase of growth is the logarithmic phase (log phase), also known as the exponential phase. The log phase is marked by rapid exponential growth. The rate at which cells grow during this phase is known as the growth rate, and the time it takes the cells to double is known as the generation time. During log phase, nutrients are metabolised at maximum speed until one of the nutrients is depleted and starts limiting growth. Once the organisms have adapted to a growth medium, Growth occurs at an exponential (log) rate. This is the healthiest time for a culture. It is the best to use for experiments or research. The population doubles in each generation time which is shown in table 1.1:

For **example**:

<b>Organisms per ml</b>	<b>Time (minutes)</b>
1,000	0 minutes
2000	20 minutes
4000	40 minutes
8000	60 minutes or 1 hr
64000	2 hr
512000	3 hr

**Table 1.1: Generation time**

Such growth is said to be exponential or logarithmic. The generation time for most bacteria is between 20 minutes and 20 hours, with most typically less than 1 hour.

### **1.1.2.3 Stationary Phase**

The third phase of growth is the stationary phase and is caused by depleted nutrients. The cells reduce their metabolic activity and consume non-essential cellular proteins. The stationary phase is a transition from rapid growth to a stress response state and there is increased expression of genes involved in DNA repair, antioxidant metabolism and nutrient transport. Cell division decreases to a point that new cells are produced at same rate as old cell die. The number of live cells stays constant. No net increase or decrease in cell numbers.

### **1.1.2.4 Decline (Death) Phase**

If incubation continues after the population reaches stationary phase, a death phase follows, in which the number of cell population declines. During the death phase, the number of live cells decreases geometrically (exponentially), essentially the reverse of growth during the log phase. Conditions in the medium become less and less supportive of cell division. Cell loses their ability to divide and thus die.

## **1.2 Agar plate**

An **agar plate** is a sterile Petri dish that contains a growth medium (typically agar plus nutrients) used to culture microorganisms. Selective growth compounds may also be added to the media, such as antibiotics.

Individual microorganisms placed on the plate will grow into individual colonies, each a clone genetically identical to the individual ancestor organism (except for the low, unavoidable rate of mutation). Thus, the plate can be used either to estimate the concentration of organisms in a liquid culture or a suitable dilution of that culture, using a colony counter, or to generate genetically pure cultures from a mixed culture of genetically different organisms, using a technique known as **streaking**. In this technique, a drop of the culture on the end of a thin, sterile loop of wire is "streaked" across the surface of the agar leaving organisms behind, a higher number at the beginning of the streak and a lower

number at the end. At some point during a successful "streak", the number of organisms deposited will be such that distinct individual colonies will grow in that area which may be removed for further culturing, using another sterile loop [http:Wiki-a]. Agar plate is shown in figure 1.4:



**Figure 1.4: Agar plate**

### **1.3 Bacterial Colony**

Bacterial colony is a group of bacteria supposed to be derived from a single bacterium. Usually Bacteria is grown on a culture medium and a single bacterium divides by binary fission (some archaea also show budding) to produce identical copies of itself; therefore also called clones of each other. So, bacterial colony is a group of bacteria growing on a plate that is derived from one original starting cell. All of the bacterial cells in one colony are clones of that original cell since the bacteria reproduce through binary fission.

Bacteria grow on solid media as colonies. A colony is defined as a visible mass of microorganisms all originating from a single mother cell; therefore a colony constitutes a clone of bacteria all genetically alike.

Bacteria grow tremendously fast when supplied with an abundance of nutrients. Different types of bacteria will produce different-looking colonies, some colonies may be colored, some colonies are circular in shape, and others are irregular. The characteristics of a

colony (shape, size, pigmentation, etc.) are termed the colony morphology. Colony morphology is a way scientists can identify bacteria. In fact there is a book called Bergey's Manual of Determinative Bacteriology (commonly termed Bergey's Manual) that describes the majority of bacterial species identified by scientists so far. This manual provides descriptions for the colony morphologies of each bacterial species.

### **1.3.1 Features of Bacterial Colony**

**Form** - is the basic shape of the colony. For example: circular, filamentous etc.

**Elevation** - is the cross sectional shape of the colony which is Turn the Petri dish on end.

**Margin** - is the magnified shape of the edge of the colony.

**Surface** – this shows the appearance of the surface of the colony. For example, smooth, glistening, rough, dull (opposite of glistening), rugose (wrinkled), etc.

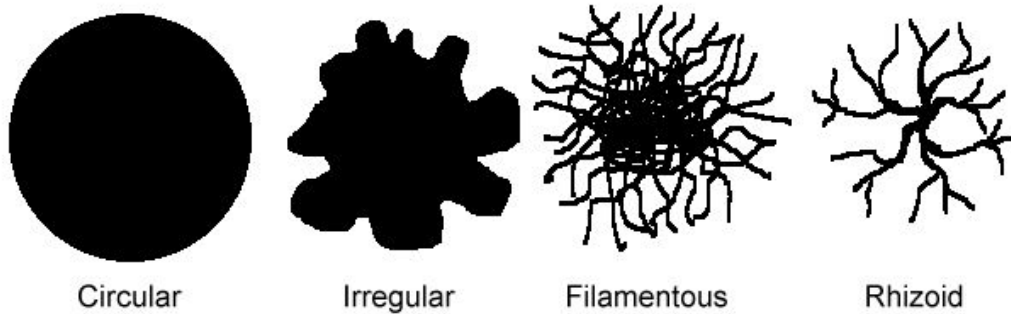
**Opacity** - For example, transparent (clear), opaque, translucent (almost clear, but distorted vision, like looking through frosted glass), iridescent (changing colors in reflected light), etc.

**Chromogenesis (pigmentation)** - For example, white, buff, red, purple etc.

In bacterial colony assays, the patterns are formed within culture media that has been inoculated with bacterial cells. This allows the cells to reproduce and form bacterial colonies within and/or on the surface of the medium. When the colonies are sufficiently large, they are usually visible to the naked eye, which allows researchers to determine the number of colonies formed. In addition, various visual characteristics of the colonies, such as shape, size, pigmentation, and opacity, can be used to help determine the type of bacterium present. In bacterial colony counting, the colonies formed are enumerated, either manually or using automated image analysis techniques. A general problem for microbiologists is determining the number of phenotypically similar colonies growing on an agar plate that must be analyzed in order to be confident of identifying all of the different strains present in the sample. If a specified number of colonies are picked from a plate on which the number of unique strains of bacteria is unknown, assigning a

probability of correctly identifying all of the strains present on the plate is not a simple task. Refer to the figure1.5 for illustrated examples of form, elevation, and margin:

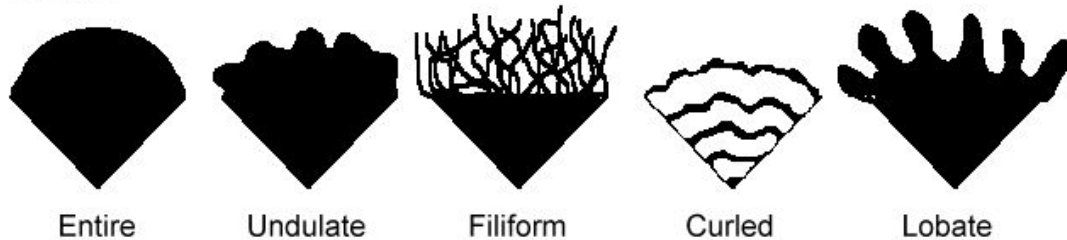
### Form



### Elevation



### Margin



**Figure1.5: Form, Elevation and Margin**

With *Escherichia coli* of avian cellulites origin as a case study, a statistical model was designed that would delineate sample sizes for efficient and consistent identification of all the strains of phenotypically similar bacteria in a clinical sample. This model enables the microbiologist to calculate the probability that all of the strains contained within the sample are correctly identified and to generate probability-based sample sizes for colony identification. The probability of cellulites lesions containing a single strain of *E. coli* was 95.4%. If one *E. coli* strain is observed out of three colonies randomly selected from a future agar plate, the probability is 98.8% that only one strain is on the plate [[http: Colony](http://Colony)].



### 1.3.2 Bacteria in a Colony

The colony was one cell at the beginning. If you grew the culture for ~10 h. on LB without antibiotic => assume that cells divide every 20 min => your cells had 30 division cycles => there is  $2^{30} = 1048576$  cells in a colony. Here we assume that there is no growth substrate depletion taking place as long as a colony grows. In reality, cells that are on top of the colony are much depleted in nutrients, and grow very slowly, if at all, and the fastest growing cells are those growing in contact with the medium and on the outer rim of the colony. Thus, what we can say without titration is that the real number of cells in a colony is lower than theoretical. How much lower depends of richness of medium and solid support, morphology of colonies, etc. Cells growing in presence of antibiotics will grow slower; they will divide every 40-60 min [Khatipov, 2001].

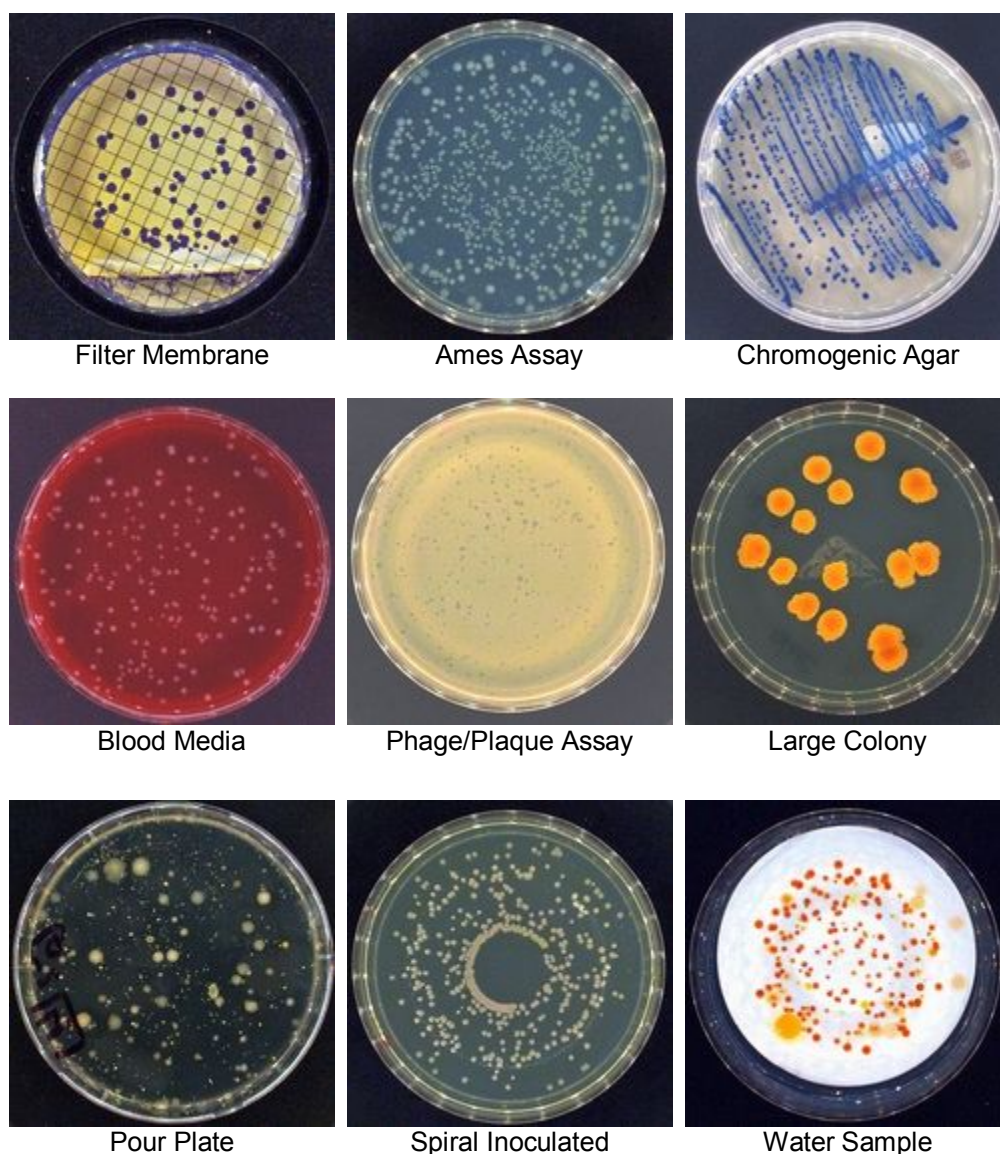
### 1.4 Colony Counter

A **colony counter** is an instrument used to count colonies of bacteria or other microorganisms growing on an agar plate. Early counters were merely lighted surfaces on which the plate was placed, with the colonies marked off with a felt-tipped pen on the outer surface of the plate while the operator kept the count manually. More recent counters attempt to count the colonies electronically, by identifying individual areas of dark and light according to automatic or user-set thresholds, and counting the resulting contrasting spots.

Such counters are used to estimate the density of microorganisms within a liquid culture. An appropriate dilution, or several dilutions within the estimated appropriate range, is spread using sterile technique on the agar plate, which is then incubated under the appropriate conditions for growth until individual colonies appear. Each colony marks the spot where a single organism was originally placed, thus the number of colonies on the plate equals the number of organisms within the volume of liquid spread on the plate. That concentration is then extrapolated by the known dilution from the original culture, to estimate the concentration of organisms within that original culture. The maximum number of colonies which may be effectively counted on a single plate is somewhere between 100 and 1,000, depending on the size of the colony and the type of organism [http:Wiki-c].

### 1.4.1 Colony Plate Types

Digital Counter excels at counting difficult to read plates. High resolution color digital imaging better discriminates colonies. True color enables counting mixed cultures and discriminates colonies from debris. Lighting above and below (dark field or bright field) the samples provides better contrast and more accurate reading. Giles Scientific will work with laboratory to optimize our digital imaging for assays posing unique difficulties and challenges. Figure 1.6 shows the types of colony plates. In our work, we have used different type of colony plates like nutrient agar media, blood media & water sample.



**Figure 1.6: Colony Plate Types**

## 1.5 Literature Survey

**Caldwell R. et.al** had given Medium without Rumen Fluid for Nonselective Enumeration and Isolation of Rumen Bacteria in 1966. In this Method, Colony counts which approximated those in a habitat-simulating, rumen fluid-agar medium (RFM) were obtained in medium 10. Colony counts were also reduced when medium 10 was modified to contain higher concentrations of Trypticase or volatile fatty acids. Significant differences were found between colony counts obtained from diluted rumen contents of animals fed a cracked corn-urea diet, and the colony counts obtained from animals fed either a cracked corn-soybean oil meal or an alfalfa hay-grain diet. The results show that medium 10 is suitable for enumeration and isolation of many predominant rumen bacteria [Caldwell R. et.al, 1966].

**Gilchrist E. et.al** had given Spiral Plate Method for Bacterial Determination in 1973. A method is described for determining the number of bacteria in a solution by the use of a machine which deposits a known volume of sample on a rotating agar plate in an ever decreasing amount in the form of an Archimedes spiral. After the sample is incubated, different colony densities are apparent on the surface of the plate. A modified counting grid is described which relates area of the plate of volume of sample. By counting an appropriate area of the plate, the number of bacteria in the sample is estimated. This method was compared to the pour plate procedure with the use of pure and mixed cultures in water and milk [Gilchrist E. et.al, 1973].

**Coyne M. et.al** had given Bacterial Colony Counting Using the M.R.C. Image Analyzer in 1974. In this method, the optical considerations in illuminating bacterial colonies grown on agar plates are described. An arrangement which gives a high contrast image with almost total background suppression and good separation of touching and overlapping colonies is shown [Coyne M. et.al, 1974].

**Dubuisson Marie-Pierrea et.al** had given Segmentation and classification of bacterial culture images in 1994. A procedure using simple image processing and pattern recognition techniques to automatically extract and count organisms using images of bacterial cultures is described in this paper. The methanogens were grown in anaerobic serum vials for 1–3 week period. Samples of the growing cultures were withdrawn from the vials and

photographed. The procedure was applied to images containing one kind of organism as well as images containing both types of organisms. The image analysis and pattern recognition techniques described in this paper are simple, fast, and able to identify organisms based on their shapes [Dubuisson Marie-Pierrea et.al, 1994].

**Mukherjee Dipti et.al** had given Water quality analysis which is based on a pattern recognition approach in 1995. In this method, the development of a microcomputer-based system for the automatic counting of colonies of indicator organisms (that is, bacteria whose presence is indicative of the presence of pathogens) trapped on membrane filters from water samples is described. Three different approaches, based on mathematical morphology, distance transform and fuzzy c-means clustering respectively are implemented [Mukherjee Dipti et.al, 1995].

**Kawai M. et.al** had given Rapid enumeration of physiologically active bacteria in purified water used in the pharmaceutical manufacturing process in 1999. According to this method, Bacteria with growth potential were enumerated using the micro-colony technique and direct viable counting (DVC). Respiring and esterase-active bacteria were detected by fluorescent staining. A large number of bacteria in purified water retained physiological activity, while most could not form colonies on conventional media. The techniques applied in this study enabled bacteria to be counted within 24 h so results could be available within one working day. These rapid and convenient techniques should be useful for the systematic monitoring of bacteria in water used for pharmaceutical manufacturing[Kawai M. et.al, 1999].

**Paul C. Olsztyn et.al** had given Bacteria colony counter and classifier in 1999. This invention relates to a colony counter and classifier for bacterial specimens on a Petri dish. The device includes a housing having a tray which accepts the Petri dish containing the bacteria colonies. A line scan camera is mounted within the housing above the tray. Upon actuation, a linear motor transports the tray across the scanning line of vision for the line scan camera. Simultaneously, a fiber optic illumination system illuminates the Petri dish from its side opposite the camera. As the tray is transported by the linear motor, the line scan camera successively obtains an optical image of the Petri dish containing the bacteria colonies. Each line scan has a width equal to one pixel and the resulting optical images from the line scan camera are combined together to form an overall optical image of high

resolution. That image is stored utilizing a digital computer. The computer then analyzes the stored optical image to both identify and count the bacteria colonies [Paul C. Olsztyn et.al, 1999].

**John M. Kramer et.al** had searched Evaluation of the spiral plate and laser colony counting techniques for the enumeration of bacteria in foods in 2004. In this method, samples of food and dairy products, bacterial cultures and spore suspensions were examined by two operators using both the spiral plate and surface drop techniques for counting bacteria. An electronic laser counter used in this study was found to give comparable results ( $r=0.966$ ) to the grid-method of colony counting in a substantially shorter time. Analysis of operation times and material requirements for each method showed that significant savings in cost, time, space and support labour could be achieved with the spiral plate method over conventional techniques [John M. Kramer et.al, 2004].

**Putman M. et.al** had given simplified method to automatically count bacterial colony forming unit in 2005. This method shows that an automated colony counter can process images obtained with a digital camera or document scanner and that any laboratory can efficiently have bacterial colonies enumerated by sending the images to a laboratory with a colony counter via internet. Bacterial colony counting is a significant technical hurdle for vaccine studies as well as various microbiological studies [Putman M. et.al 2005].

**Bang Wei et.al** had given an Effective and Robust Method for Automatic Bacterial Colony Enumeration in 2007. In this study, they propose a fully automatic colony counter and compare its performance with Clono-Counter. These experimental results show that the proposed method outperforms Clono-Counter in terms of Precision, Recall, and F-measures [Bang Wei et.al, 2007].

## Chapter 2

### Machine Vision

---

We see the world around through our eyes. Our eyes are the sensory organs that capture images and transmit to our brain at very fast rate. The image is representation of real scene either in black & white or in color. The brain performs various processing functions and vision is perceived. In human beings we make use of vision for accomplishing majority of our tasks. Blindfolding ourselves and observing how our daily routine is seriously hampered without our vision can easily verify this fact.

Although the first machine that captured image was a pinhole camera that was invented way back in 1850s, which was followed by many advances in image capturing techniques [http: Machine-b]. Black & white camera gave way to colored camera, resolution of picture captured enhanced, moving pictures were captured using monochrome T.V. Camera followed by colored T.V. camera and now a days we have digital cameras as small as a size of button, embedded in our mobile phones, at a price, a student can afford from his pocket money

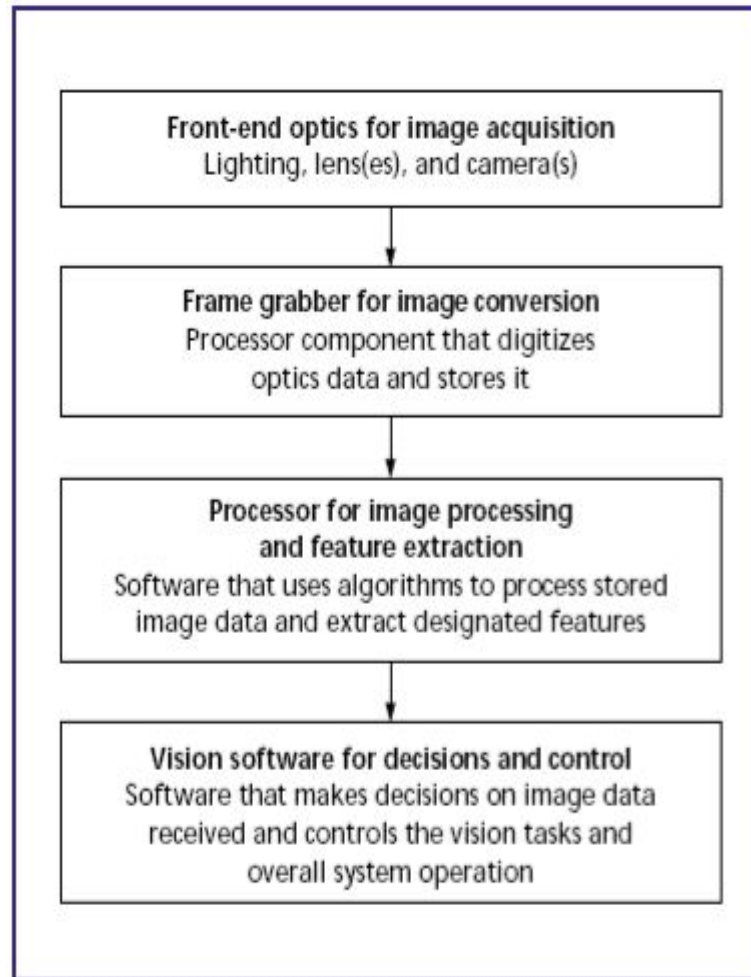
However, when we talk of machine vision, these cameras alone do not “see” the way we do. With human vision the process of “seeing” i.e. capturing images takes place at a very fast rate, almost continuously for several hours. The most important aspect is equally fast processing of these images so as to accomplish necessary decisions and actions. Image processing is collection of programs and techniques that improve, simplify, enhance or otherwise alter an image. Machine vision also aims at aping this task of capturing images at fast rates and simultaneously processing and analysis of these images so that machine can “see”, the way we do.

Machine Vision is concerned with the engineering of integrated mechanical-optical-electronic-software systems for examining natural objects and materials, human artifacts and manufacturing processes, in order to detect defects and improve quality, operating efficiency and the safety of both products and processes [Whelan F.,2001].

## 2.1 Anatomy of Machine Vision

There are a variety of components that comprise a machine vision system. Decisions about components depend on the environment, application, and budget. There are, however, several common ingredients to all vision systems [http: Machine-a].

Anatomy of machine vision system is shown in figure 2.1:



**Figure 2.1: Anatomy of a machine vision system**

### Front-end optics

The front end includes the lighting, the lens, and the camera. The lighting highlights those features important to the measurement, while minimizing distracting artifacts. The lens

must be capable of faithfully imaging the critical features of the object. The camera electronically records the images.

### **Frame grabber**

A frame grabber is a processor board that accepts the video input from the camera, digitizes it, and stores it for analysis. Some frame grabbers include special processing electronics that speed the image processing and feature extraction tasks.

### **Processor**

A processor is required to control the vision application. Pentium-class processors are capable of running vision applications with no special processing hardware.

### **Software**

Computer software to control and execute the vision tasks is needed. There are many flexible machine vision software packages that provide extensive vision tools for programmers to use when writing applications. Developing vision software requires skill and experience, even when using good tools.

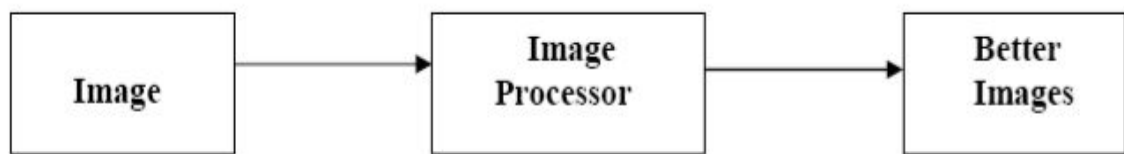
## **2.2 Components of Machine Vision**

Machine Vision is related to artificial intelligence. We take the images in real time and process them for extracting useful information from them. The system making use of machine vision has basically three components: Camera (Prosilica 2.0.1), Image Processor (CVS-1450), Display Unit (PC monitor).

**2.2.1 Camera** is used for taking pictures of desired field, spot, point or any moving object. Camera must have high grabbing or sampling rate (Frames to be taken in one second). Frames are to be taken at fast rate such that we do not lose information about the object. These frames are to be taken in real time that's why frames are blurred and this needs image processing. Cameras used are mainly CCD (Charge coupled Devices) type or digital cameras. Resolution of camera must be strictly decided according to the object's area and size. In our work we have used digital camera PROSILICA 2.0.1, IEEE-1394 compatible.



**2.2.2 Image processor** is used to get the best out of grabbed or captured images we make use of image processor. Image processor is usually a computer, laptop or a digital signal processor. An image is defined as the representation of the real scene, either in black and white or in color, and either in print form or in digital form. Printed images are reproduced either by multiple colors and, gray scales or by a single ink source. Most print application, only one color of ink is available (such as black ink on white paper in a newspaper or copier). In such cases changing the ratio of black versus white areas produces all gray levels. The effect of image processor is shown in figure 2.2:



**Figure 2.2: Effect of image processor**

Image is divided into small sections on screen called picture cells, or pixels, where the size of all pixels is the same, while intensity of light in each pixel is varied to create the images. Image processor processes the blurred image pixel by pixel or sometimes reference points are processed which represent the whole image and make the picture more clearly to the operator for further analysis. The more the pixels present, the better the resolution of camera. We have used Machine Vision Assistant 7.1 simulator software to perform image-processing tasks with CVS-1450 [http: Machine-b].

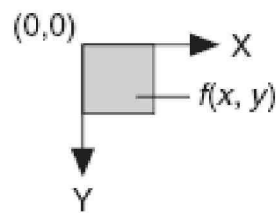
**2.2.3 Display unit** is mainly a screen having pixels to display the grabbed images. A personal computer or laptop is used for this purpose. Sometimes storage of grabbed images is also done with computer for offline analysis. We have used simple PC monitor to get images from camera. Camera is connected to monitor through CVS-1450 using LAN, which directly displays the acquired image on monitor.

## **2.3 Basics of Digital image**

Digital Image Processing is done to process any image to get useful information from it. For Image processing, we must know about the image basics. These basics are given as:

### 2.3.1 Digital image

An image is a 2D array of values representing light intensity. For the purposes of image processing, the term image refers to a digital image. An image is a function of the light intensity  $f(x, y)$  where  $f$  is the brightness of the point  $(x, y)$ , and  $x$  and  $y$  represents the spatial coordinates of a picture element, or pixel. By convention, the spatial reference of the pixel with the coordinates  $(0, 0)$  is located at the top, left corner of the image. We can noticed in figure2.3 that the value of  $x$  increases moving from left to right, and the value of  $y$  increases from top to bottom.



**Figure 2.3: Spatial reference of the  $(0, 0)$  pixel**

In digital image processing, an imaging sensor converts an image into a discrete number of pixels. The imaging sensor assigns to each pixel a numeric location and a gray level or color value that specifies the brightness or color of the pixel.

### 2.3.2 Properties of a digitized image

A digitized image has three basic properties: resolution, definition, and number of planes. These three properties are described:

#### 2.3.2.1 Image resolution

Image Resolution is spatial resolution of an image, which is determined by its number of rows and columns of pixels.

#### 2.3.2.2 Image definition

Image definition indicates the number of shades that we can see in the image. The bit depth of an image is the number of bits used to encode the value of a pixel. For a given bit depth of  $n$ , the

image has an image definition of  $2^n$ , meaning a pixel can have  $2^n$  different values. For example, if  $n$  equals 8 bits, a pixel can have 256 different values ranging from 0 to 255. If  $n$  equals 16 bits, a pixel can have 65,536 different values ranging from 0 to 65,535 or from  $-32,768$  to  $32,767$  [NI-c]. The manners in which we encode our image depends on the nature of the image acquisition device, the type of image processing need to use, and the type of analysis we need to perform.

For example, 8-bit encoding is sufficient if we need to obtain the shape information of objects in an image. However, if we need to precisely measure the light intensity of an image or region in an image, we must use 16-bit or floating-point encoding. Software does not directly support other types of image encoding, particularly images encoded as 1-bit, 2bit, or 4-bit images. In these cases, Software automatically transforms the image into an 8-bit image the minimum bit depth for Software when opening the image file.

#### **2.3.2.3 Number of planes**

Number of planes corresponds to the number of arrays of pixels that compose the image. A grayscale or pseudo-color image is composed of one plane; while a true-color image is composed of three planes one each for the red component, blue component, and green component.

In true-color images, the color component intensities of a pixel are coded into three different values. A color image is the combination of three arrays of pixels corresponding to the red, green, and blue components in a Red Green Blue (RGB) image. Hue Saturation Luminescence (HSL) images are defined by their hue, saturation, and luminance values.

#### **2.3.3 Image types**

There are three types of images: grayscale, color, and complex images. For an identical spatial resolution, a color image occupies four times the memory space of an 8bit grayscale image, and a complex image occupies eight times the memory of an 8-bit grayscale image. These three types are described:

### 2.3.3.1 Grayscale images

Grayscale images are composed of a single plane of pixels. Each pixel is encoded using one of the following single numbers:





- An 8-bit unsigned integer representing grayscale values between 0 and 255
- A 16-bit signed integer representing grayscale values between  $-32,768$  and  $+32,767$
- A single-precision floating-point number, encoded using four bytes, represents grayscale values ranging from  $-\infty$  to  $\infty$ .

### 2.3.3.2 Color images

Color images are encoded in memory as a red, green, and blue (RGB) image. Color image pixels are a composite of four values. RGB images store color information using 8 bits each for the red, green, and blue planes. RGB 64 bit images store color information using 16 bits each for the red, green, and blue planes. In the RGB and HSL color models, an additional 8-bit value goes unused. This representation is known as 4- $\times$  8-bit or 32-bit encoding. In the RGB 64 bit color model, an additional 16-bit value goes unused. This representation is known as 4- $\times$  16-bit or 64-bit encoding.

### 2.3.3.3 Complex images

Complex image contains the frequency information of a grayscale image. We can create a complex image by applying a Fast Fourier transform (FFT) to a grayscale image. After we transform a grayscale image into a complex image, we can perform frequency domain operations on the image. Each pixel in a complex image is encoded as two single-precision floating-point values, which represent the real and imaginary components of the complex pixel. We can extract the following four components from a complex image: the real part, imaginary part, magnitude, and phase. Table 2.1 shows how many bytes per pixel grayscale, color, and complex images use [NI-c].

Image Type	Number of Bytes per Pixel Data
<b>8-bit (Unsigned) Integer Grayscale</b>  (1 byte or 8-bit)	 8-bit for the grayscale intensity
<b>16-bit (Signed) Integer Grayscale</b>  (2 bytes or 16-bit)	 16-bit for the grayscale intensity
<b>32-bit Floating- Point Grayscale</b>  (4 bytes or 32-bit)	 32-bit for the grayscale intensity
<b>RGB Color</b>  (4 bytes or 32-bit)	 8-bit for the alpha value (not used)      8-bit for the red intensity      8-bit for the green intensity      8-bit for the blue intensity

**Table 2.1: Bytes per pixel**

## 2.4 Functions of Machine Vision

Machine vision functions are the capabilities of machine vision systems – the type of information they extract from the image without regard to how that information is used. The functional categories are:

**Finding location and orientation** – This is an especially significant machine vision function. It is used to locate parts for operations for a variety of purposes.

**Measurements** – One capability inherent in machine vision that is not a capability of human vision is making measurements. Machine vision systems make measurements with a precision of one part in ten thousand. Human vision can only make rough comparative judgments on size.

**Flaw detection** – A flaw is a defect of unknown size, shape, and location. The most common use in industry of human visual inspection is to find flaws.

**Verification** – Vision systems verify the presence or absence of a part or feature. A vision system can tell if a cap has been installed on a bottle, or it can tell if all the surgical components are in a packaged kit.

**Identification** – Vision systems identify objects by performing optical character recognition (OCR) or by reading two-dimensional codes.

**Recognition** – Vision systems recognize parts based on their shape and other characteristics.

**Tracking** – Vision systems track the movement of parts through their field-of-view and, if desired, provide information so that robots can be guided to the moving part [[http: functions](#)].

## **2.5 Applications of Machine Vision**

Machine vision applications are categories describing how people use vision systems. The categories of applications are:

**Robot or machine guidance** – This ranges from using machine vision to locate a part and then repositioning the part or the robot to using machine vision to track the part's motion and maintain the part or the robot in position.

**Quality assurance** – This involves inspecting parts for conformance to requirements. The requirements may be to verify correctness of an assembly, to measure critical dimensions, or to insure the absence of objectionable flaws.

**Test and calibration** – Machine vision performs test and calibration by observing the positions of controls on an assembly or reading a display to verify the correct output.

**Real-time process control** – This is performed when machine vision is used to feed back or feed forward information it gets from imaging a part so that the process is held in control.

**Data collection** – This involves storing data gathered by the machine vision system as it observes parts coming through its field-of-view. Most often, data collection is used for statistical process control.

**Machine monitoring** – This involves using machine vision to insure a machine's correct operating condition. An example would be to insure that a drill bit is not broken on an automatic drilling machine.

**Material handling** – This is facilitated by machine vision when it identifies, recognizes, or routes products through a factory or warehouse.

**Sorting** – Machine vision sorts a variety of products ranging from nuts, berries, beans, and potatoes to screws and other fasteners.

**Counting** – Objects, such as those being mass conveyed, are easily counted by machine vision where other counting means do not work.

**Safety** – Machine vision has the potential to improve safety in the factory. To date, this has not been pursued commercially. However, advances being made in automotive design using embedded vision systems for highway safety may open the door to this application category.

The table 2.2 shows how the functional categories and the application categories of machine vision overlap. For any application there is more than one function that can serve it. And any function can be applied in more than one way [http: functions].

	<b>Tracking</b>	<b>Recognition</b>	<b>Identification</b>	<b>Verification</b>	<b>Flaw detection</b>	<b>Gauging</b>	<b>Finding position</b>
<b>Guidance</b>	✓	✓				✓	✓
<b>Quality assurance</b>				✓	✓	✓	
<b>Test and calibration</b>		✓		✓		✓	✓
<b>Process control</b>		✓	✓	✓	✓	✓	✓
<b>Data collection</b>		✓	✓	✓	✓	✓	✓
<b>Machine monitoring</b>				✓		✓	✓
<b>Material handling</b>	✓	✓	✓	✓		✓	✓
<b>Sorting</b>		✓	✓		✓	✓	
<b>Counting</b>		✓	✓			✓	
<b>Safety</b>	✓	✓		✓		✓	✓

Table 2.2: Functions and Applications of Machine Vision

## 2.6 Advantages and Limitations of Machine Vision

Machine vision can be implemented to various industrial locations where cameras can replace two or three workmen. It can also be employed in hazardous conditions where it is very difficult for operator to watch the process disturbances continuously. The decisions are taken by processor in real time therefore it is very useful in real time process control. The grabbing rate of some specialized cameras can be too high as compared to human perception therefore it can detect very fast and minute changes in case moving object.

System performance is affected due to vibration and noise because camera produces blurred image in these conditions. Natural light affect the camera performance due to changing of brightness of natural light from day to night. Sometimes the images that are heavily distorted may loose useful information during various processing functions. The accuracy of decision made by processor is simply dependent upon the resolution of camera. The more the resolution the finer will be the image [http: Machine-a].



## 2.7 Comparison between Machine Vision and Human Vision

The table 2.3 is showing the comparison between Machine Vision and Human Vision:

Feature	Machine Vision	Human Vision
Spectral range	Gamma rays to microwaves ( $10^{-11} - 10^{-1}$ m)	Visible light ( $4.10^{-7} - 7.10^{-7}$ m)
Spatial Resolution	Currently (2002) $4.10^6$ pixels (area scan, growing rapidly), 8192 (line-scan)	Effectively approximately 4000x4000 pixels
Sensor size	Small (approx. $5 \times 5 \times 15$ mm <sup>3</sup> )	Very large
Quantitative	Yes. Capable of precise measurement of size, area	No
Ability to cope with unseen events	Poor	Good
Performance on repetitive tasks	Good	Poor, due to fatigue and boredom
Intelligence	Low	High
Light level variability	Fixed, closely controlled	Highly variable
Light level (min)	Equivalent to cloudy moonless night	Quarter-moon light (greater if dark- adaptation is extended)
Strobe lighting and lasers	Possible (good screening is needed for safety)	Unsafe
Consistency	Good	Poor
Capital cost	Moderate	Low
Running cost	Low	High
Inspection cost, per unit	Low	High
Ability to “program” <i>in situ</i>	Limited. Special interfaces make task easier	Speech is effective
Able to cope with multiple views in space and/or time	Versatile	Limited

Able to work in toxic, biohazard areas	Yes	Not easily
Non-standard scanning methods	Line scan, circular scan, random scan, spiral-scan, radial scan	Not possible
Image storage	Good	Poor without photography or digital storage
Optical aids	Numerous available	Limited

**Table 2.3: Comparison between Machine Vision and Human Vision**

## 2.8 Comparison between Machine Vision and Computer Vision

The comparison between Machine Vision and Computer Vision as shown in table 2.4:

<b>Feature</b>	<b>Machine Vision</b>	<b>Computer Vision</b>
Motivation	Practical	Academic
Theoretical	Unlikely. (Practical issues are likely to dominate)	Yes. Many academic papers contain a lot of "deep" mathematics
Cost	Critical	Likely to be of secondary importance
Dedicated electronic hardware	Possibly needed to achieve high speed processing	No (by definition)
Use non-algorithmic solutions	Yes	There is a strong emphasis on proven algorithmic methods
<i>In situ</i> programming	Possible to accommodate new products	Unlikely
Data source	Human artifact, such as a piece of metal, plastic, glass, wood, etc.	Computer file. CV specialists are rarely able to control the image acquisition environment, or redefine the application to make it tractable

Models human vision	Most unlikely	Very likely
Most important criteria	(a) Easy to use (b) Cost-effective (c) Consistent & reliable (d) Fast	Performance judged in a specific way (e.g. accuracy of measurement, probability of recognising critical features, etc.)
Multi-disciplinary	Yes	No
Criterion for good solution	Satisfactory performance	Optimal performance
Nature of subject	Systems Engineering (pragmatic)	Mathematics / Computer Science (often academic / theoretical)
Human interaction	(a) For interactive prototyping system: experienced vision engineer.  (b) For target system in factory: low skill level during set-up. Autonomous in inspection mode.	Often relies on user having specialist skills (e.g. medicine, satellite imagery, forensic science, etc.)
Operator skill level required	(a) For interactive prototyping system: medium / high  (b) For target system in factory: must be able to cope with low skill level	May rely on user having specialist skills (e.g. medical)
Output data	Simple signal, to control external equipment (e.g. simple accept/reject device, or multi-axis robot)	Complex signal, for human being
Prime factor determining processing speed	(a) For interactive prototyping system: human interaction  (b) For target system in factory: speed of production	Human interaction. Speed is often of secondary importance

**Table 2.4: Comparison between Machine Vision and Computer Vision**

For acquiring images in real time we have to configure system according to our requirements. There are total four CDs to be installed in system. Drivers for Data Acquisition Instrument Control Motion and Vision are in two CDs, NI vision Development module are in one CD and IEEE-1394 PROSILICA camera 2.0.1 are in one CD. We used NI IEEE-1394 DCAM for capturing the image and NI Vision Assistant 7.1 for processing and analyzing it. NI CVS-1450 Series devices are easy-to-use, distributed, real-time imaging systems that acquire, process, and display images from IEEE 1394 camera. NI CVS-1450 acts as an interface between the camera and the development computer which is used for image processing and analysis. An Ethernet connection between the CVS-1450 device and a development computer allows you to display measurement results and status information and to configure the CVS-1450 device settings. When configured, the CVS-1450 device can run applications without a connection to the development computer. NI CVS-1450 series also provides multiple digital input/output (I/O) options for communicating with external devices to configure and start an inspection and to indicate results [NI-a].

### 3.1 Setup Overview

The sequence for setting up and getting started with the CVS-1450 device is:

**Setting up the Hardware**—this section explains how to connect a camera, monitor, and power supply to a CVS-1450 device.

**Setting up the Development Computer**—this section explains how to use either Vision Builder AI or LabVIEW Real-Time with the Vision Development Module to perform the following tasks: Connect the CVS-1450 device to the development computer and Install application and driver software.

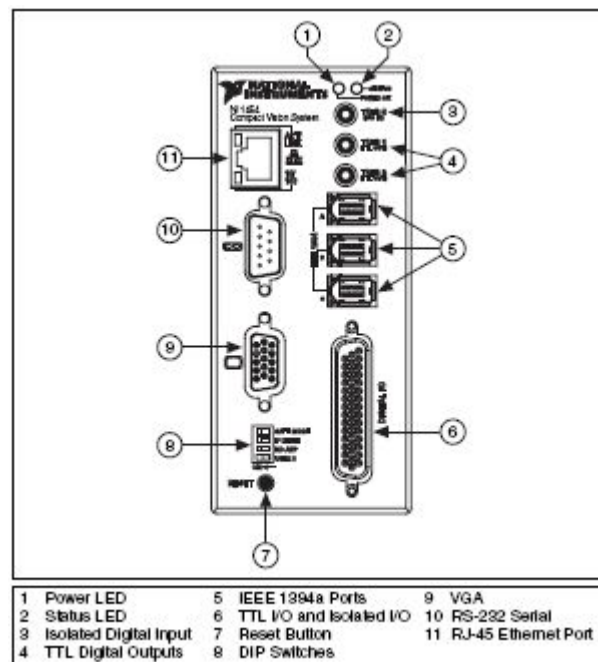
**Acquiring an Image**—this section explains how to use either Vision Builder AI or LabVIEW Real-Time with the Vision Development Module to acquire an image.

### 3.1.1 Setting up the Hardware

This section describes how to connect the basic hardware components of the CVS-1450 device. When these basic components are connected for the first time, the CVS-1450 device runs a program that acquires images. This program verifies that all hardware components are properly connected and functioning. The following items are necessary for hardware setup:

1. CVS-1450 device
2. 24 VDC  $\pm 10\%$ , 50 W power supply
3. DCAM-compliant IEEE 1394 camera
4. IEEE 1394 cable
5. Ethernet cable
6. Monitor

CVS-1450 hardware setup is shown in figure 3.1:



**Figure 3.1: CVS-1450 Series Front Panel**

### **3.1.1.1 Subnet Considerations**

To configure the CVS-1450 device, it must reside on the same subnet as the development computer. Once the CVS-1450 device is configured, other subnets can access and use it.

To use the CVS-1450 device on a subnet other than the one on which the development computer resides, first connect and configure the CVS-1450 device on the same subnet as the development computer. Next, physically move the CVS-1450 device to the other subnet and reassign an IP address. Contact network administrator for assistance in setting up the development computer and CVS-1450 device on the same subnet.

### **3.1.1.2 CVS-1450: Hardware**

The following items are necessary for setting up the CVS-1450 device.

1. CVS-1450 device
2. Ethernet-equipped development computer running Windows 2000/XP/Me/98
3. DCAM-compliant IEEE 1394 camera
4. Any standard IEEE 1394 cable—Facilitates plug-and-play connection from the CVS-1450 device to up to three 1394 cameras. To maintain signal integrity, the IEEE 1394 cable length must be no longer than 4.5 m.
5. NI desktop power supply (part number 778794-01) or any 24 VDC  $\pm 10\%$ , 50 W power supply
6. 10 m 10/100Base-T Ethernet cable —Standard Category 5 (CAT-5) 10/100Base-T Ethernet cable that connects the CVS-1450 device to a network port. To connect the CVS-1450 device directly to a local development computer, use an Ethernet crossover cable. To maintain signal integrity, the Ethernet cable length must be no longer than 100 m.

### **3.1.1.3 Connecting the CVS-1450 Device to a Network**

We use a standard CAT-5 or CAT-6 Ethernet cable to connect the CVS-1450 device to an Ethernet network. If the development computer is already configured on a network, we must configure the CVS-1450 device on the same network.

If the development computer is not connected to a network, we can connect the two directly using a CAT-5 or CAT-6 Ethernet crossover cable.

#### 3.1.1.4 Connecting a Camera and Monitor to the CVS-1450 Device

Before connecting a camera and monitor to the CVS-1450 device, we have sure that all CVS-1450 device DIP switches are in the **OFF** position.

To connect an IEEE 1394 camera and a monitor to the CVS-1450 device, refer to figure 3.2 while completing the following steps:

1. Connect the VGA cable from the monitor to the VGA port on the CVS-1450 device.
2. Plug the IEEE 1394 cable into one of the IEEE 1394a ports on the CVS-1450 device. Plug the other end of the cable into the IEEE 1394 port on the camera.

If our camera requires an external power supply, connect it to the camera, and verify that the camera is powered on.

3. Plug in and power on the monitor.

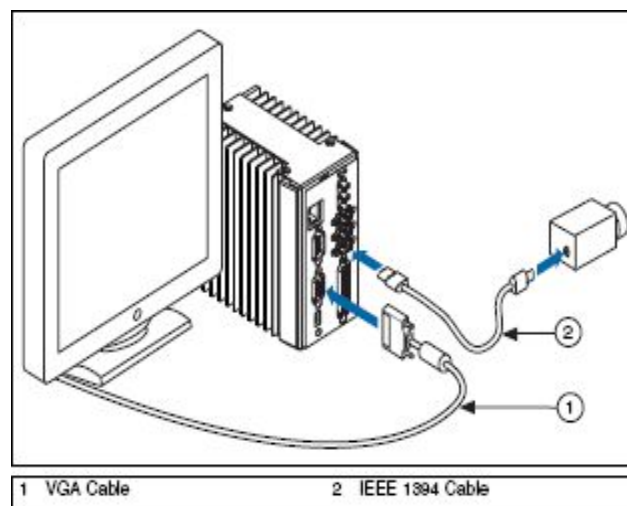
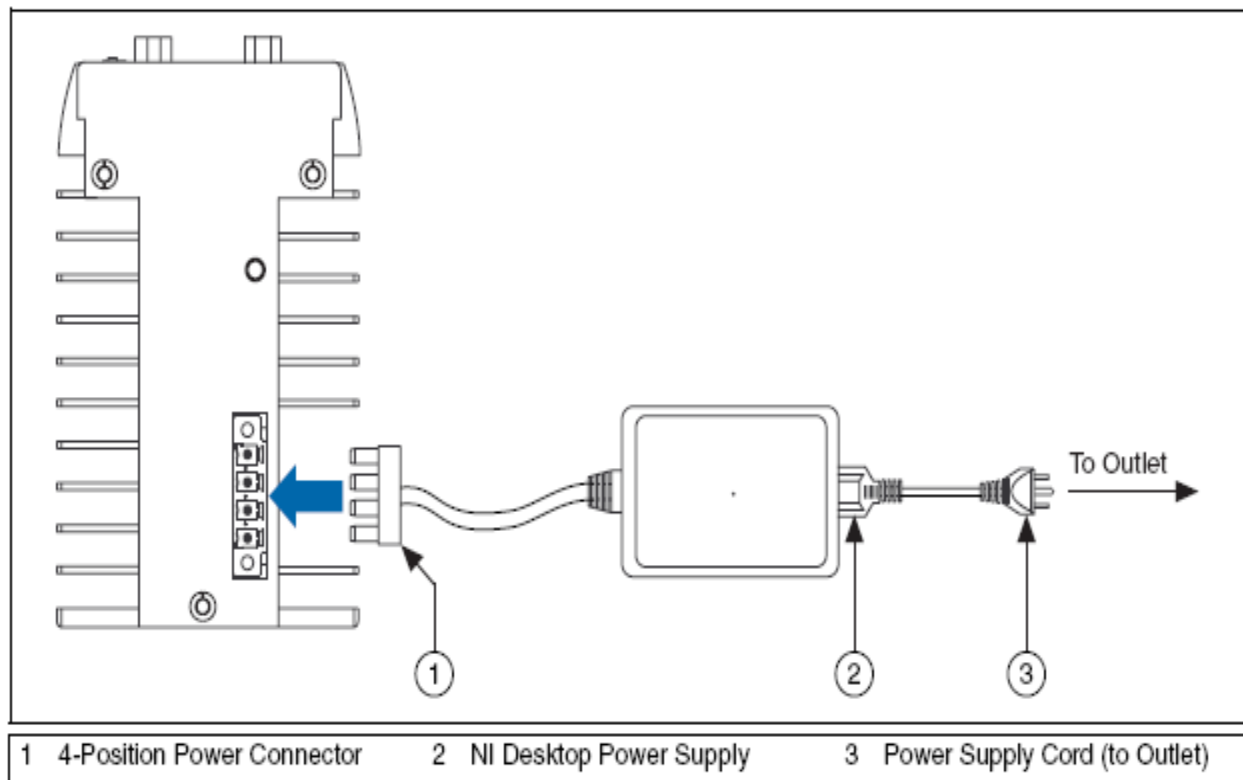


Figure 3.2: Basic Hardware

### 3.1.1.5 Wiring Power to the CVS-1450 Device

Connect the power to the CVS-1450 device, as described in figure 3.3 while completing the following steps:

1. Plug the 4-position connector from the power supply into the power receptacle on the CVS-1450 device.
2. Plug the power cord into the power supply.
3. Plug the power cord into an outlet.



**Figure 3.3 Wiring Power to the CVS-1450 Device**

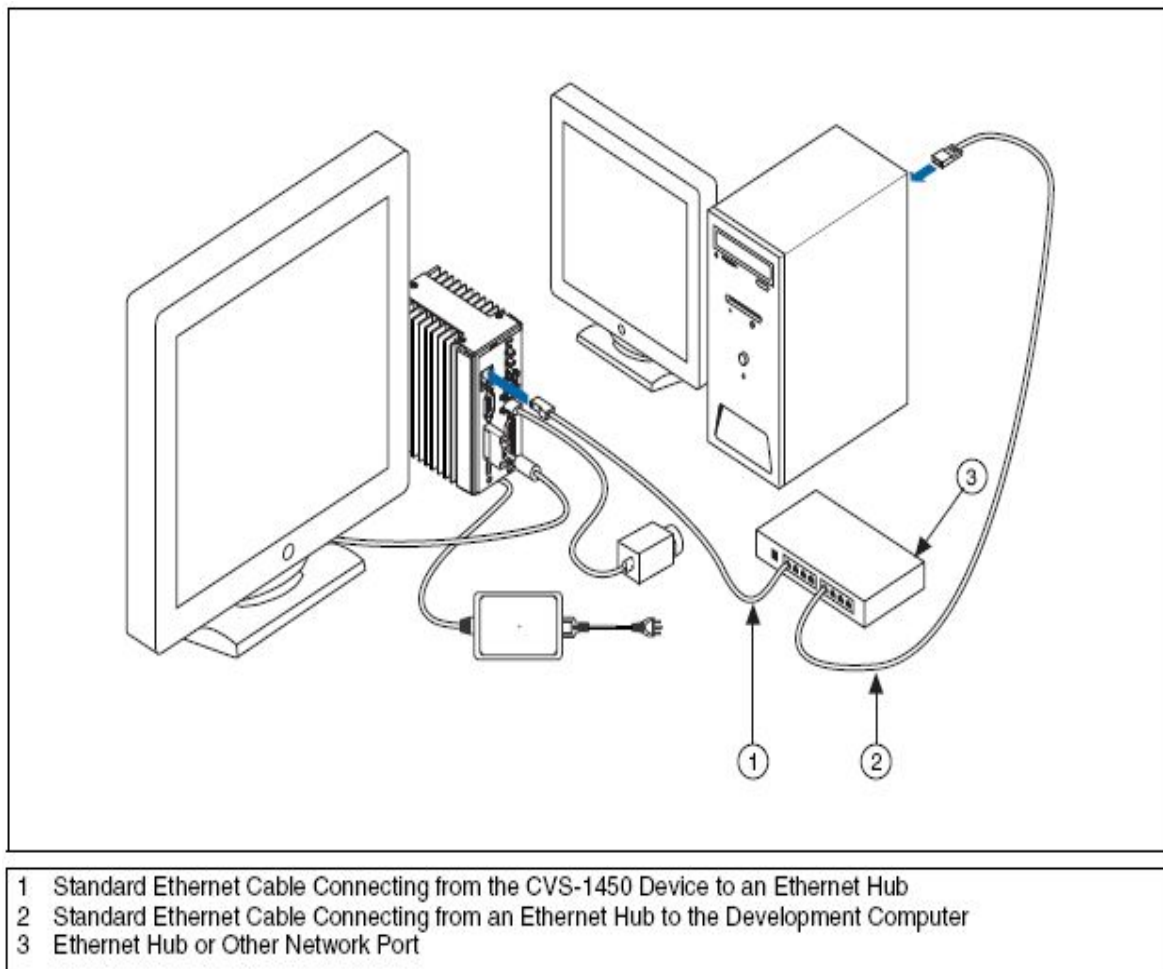
Do not connect the CVS-1450 device main power to a source other than 24 VDC  $\pm 10\%$ . Do not connect the CVS-1450 device isolated power to a source less than 5 VDC or greater than 30 VDC. Doing so could damage the CVS-1450 device.



### 3.1.1.6 Connecting the CVS-1450 Device to the Development Computer

To connect the CVS-1450 device to the development computer, refer to figure 3.4 while completing the following steps:

1. Verify that the development computer is connected to the network and is powered on.
2. Using a standard CAT-5 Ethernet cable, connect from the network port to the Ethernet port on the CVS-1450 device.
3. Using a standard CAT-5 Ethernet cable, connect from the network port to the Ethernet port on the development computer.



**Figure 3.4: Ethernet Connection**

The development computer communicates with the CVS-1450 device over an Ethernet connection. Use a standard Ethernet cable to connect from the network port to the CVS-1450 device. If CVS-1450 is not connecting through a network, we use an Ethernet crossover cable to connect the CVS-1450 device directly to the development computer.

### 3.1.2 Setting up the Development Computer

We must install LabVIEW, LabVIEW Real-Time, and the Vision Development Module software before installing the NI-IMAQ FOR IEEE 1394 Cameras driver software.

We can complete the following steps to install LabVIEW, LabVIEW Real-Time, the Vision Development Module and NI-IMAQ for IEEE 1394 Cameras into the development computer.

1. Insert the LabVIEW CD into the CD-ROM drive.
2. When the installation splash screen appears, click **Install** LabVIEW and follow the setup instructions.
3. Insert the LabVIEW Real-Time CD into the CD-ROM drive.
4. When the installation splash screen appears, click **Install** LabVIEW real time and follow the setup instructions.
5. Insert the Vision Development Module CD into the CD-ROM drive.
6. When the installation splash screen appears, click **Install** Vision Development Module and follow the setup instructions.
7. Insert the NI-IMAQ for IEEE 1394 Cameras CD into the CD-ROM drive.
8. When the installation splash screen appears, click **Install NI-IMAQ for IEEE 1394 Cameras**, and follow the setup instructions.
9. When prompted, click yes to reboot the development computer.

### **3.1.3 Acquiring an Image**

We use NI-IMAQ to configure IEEE-1394 camera which gives us the ability to use IEEE-1394 industrial digital video cameras to acquire images. The camera configuration is saved in a camera file. The camera may operate at various resolutions and frame rates depending on camera capabilities.

#### **Configuration**

When the camera is successfully connected, we can configure the camera. Complete the following steps to configure the IEEE-1394 camera for a grab acquisition:

1. Connect the camera to the IEEE-1394 PORT on the computer.
2. Expand NI-IMAQ IEEE1394 Devices to obtain a list of available cameras.
3. Click the camera name to select the appropriate camera.
4. Click Snap to acquire a single image or Grab to acquire images continuously while focusing the camera, click Grab again to stop the acquisition.
5. Acquisition Parameters and Camera Attributes tabs to modify the camera video parameters.
6. Click Snap after adjusting the parameters to view
7. Click Save to save the current configuration.

### **3.2 IMAQ for IEEE 1394 (Fire wire) compatible cameras**

The NI-IMAQ for IEEE 1394 compatible cameras software, lists the supported application development environments describes the fundamentals of creating applications using NI-IMAQ for IEEE 1394 Cameras, describes the files used to build these applications. NI-IMAQ Software for IEEE 1394 Cameras gives us the ability to use IEEE-1394 industrial digital video cameras to acquire images. The cameras may operate at various resolutions and frame rates, depending on camera capabilities.

We use National Instruments Measurement & Automation Explorer (MAX) to configure our IEEE-1394 camera and Refer to the NI-IMAQ for IEEE 1394 Cameras help for information about configuring our IEEE-1394 camera. The camera configuration is saved in a camera file, which the NI-IMAQ for IEEE-1394 Cameras VIs and functions use to configure a camera and supported attributes. The NI-IMAQ for IEEE-1394 cameras application programming interface (API) is divided two main function groups: high-level and low-level [NI-a].

### 3.2.1 High-level functions

Use to capture images quickly and easily. If we need more advanced functionality, we can mix high-level functions with low-level functions.

**Snap functions** capture all or a portion of a single image to the user buffer.

**Grab functions** perform an acquisition that loops continually on one or more internal buffers. We can copy the last acquired buffer to a separate user buffer for processing or analysis.

**Sequence functions** acquire a specified number of internal buffers and then stops.

**Trigger functions** control the trigger mode of the IEEE-1394 camera.

### 3.2.2 Low-level functions

Use when we require more direct control of the image acquisition.

**Acquisition functions** configure, start, stop, and unconfigure an image acquisition, or examine a user buffer during an acquisition.

**Attribute functions** examine and change the acquisition or camera attributes.

**Utility functions** display an image in a window, save an image to a file, or to get detailed error information.

Both high-level and low-level functions support snaps, grab, sequence, and triggered acquisitions. Using high-level functions, we can write programs quickly without having to learn

the details of the low-level API and driver. The low-level functions give us finer granularity and control over the image acquisition process, but we must understand the API and driver in greater detail to use these functions.

### 3.3 Acquisition flow

The basic steps of performing an acquisition with the NI-IMAQ for IEEE 1394 cameras software are initialization, configuration, and acquisition [NI-a].

#### 3.3.1 Initialization

To acquire images using the high-level or low-level functions, we first must initialize a camera session. A camera session is a process-safe handle to an IEEE 1394 camera. The driver uses a camera session to identify the camera to which further NI-IMAQ for IEEE 1394 Cameras functions apply. We can simultaneously open as many camera sessions as there are cameras connected to our system. When initializing the camera session, we need to specify two parameters: Camera name and Camera control mode.

##### Camera name

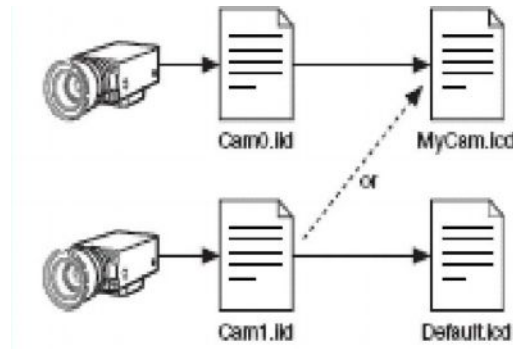
NI-IMAQ for IEEE 1394 Cameras references all camera sessions by a name. The driver creates default names for each camera in our system in the order that the cameras are connected. The names observe the convention shown in Table 3.1.

Camera Name	IEEE 1394 Camera Installed
Cam0	Device 1
Cam1	Device 2
Cam.....n	Device.....n

**Table 3.1: Camera naming conventions**

**Interface files** store information about which physical camera is associated with a camera name. Only a single camera can use each interface file.

**Camera files** store all the configurable attributes. Camera files can be shared between identical cameras. Use MAX to configure the default state of a particular camera. Figure 3.5 shows the relationship between cameras, interface files, and camera files.



**Figure 3.5: Relationships between cameras, interface files, and camera files**

Use the Enumerate function to query the number and names of available cameras. When we open a camera session with the Initialize function, the camera with the unique serial number described by the interface file camn.iid opens, where n is the reference to the camera.

If the camera is not present and a camera of the same make and model is present, as described in the interface file, the driver opens the available camera. The interface file updates to use the new camera. The camera file described by the interface file opens, and all the user attributes are set in the driver. If no camera of the same make and model is present, the Initialize function returns an error.

### **Camera control mode**

The camera control mode parameter has two options: controller and listener. The default option controller controls the camera and receives video data. The listener only receives video data. We use the listener option in broadcasting applications.

### **3.3.2 Configuration**

After initializing the interface, configure the interface for acquisition by specifying the following parameters: whether the acquisition is one-shot or continuous, the number of internal buffers to use, and the region of interest for the acquisition. During configuration, the driver validates all the user-configurable attributes. If any attributes are invalid or out of range, the driver returns an error and does not configure the acquisition. If we want to reconfigure the acquisition, call the Clear Acquisition function before calling the configure function again.

#### **One-shot/Continuous acquisition**

We use a one-shot acquisition to start an acquisition, perform the acquisition, and stop the acquisition using a single function. The number of images acquired is equal to the number of images in the images collection. With a one-shot acquisition, we specify a certain number of internal buffers. The camera transfers each image up to and including the specified number of buffers. The driver acquires every image during a one-shot acquisition. National Instruments recommends one-shot acquisition for applications that do not require real-time acquisition or processing.

#### **Continues acquisition**

We use a continuous acquisition to start an acquisition, continuously acquire images into the internal buffers, and explicitly stop the acquisition. With continuous acquisition, the driver acquires video data continuously from the camera and enables us to examine the most current buffer. National Instruments recommends continuous acquisition for real-time acquisition and processing.

#### **Number of buffers**

Another aspect of configuration is specifying the number of internal buffers into which we want to acquire image data. During configuration, buffers are allocated from system memory and page-locked. Once the acquisition starts, the camera transfers video data over the IEEE-1394 bus to the IEEE-1394 interface card FIFO. Then, video data is directly transferred to the internal

buffer. This transfer requires negligible CPU resources. Each internal buffer we allocate is the exact size of the raw data being transmitted by the camera. For continuous acquisitions, allocate three or more buffers. Allocating a single buffer for a continuous acquisition may result in a high number of lost images. For one-shot acquisitions, specify the number of buffers that the application requires. For example, if the application runs for two seconds, and the camera acquires at 30 frames per second, allocate 60 buffers to capture each image.

### **Region of interest**

The region of interest (ROI) specifies a rectangular portion of the image to be captured. In Partial Image Size Format (Format 7) video modes, the ROI defines the portion of the image to transfer from the camera to system memory. In non-Format 7 video modes, the entire image is transferred from the camera to system memory. In all video modes, the ROI specifies the amount of data decoded by the driver while acquiring into a user buffer. By default, the driver transfers the entire image. We specify a smaller ROI for the following reasons:

- To acquire only the necessary subset of data
- To increase the acquisition speed by reducing the amount of data transferred and/or decoded
- To allow for multiple simultaneous acquisitions by reducing bandwidth usage

### **3.3.3 Acquisition**

After configuring and starting our acquisition, the camera sends data to the internal buffers. To process the acquired image data, we must copy the data from the internal buffer into our user buffer.

### **User buffer**

Before starting the acquisition, we must allocate a user buffer in addition to configuring internal buffers. The driver copies or decodes image data from the internal buffer into the user buffer during acquisition. Then, process and analyze the image in the user buffer. When acquiring data



into an IMAQ Vision image, the driver resizes and casts the image as needed. However, if we acquire data into a user buffer, we must allocate enough space for one decoded image.

A buffer number is a zero-based index that represents the cumulated transferred image count. For example, during a continuous acquisition with three internal buffers, the buffer number is updated as follows: 0, 1, 2, 3, 4, 5, and so on. Buffer numbers 0 and 3 refer to the same internal buffer in the buffer ring. For a one-shot acquisition, we can request only one of the available buffer numbers. For a continuous acquisition, we can request any present or future buffer number. We can also request the next logical buffer or the buffer containing the most recently acquired data. With high-level grab acquisitions, the buffer number defaults to the next transferred buffer. When we complete the buffer acquisition step, the driver returns the actual buffer number with the image.

### **Overwrite mode**

A continuous acquisition acquires and processes every image that is transferred from the camera. However, because of processing time fluctuations, some images from the camera may not be processed before the camera transfers the next image. Using multiple internal buffers in a continuous acquisition allows for a small amount of jitter. However, if a delay becomes too long, the camera overwrites the requested buffer with new image data. NI-IMAQ for IEEE-1394 Cameras is able to detect overwritten internal buffers. We can configure the driver to manage an overwritten buffer in one of the following ways:

- Get newest valid buffer
- Get oldest valid buffer
- Fail and return an error

In all cases, the camera continues to transfer data when a buffer is overwritten. The default overwrite mode for all types of acquisition is to get the newest valid buffer. This option, which National Instruments recommends for most applications, enables us to process the most recent image. If we need to get the image closest in time to a requested buffer, configure the driver to

get the oldest valid buffer. If our application requires that every image be processed, configure the driver to fail when a buffer is overwritten so that we are alerted.

## Timeouts

A timeout is the length of time, in milliseconds, that the driver waits for an image from the camera before returning an error. A timeout error usually occurs if the camera has been removed from the system or when the camera did not receive an external trigger signal.

## Decoding

Except for 8-bit monochrome images, all video modes require decoding before we can interpret the image data. For example, many color IEEE 1394 cameras output images of type YUV 4:2:2. However, IMAQ Vision does not natively support the YUV mode. To process and display the image, the driver automatically decodes the YUV image into a 32-bit RGB image. Table 3.2 lists common video modes and their corresponding image types after being decoded by NI-IMAQ for IEEE 1394 Cameras.

Raw Camera Output	Decoded Destination Image
8-bit monochrome	8-bit monochrome
16-bit monochrome	16-bit monochrome
YUV 4:1:1	32-bit color
YUV 4:2:2	32-bit color
YUV 4:4:4	32-bit color
24-bit RGB	32-bit color
48-bit RGB	64-bit color
8-bit Bayer	32-bit color
16-bit Bayer	32-bit color

**Table 3.2: Decoder inputs and corresponding outputs**

#### 4.1 Introduction to Vision Assistant 7.1

Vision Assistant is a tool for prototyping and testing image processing applications. To prototype an image processing application, build custom algorithms with the Vision Assistant scripting feature. The scripting feature records every step of the processing algorithm. After completing the algorithm, we can test it on other images to make sure it works. The algorithm is recorded in a Builder file, which is an ASCII text file that lists the processing functions and relevant parameters for an algorithm that we prototype in Vision Assistant. We must have LabVIEW 6.1 or later and IMAQ Vision 7.1 for LabVIEW or later installed to use the LabVIEW VI Creation Wizard.

#### 4.2 System requirements and installation

To run Vision Assistant, system must meet the following minimum requirements

1. Personal computer using a 233 MHz Pentium-class processor. Using a Pentium III or Celeron 600 MHz or equivalent is recommended.
2. Microsoft Windows 2000/NT/XP. If we are using Windows NT 4.0, we must have Service Pack 6 or later installed to run Vision Assistant.
3. 1024, 768 resolution or higher video adapter; 65,536 colors, 16-bit or higher.
4. Minimum of 128 MB RAM; 256 MB recommended.
5. Minimum of 200 MB of free hard disk space.
6. If we are acquiring images, the system must have National Instruments image acquisition (IMAQ) hardware and NI-IMAQ 3.0 or later or NI-IMAQ for IEEE 1394 Cameras 1.5 or later installed.

### 4.2.1 Installing Vision Assistant

- To install Vision Assistant on a Windows 2000/NT/XP system, we must be logged in with Administrator privileges.
- Insert the Vision Assistant CD into the CD-ROM drive.
- If we do not have auto run enabled, double-click auto run. exe. If we have auto run enabled, auto run. exe runs automatically.

### 4.2.2 Features

Vision Assistant offers the following features:

**Script window:** It Records a series of image processing steps and the settings we use for each of those steps. We can run scripts on single images or in a batch to analyze a collection of images. We also can modify and save scripts.

**Image browser:** It contains all of the images currently loaded in Vision Assistant. We can select an image to process by double-clicking it in the Image Browser.

**Processing window:** It updates the image as we change parameters. Because this view immediately reflects the changes we have made in the Parameter window, we can continue modifying parameters until we get the result we want.

**Functions window/Parameter window:** It displays a list of image processing functions we can use to develop an algorithm, or displays parameters that we can set for an image processing function. Each function available through the Functions window has a Parameter window in which we set the parameters for that function.

**Reference window/Embedded Help window:** The Image tab of the Reference window displays the image source as we manipulate it in the Processing window. The other tabs in the Embedded Help window contain context help for the function we are using.

**Solution wizard:** It displays a list of industries and corresponding quality-assurance tasks that those industries perform. The wizard loads an IMAQ Vision-based solution for the task we

select.

**Performance meter:** It estimates how long a script will take to complete on a given image.

**LabVIEW VI creation:** It creates a LabVIEW VI corresponding to the algorithm we prototype in Vision Assistant. Based on the options we select, the LabVIEW VI Creation Wizard creates a new VI that implements the image processing steps of the current script or of a saved script file.

**C Code creation:** It creates a C file corresponding to the algorithm we prototype in Vision Assistant. Based on the options we select, the C Code Creation Wizard creates a C function that implements the image processing steps of the current script.

**Builder file:** ASCII text file that lists the C and Microsoft Visual Basic functions and parameters for the algorithm we prototyped in Vision Assistant.

## 4.3 Processing functions of NI Vision Assistant

### 4.3.1 Image functions

The image functions that analyze the content of an image are included in this group such as:

**Histogram** counts the total number of pixels in each grayscale value and graphs it.

**Line Profile** displays the grayscale distribution along a line of pixels in an image.

**Measure** calculates measurement statistics associated with a region of interest in the image.

**3D View** displays the light intensity of an image in a three-dimensional coordinate system.

**Image Mask** builds a mask from an entire image or an ROI.

**Geometry** modifies the geometrical representation of an image.

**Image Buffer** stores and retrieves images from buffers.

**Get New Image** opens a new image from the script.

**Calibrate Image** calibrates an image to perform measurements in real-world units.

**Calibrate from Image** applies the calibration information in an image file to the current image.

**Image Correction** transforms a distorted image acquired in a calibrated setup by correcting perspective errors and lens distortion.

#### **4.3.2 Color functions:**

The image functions that analyze color images are included in this group such as:

**Color Operators** performs arithmetic and logical operations on color images.

**Extract Color Planes** extracts the RGB, HSV, and HSL planes from an image.

**Color Threshold** applies a threshold to the three planes of a color image and places the result in an 8-bit image.

**Color Location** locates colors in a color image.

**Color Pattern Matching** checks the presence of a template file in the entire color image or in an ROI.

**Color Matching** learns the color content of an ROI in an image and compares it to the color content of another ROI.

#### **4.3.3 Grayscale Functions**

The image functions that analyze grayscale images are included in this group such as:

**Lookup Table** improves contrast and brightness in an image by applying a lookup table.

**Filters** prepare an image for processing so we can extract only the information you need from the image.

**Gray Morphology** modifies the shape of objects in an image.

**FFT Filter** applies a frequency filter to the image.

**Operators** perform arithmetic and logical operations on an image.

**Conversion** converts the current image to the specific image type.

**Threshold** selects ranges of pixels in grayscale images.

**Quantify** quantifies the content of an image or ROIs within the image.

**Centroid** calculates the energy center of the image

#### **4.3.4 Binary Functions**

The image functions that analyze binary images are included in this group such as:

**Basic Morphology** modifies the shape of binary objects in an image.

**Adv. Morphology** performs high-level operations on particles in binary images.

**Particle Filter** removes or keeps particles in an image as specified by the filter criteria.

**Invert Binary Image** reverses the dynamic of an image that contains two different grayscale populations.

**Shape Matching** searches an image for objects that are shaped similarly to a template object.

**Particle Analysis** displays measurement results for selected particle measurements that are performed on the image.

**Circle Detection** finds the center and radius of circular particles in an image.

#### **4.3.5 Machine Vision Functions**

Vision Assistant provides the following machine vision functions:

1. Detecting the presence or absence of parts in an image
2. Measuring the dimension of parts to determine if they meet specifications
3. Locating objects in an image
4. Reading 1D and 2D barcodes
5. Reading text in an image
6. Classifying samples in an image

The image functions that perform the following common machine vision inspection tasks are included in this group such as:

**Edge Detection** finds edges along a line that we draw with the Line Tool from the toolbar.

**Find Straight Edge** finds points within the edge of an object and then finds a line describing the edge.

**Find Circular Edge** locates the intersection points between a set of search lines within a circular area (annulus), and then finds the best fit circle.

**Clamp** finds edges within a rectangular ROI drawn in the image and measures the distance between the first and last edge.

**Match Pattern** locates regions of a grayscale image that match a predetermined template. Pattern Matching can find template matches regardless of poor lighting, blur, noise, shifting of the template, and rotation of the template.

**Caliper** computes measurements—such as distances, areas, and angles—based on results returned from other machine vision and image processing functions.

**Read 1D Barcode** reads values encoded in 1D barcodes.

**Read Data Matrix Code** reads values encoded in a Data Matrix code.

**Read PDF417 Code** reads values encoded in a PDF417 code.

**OCR** reads characters in a region of an image.



**Classification** classifies samples in an image

We used mainly two techniques in our work i.e. **Particle Analysis and Threshold**

#### **4.4 Particle Analysis**

Particle analysis consists of a series of processing operations and analysis functions that produce some information about the particles in an image. A particle is a contiguous region of nonzero pixels. We can extract particles from a grayscale image by thresholding the image into background and foreground states. Zero valued pixels are in the background state, and all nonzero valued pixels are in the foreground. In a binary image, the background pixels are zero, and every non-zero pixel is part of a binary object.

We perform a particle analysis to detect connected regions or groupings of pixels in an image and then make selected measurements of those regions. Using particle analysis, we can detect and analyze any two-dimensional shape in an image [NI-c].

##### **4.4.1 Applications**

Particle analysis can be used in the following applications:

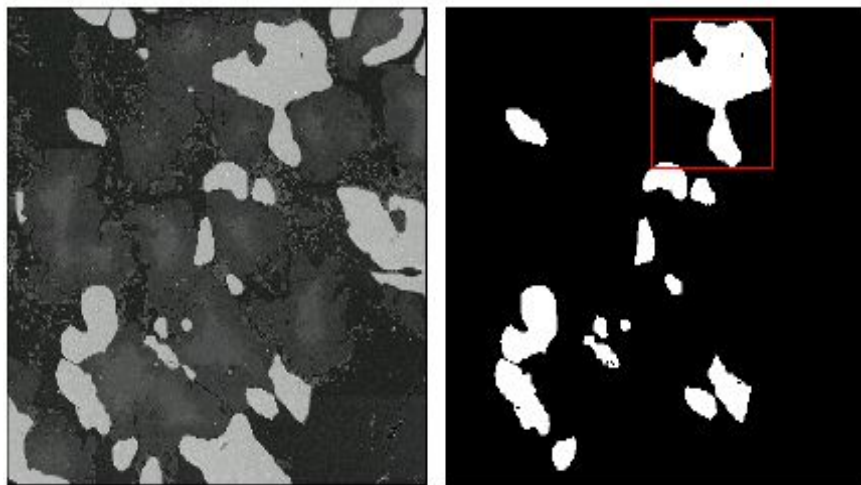
1. When we are interested in finding particles whose spatial characteristics satisfy certain criteria.
2. In many applications where computation is time-consuming, we can use particle filtering to eliminate particles that are of no interest based on their spatial characteristics, and keep only the relevant particles for further analysis.
3. We can use particle analysis to find statistical information such as the presence of particles, their number and size, and location.
4. We also can locate objects in motion control applications.
5. In applications where there is a significant variance in the shape or orientation of an object, particle analysis is a powerful and flexible way to search for the object.

6. We can use a combination of the measurements obtained through particle analysis to define a feature set that uniquely defines the shape of the object.
7. Machine vision inspection tasks such as detecting flaws on silicon wafers, detecting soldering defects on electronic boards
8. Web inspection applications such as finding structural defects on wood planks or detecting cracks on plastics sheets.

#### 4.4.2 Concepts

A typical particle analysis process scans through an entire image, detects all the particles in the image, and builds a detailed report on each particle. For example, we could use the area of the template particle as a criterion for removing all particles that do not match it within some tolerance. We then can perform a more refined search on the remaining particles using another list of parameter tolerances.

The figure 4.1 shows a sample list of parameters that we can obtain in a particle analysis application. The binary image in this example was obtained by thresholding the source image and removing particles that touch the border of the image. We can use these parameters to identify and classify particles.



**Figure 4.1: Sample list of parameters**

We can use multiple parameters such as perimeter, angle, area, and center of mass to identify and classify these particles. Using multiple parameters can be faster and more effective than pattern matching in many applications. Also, by using different sets of parameters, we can uniquely identify a feature in an image.

The following table shows the values obtained for the particle enclosed in a rectangle, shown in table 4.1:

<b>Particle Measurement</b>	<b>Values</b>
Area	2456
Number of Holes	1
Bounding Rect	
Left	127
Top	8
Right	200
Bottom	86
Center of Mass	
X	167.51
Y	37.61
Orientation	82.36°
Dimensions	
Width	73
Height	78

**Table 4.1: Values of the Particle**

To use particle analysis, first create a binary image using a thresholding process. We then can improve the binary image using morphological transformations and make measurements on the particles in the image.

## 4.5 Thresholding

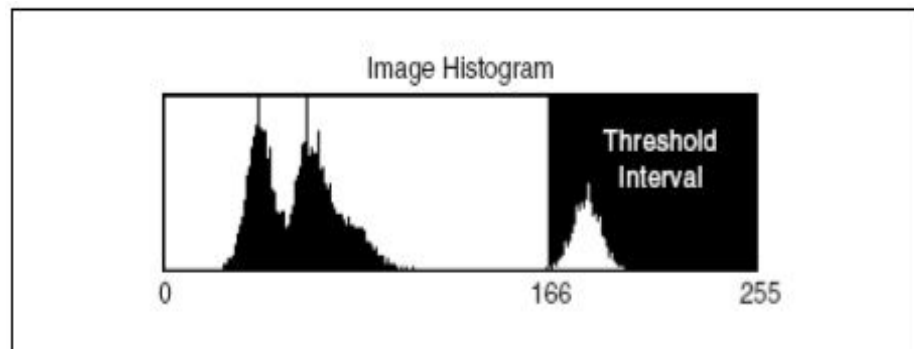
Threshold consists of segmenting an image into two regions: a particle region and a background region. This process works by setting to 1 all pixels that belong to a gray-level interval, called the threshold interval, and setting all other pixels in the image to 0.

Use thresholding to isolate objects of interest in an image. Thresholding converts the image from a grayscale image, with pixel values ranging from 0 to 255, to a binary image, with pixel values of 0 or 1.

Thresholding enables to select ranges of pixel values in grayscale and color images that separate the objects under consideration from the background [NI-c].

### 4.5.1 Applications

Use thresholding to extract areas that correspond to significant structures in an image and to focus the analysis on these areas as shown in figure 4.2:



**Figure 4.2: Image Histogram**

Pixels outside the threshold interval are set to 0 and are considered as part of the background area. Pixels inside the threshold interval are set to 1 and are considered as part of a particle area.

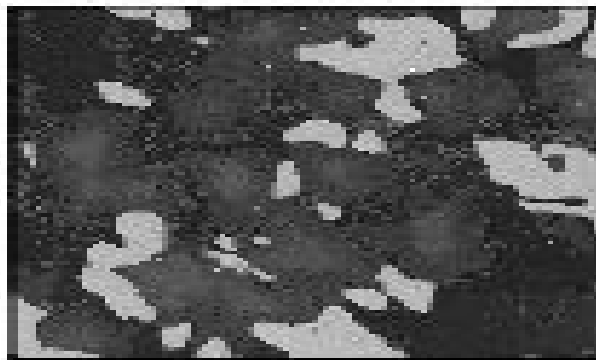
### 4.5.2 Intensity Threshold

Particles are characterized by an intensity range. They are composed of pixels with gray-level values belonging to a given threshold interval (overall luminosity or gray shade). All other pixels are considered to be part of the background.

The threshold interval is defined by the two parameters—**Lower Threshold** and **Upper Threshold**. All pixels that have gray-level values equal to or greater than the **Lower Threshold** and equal to or smaller than the **Upper Threshold** are selected as pixels belonging to particles in the image.

#### 4.5.3 Thresholding Example

This example uses the following source image.



**Figure 4.3: Example of Thresholding**

Highlighting the pixels that belong to the threshold interval [166, 255] (the brightest areas) produces the following image.



**Figure 4.4: Brightest area of Thresholding**

A critical and frequent problem in segmenting an image into particle and background regions occurs when the boundaries are not sharply demarcated. In such a case, the choice of a correct threshold becomes subjective. Therefore, we may want to enhance our images before thresholding to outline where the correct borders lie. We can use lookup tables, filters, FFTs, or equalize functions to enhance images. Observing the intensity profile of a line crossing a boundary area is also helpful in selecting a correct threshold value. Finally, morphological transformations can help us to retouch the shape of binary particles and therefore correct unsatisfactory selections that occurred during the thresholding.

#### **4.5.4 Automatic Threshold**

Various automatic thresholding techniques are available.

Clustering

Entropy

Interclass Variance

Metric

Moments

In contrast to manual thresholding, these methods do not require that we set the minimum and maximum light intensities. These techniques are well suited for conditions in which the light intensity varies. Depending on our source image, it is sometimes useful to invert the original grayscale image before applying an automatic threshold function, such as moments and entropy.

This is especially true for cases in which the background is brighter than the foreground. Clustering is the only multi-class thresholding method available. Clustering operates on multiple classes so we can create tertiary or higher-level images. The other four methods—entropy, metric, moments, and interclass variance—are reserved for strictly binary thresholding techniques. The choice of which algorithm to apply depends on the type of image to threshold.

## **Clustering**

This technique sorts the histogram of the image within a discrete number of classes corresponding to the number of phases perceived in an image. The gray values are determined, and a barycenter is determined for each class. This process repeats until it obtains a value that represent the center of mass for each phase or class. Clustering, also known as multi-class threshold which is the most frequently used automatic thresholding method.

## **Entropy**

Based on a classical image analysis technique, entropy is best for detecting particles that are present in minuscule proportions on the image. For example, this function would be suitable for fault detection.

## **Interclass Variance**

Interclass variance is based on discriminant analysis. An optimal threshold is determined by maximizing the between-class variation with respect to the threshold.

## **Metric**

this technique is used in situations similar to interclass variance. For each threshold, a value is calculated that is determined by the surfaces representing the initial gray scale. The optimal threshold corresponds to the smallest value.

## **Moments**

This technique is suited for images that have poor contrast. The moments method is based on the hypothesis that the observed image is a blurred version of the theoretically binary original. The blurring that is produced from the acquisition process, caused by electronic noise or slight defocalization, is treated as if the statistical moments of average and variance were the same for both the blurred image and the original image. This function recalculates a theoretical binary image.

### Problem Statement and Solution

---

#### 5.1 Problem Statement

Bacterial colony is a group of bacteria growing on a plate that is derived from one original starting cell. Bacterial colony enumeration has applications in many different assays such as antibiotic screening, toxicology testing, and genotoxicity measuring. The number of microorganisms present in food has very important consideration. Bacterial colony counting process is usually performed by well-trained technicians manually. However, there might exist hundreds of colonies in a traditional 100mm Petri dish. Therefore, this manual enumeration process has a very low throughput and is time consuming and labor intensive in practice. In addition, the manual counting is an error-prone process since the counting results of the same plate obtained from different technicians might vary, especially when a vast number of colonies appear on the plate. Another possible cause of variation is the judgment of the indistinguishable colony overlaps. Thus, it is important to have consistent criteria for measuring overlapped colonies. To produce consistent and accurate results and improve the throughput, we have worked on the Bacteria-colony counter using Machine Vision.

#### 5.2 Proposed area of research

To overcome the problems related to manual counting of bacteria colony, we propose to automate this counting process by machine vision. Image of Petri dish is acquired using IEEE-1394 digital camera Prosilica 2.0.1 and processed using workstation Compact vision system-1450(CVS-1450).

Machine Vision is used because the images so acquired may be blurred and noisy and need to be processed to obtain better images. Threshold technique can easily select the ranges of pixel values in grayscale image. Particle Analysis technique works very efficiently for counting the



number of Bacteria in a growth medium. We perform a particle analysis to detect connected regions or groupings of pixels in an image and then make selected measurements of those regions. Using particle analysis, we can detect and analyze any two-dimensional shape in an image. Therefore, we dedicate our work to this area.

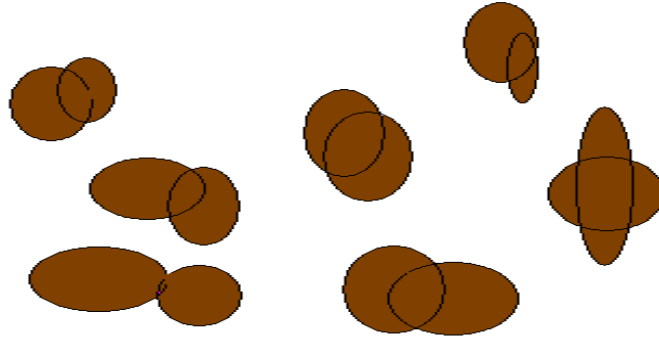
### **5.3 Problem solution**

To find the number of bacteria, we perform the following steps:

1. Acquire image of all Petri dishes to be counted in a batch by appropriate file name. Then each image so acquired is analyzed by the following steps
2. Extract one of the three color planes (RGB, HSV, or HSL) from an image
3. Filter the image to sharpen edges and ease the separation of the particles from the background
4. Threshold the image to isolate the appropriate particles
5. Proper Close the image to fill tiny holes and smoothen inner contours of objects based on the structuring element
6. Fill holes that appear in the particles after thresholding
7. Remove all objects touching the border so that we remove partial particles
8. Use a particle filter to find all circular particles and remove non-circular/elliptical particles
9. Perform a particle analysis to find the number of objects

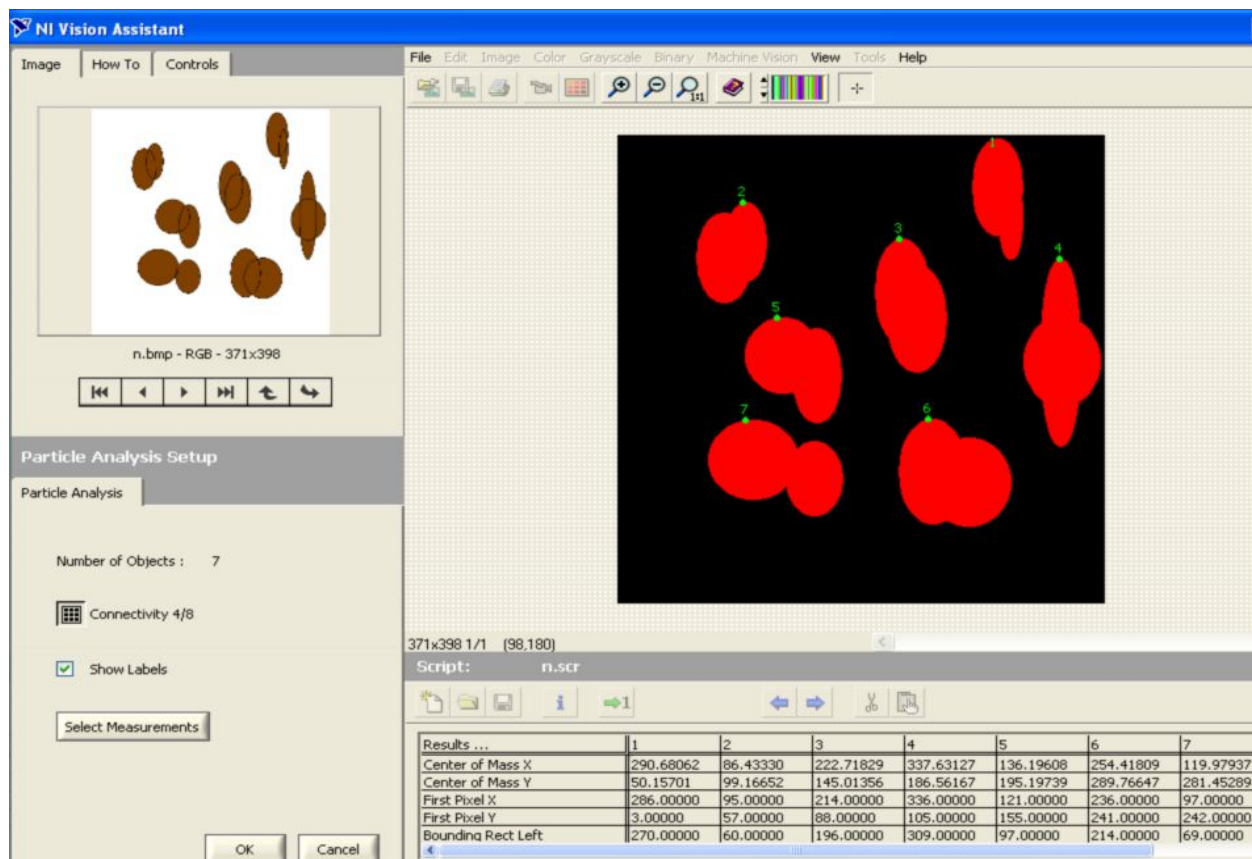
### **5.4 A warm-up exercise**

To familaise with the working of real image, we first performed a warm-up exercise involving a few steps required for bacteria colony counting. We created one synthetic image consisting of 7 pairs of overlapping circles/ellipses as shown in figure 5.1:



**Figure 5.1: Synthetic Image of particle to be counted**

The objective was to count these groups without any manual intervention. Step 2-9 given in section 5.3 were followed to give appropriate number of cells as shown in figure 5.2:



**Figure 5.2: Result of particle Counting**

The details of these steps as employed on the real image are given in the subsequent section.

## 5.5 Problem Algorithm

### 5.5.1 Acquiring image of Petri dish

The Petri dish is placed in well illuminated area, duly surrounded by light screen of avoid multiple shadows formed by ambient light, as shown in figure 5.3:

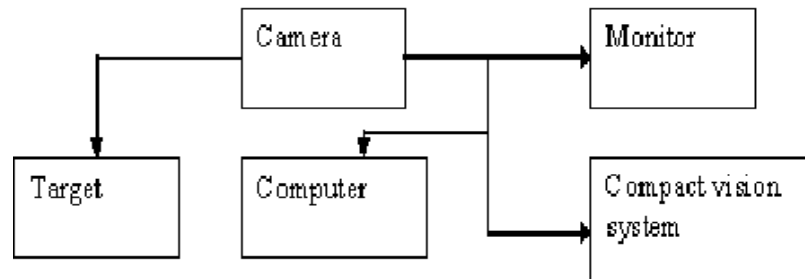


**Figure 5.3: Setup of acquiring image**

The image is acquired using IEEE-1394 digital camera Prosilica 2.0.1 and stored in mv.bmp file for further processing. In our work, we used 40 watt tube hanging on a stand of height 60cm approximately. Camera is attached with the stand.

## Hardware setup

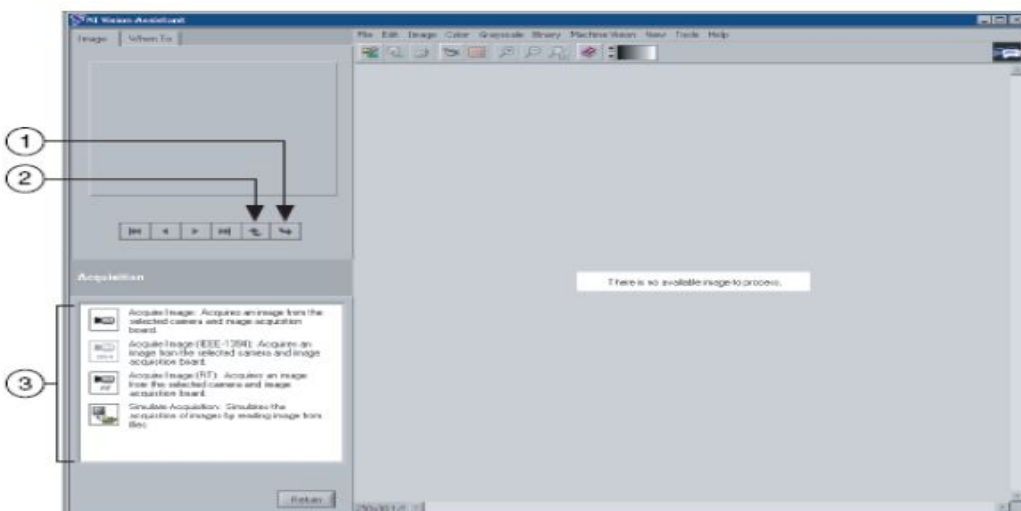
The hardware setup which we used in our work is shown in figure 5.4:



**Figure 5.4: Hardware Setup**

### 5.5.2 Opening the stored image

Image Browser contains all of the images currently loaded in Vision Assistant as shown in figure 5.5.

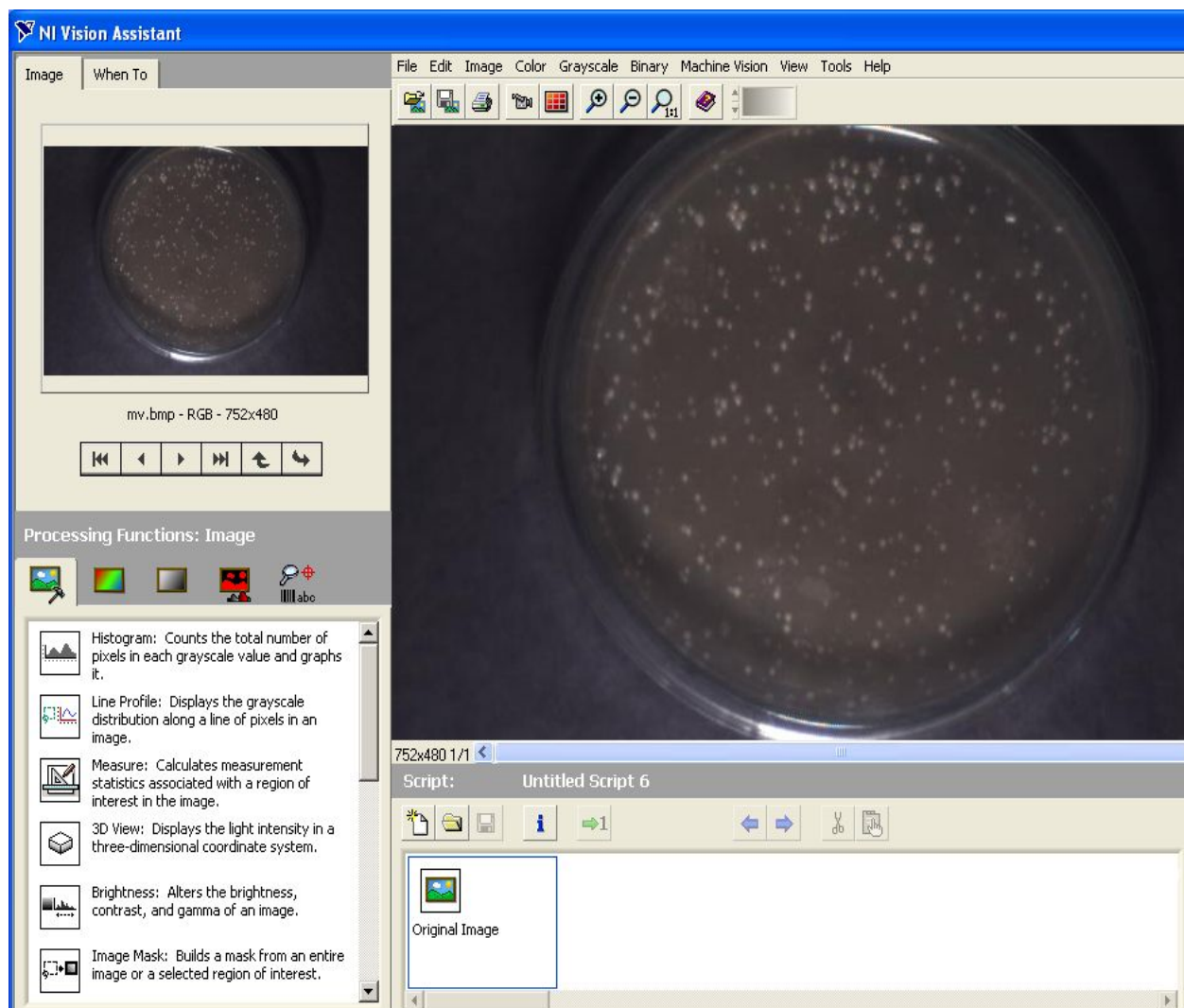


1. Make image active
2. Store acquired image in browser button
3. Acquisition functions

**Figure 5.5: Acquiring images in Vision Assistant**

We can select an image to process by double-clicking it in the Image Browser and processing window updates the image as we change parameters. Because this view immediately reflects the changes we have made in the Parameter window, we can continue modifying parameters until we get the result we want. The image is opened from the previously stored images by the following steps:

1. Select **Start»Programs»National Instruments Vision Assistant 7.1**
2. Click **Open Image** on the Welcome Screen
3. Select the required image and Click **OK** to load that image into Vision Assistant. The image so opened is shown in figure 5.6:



**Figure 5.6: Opening an image**

### 5.5.3 Extracting color planes from an image

We can extract one of the three color planes (Red, Green, Blue, Hue, Saturation, Luminance, and Value) from an image by the following steps:

1. Click Color»Extract Color Planes or **select** Extract Color Planes **in the Color tab of the Processing Functions palette**
2. Select the Red color plane
3. Click OK to add this step to the script. The extract red plane of the image is shown in figure 5.7:

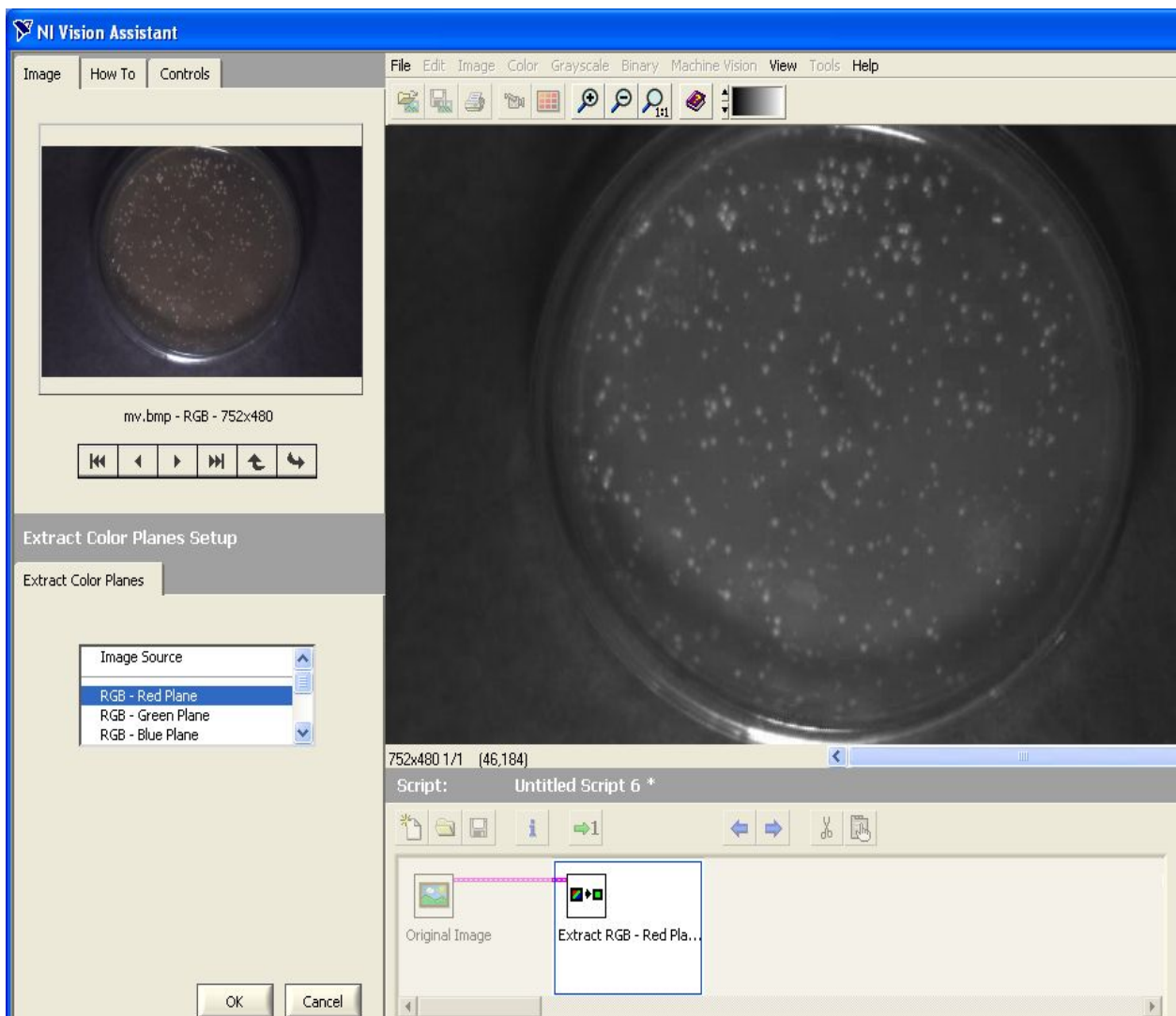


Figure5.7: Extracting RGB-Red Plane



The color plane we extract is an 8-bit grayscale image because each color plane is made up of 8 bits.

#### 5.5.4 Filtering the Image

Filters can smooth, sharpen, transform, and remove noise from an image so that we can extract the information we need. Most of these filters apply a kernel across the image. A kernel represents a pixel and its relationship to neighboring pixels. The weight of the relationship is specified by the coefficients of each neighbor. To sharpen edges, including the edges of any holes inside a particle, and create contrast between the particles and the background, by the following steps:

1. Select **Filters** in the **Grayscale** tab of the Inspection steps, or select **Grayscale»Filters**
2. Select **Convolution-Highlight Details** from the Filters list. This function detects sharp transitions and highlights edge pixels according to a *kernel* to make gaps more prominent. A kernel is a structure that represents a pixel and its relationship to its neighbors
3. Adjust the **Kernel Size** and coefficients, if necessary
4. Click **OK** to filter the image and add this step to the script. The filtered image is shown in figure 5.8:



**Figure 5.8: Filtering the Image**

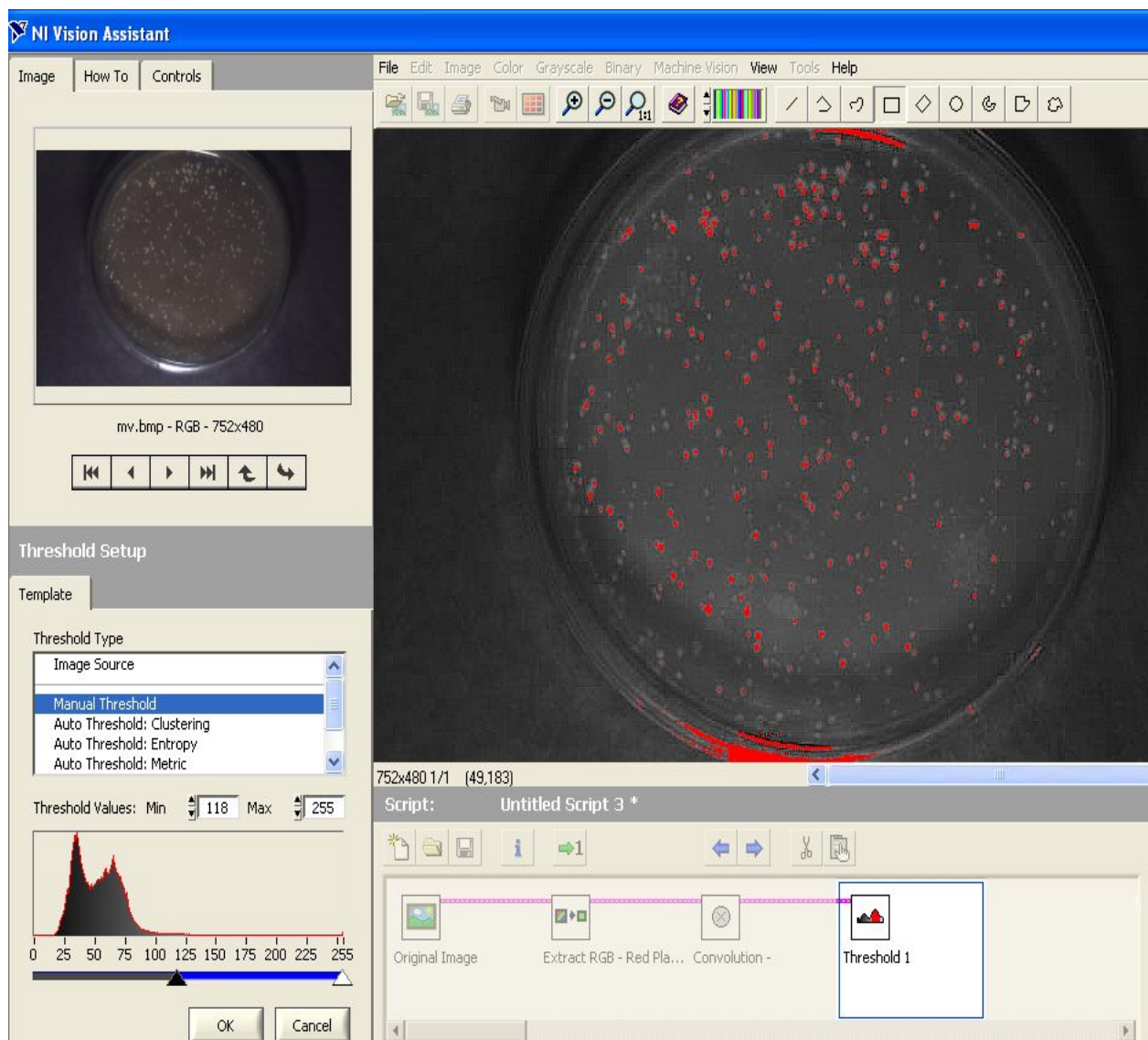
### 5.5.5 Separating Particles from the Background with Thresholding

Threshold isolates pixels that interest you and sets the remaining pixels as background pixels. Threshold also converts the image from grayscale to binary. It Selects ranges of pixel values in grayscale images by the following steps:

1. Select **Threshold** in the **Grayscale** tab of the Processing Functions palette or select **Grayscale»Threshold**. The Threshold Parameter window displays a histogram. A histogram counts the total number of pixels in each grayscale value and graphs it. From the graph, we can tell if the image contains distinct regions of a certain grayscale value. We also can select pixel regions of the image
2. Select **Manual Threshold** from the Threshold list
3. Select a range of **118 to 255**



4. We noticed that the particles of interest (circular and non-circular) are highlighted in red. When we apply the threshold, everything highlighted is set to 1, and all other pixels are set to 0. We can select the range using the pointers on the histogram rather than entering numbers in the **Min** and **Max** fields. Adjust the pointers until all of the objects we want to select are red. The black pointer marks the minimum value, and the white pointer marks the maximum value
5. Click **OK** to apply the threshold and add this step to the script. The separating particles image is shown in figure 5.9:



**Figure 5.9: Separating Particles from the Background with Thresholding**

The Manual Threshold operation enables you to select ranges of grayscale pixel values. Automatic threshold operations select threshold ranges for us. Use automatic thresholds for images in which light intensity varies.

The pixels that we selected for processing appear red. Unselected pixels appear black. The image is now a binary image, which is an image, composed of pixels with values of 0 and 1. This image is displayed using a binary palette, which displays the pixel intensities of an image with unique colors. All pixels with a value of 0 appear black and pixels set to 1 appear red. The red pixels are now referred to as particles.

### 5.5.6 Modifying Particles with Morphological Functions

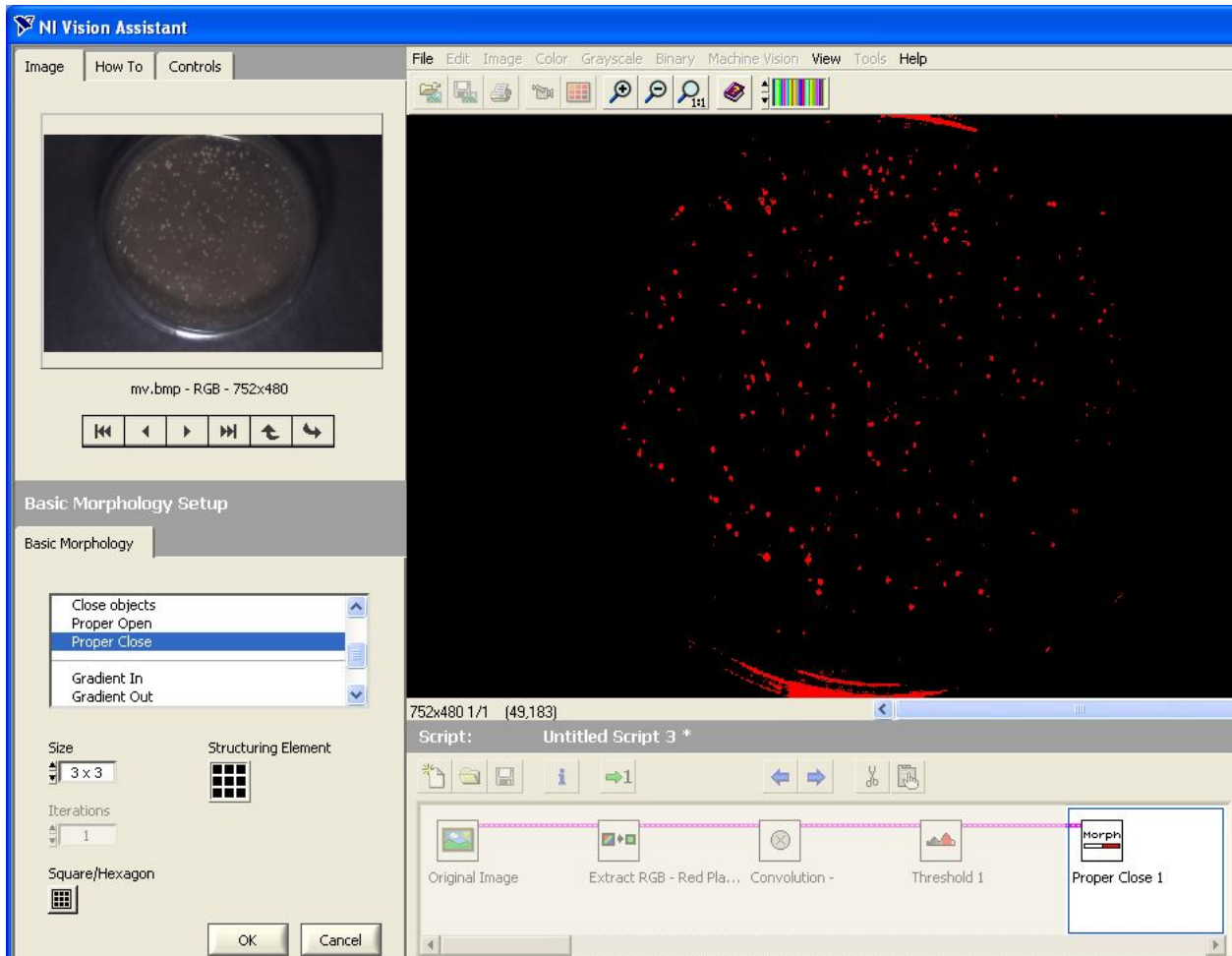
Morphological functions affect the shape of particles on an individual basis. Morphological operations prepare particles in the image for quantitative analysis such as finding the area, perimeter, or orientation.

#### Basic Morphology

It affects the shape of particles in binary images. Each particle or region is affected on an individual basis. We can use these functions for tasks such as expanding or reducing objects, filling holes, closing particles, or smoothing boundaries, which are tasks we perform to delineate objects and prepare images for quantitative analysis. **Proper Close** fills tiny holes and smoothes inner contours of objects based on the structuring element. It is a finite and dual combination of closings and openings.

The inner contours of objects are smoothened by the following steps:

1. Select **Basic Morphology** in the **Binary** tab of the Processing Functions palette or select **Binary»Basic Morphology**
2. Select **Proper Close** from the Morphology-Basic function list
3. Click **OK** to add this step to the script. The proper close image is shown in figure 5.10:



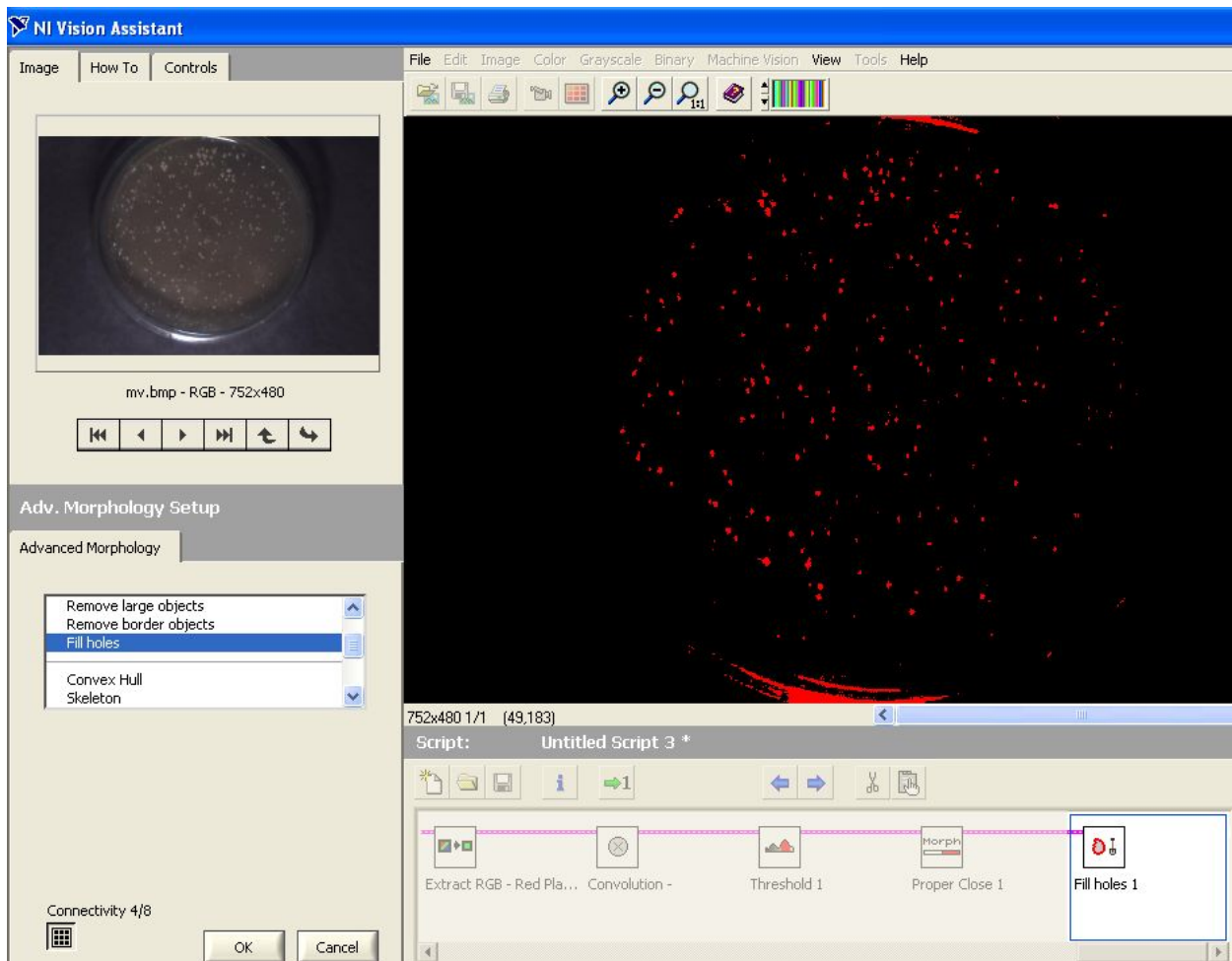
**Figure 5.10: Modifying Particles with Proper Close**

## Advanced Morphology

It performs high-level operations on particles in binary images. We are using these functions for tasks such as removing small particles from an image, labeling particles in an image, or filling holes in particles.

- **Fill holes** found in a particle. Holes are filled with a pixel value of 1. This function fills holes in the particles of the image by the following steps:
  1. Select **Adv. Morphology** in the **Binary** tab of the Processing Functions palette or select **Binary»Adv. Morphology**
  2. Select **Fill holes** from the Morphology-Advanced function list

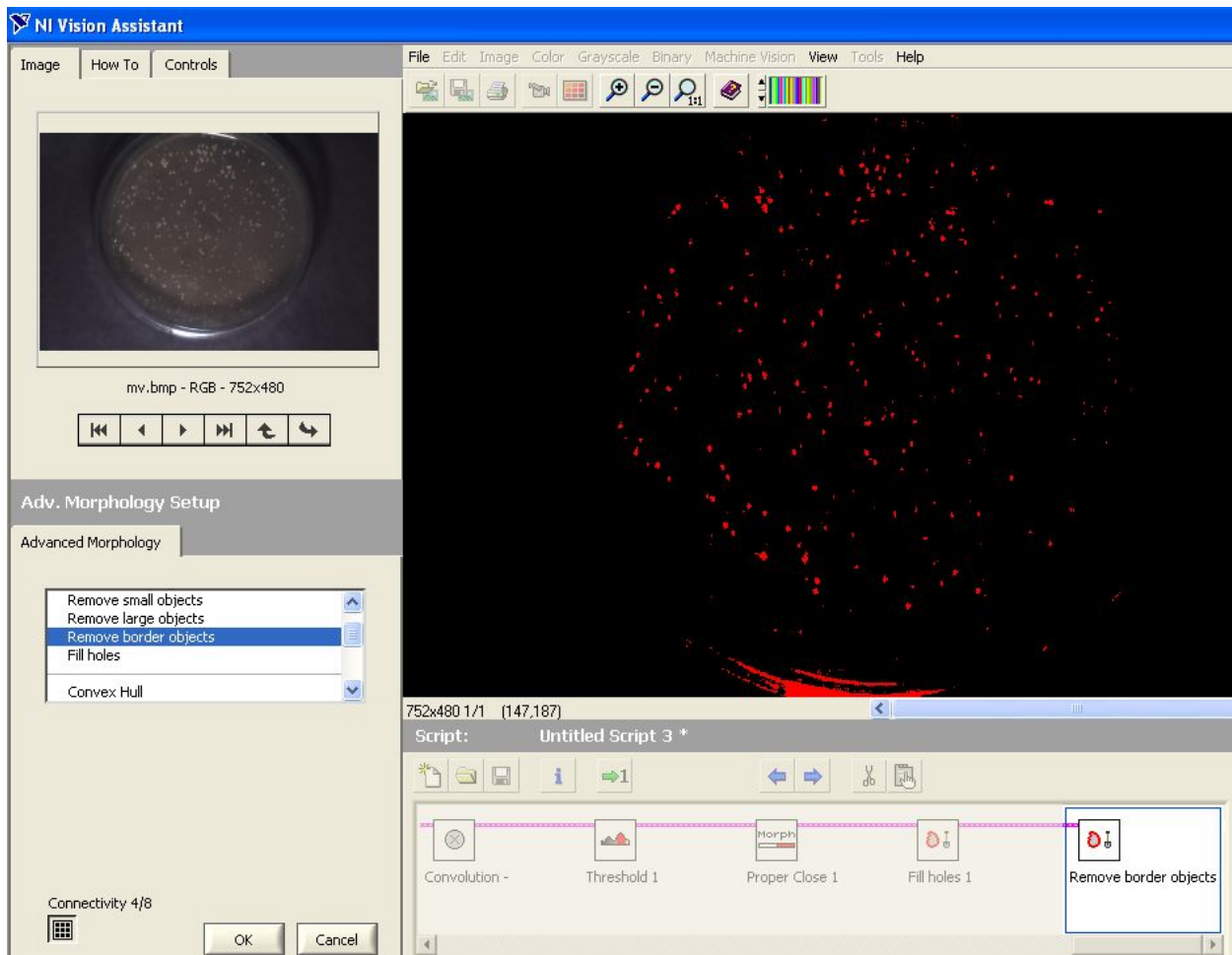
3. Click **OK** to add this step to the script. The fill holes in the particle of the image is shown in figure 5.11:



**Figure 5.11: Modifying Particles with fill holes**

- **Remove border objects** eliminate particles that touch the borders of an image by the following steps:
  1. Select **Adv. Morphology** in the **Binary** tab of the Processing Functions palette or select **Binary»Adv. Morphology**
  2. Select **Remove border objects** to remove any objects that touch the border of the image

3. Click **OK** to add this step to the script and close the Parameters window. The remove border objects of an image is shown in figure 5.12:



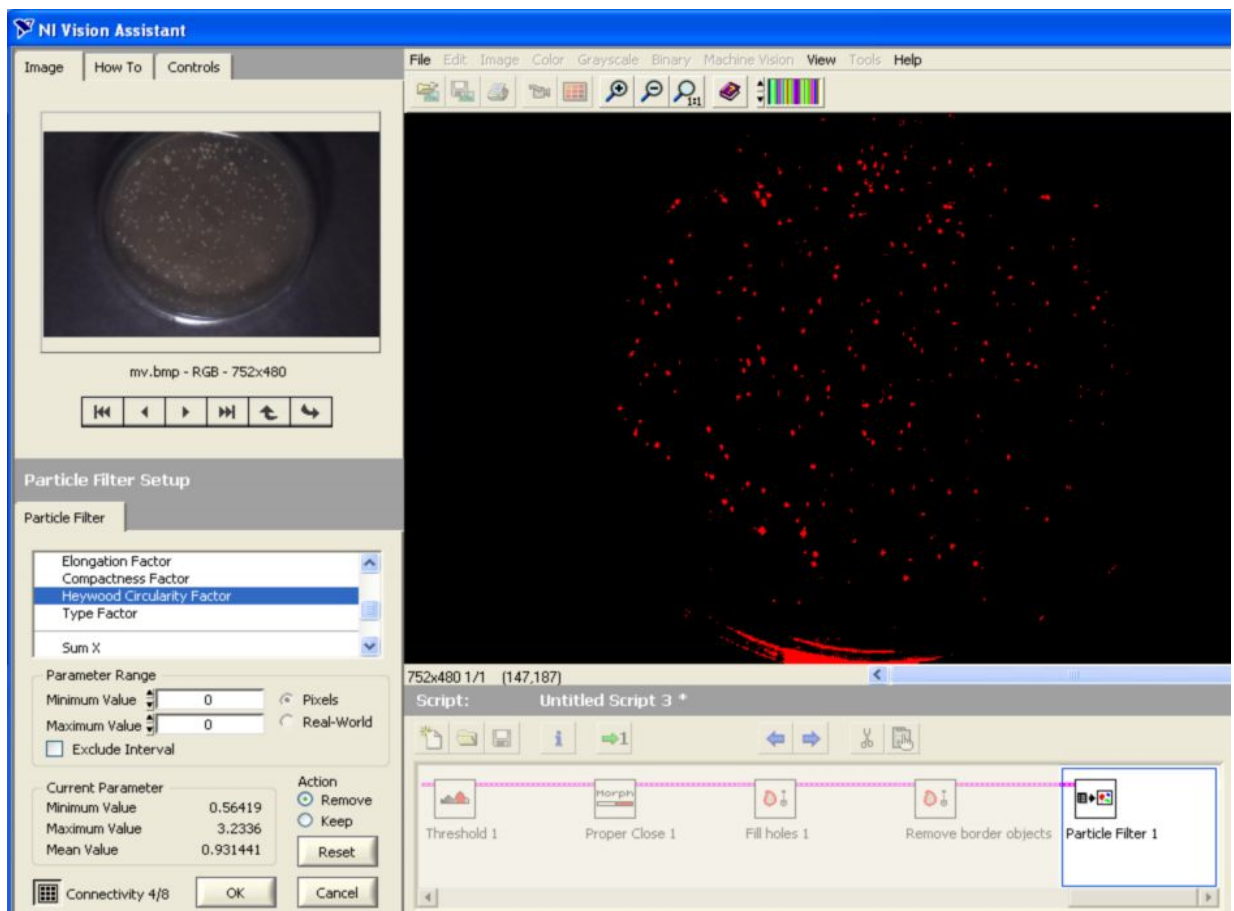
**Figure 5.12: Modifying Particles with Remove Border Objects**

### 5.5.7 Isolating Circular Particles

Particle Filter removes or keeps particles in an image as specified by the filter criteria. **Heywood Circularity Factor** is the Perimeter which is divided by the circumference of a circle with the same area. The closer the shape of a particle is to a disk, the closer the Heywood circularity factor to 1.

A particle filter that isolates and keeps the circular particles and removes the non-circular particles from the image is showing by the following steps:

1. Select **Particle Filter** from the **Binary** tab of the Processing Functions palette, or select **Binary»Particle Filter**.
2. Select **Heywood Circularity Factor** from the list of particle filters. This function calculates the ratio of the perimeter of the particle to the perimeter of the circle with the same area. The more circular the particle, the closer the ratio to 1.
3. To find more circular and less oblong particles, enter a **Minimum Value** of 0 and a **Maximum Value** of 0 for the parameter range.
4. Select the **Remove** option to remove particles that do not fit in this range.
5. Click **OK** to add this step to the script. The image now contains only circular particles.
6. If we want the filter to be applied outside of the specified parameter range, check the **Exclude Interval** option. The image of isolating circular particles is shown in figure 5.13:



**Figure 5.13: Isolating Circular Particles**

### 5.5.8 Analyzing Circular Particles

Particle Analysis displays measurement results for selected particle measurements performed on the image.

**Results**—display the results for each particle in the image.

**Number of Objects**—displays the total number of particles found in the image.

**Connectivity 4/8**—defines which of the surrounding pixels for any given pixel constitute its neighborhood. **Connectivity-8** shows that all adjacent pixels are considered neighbors and **Connectivity-4** shows that only pixels adjacent in the horizontal and vertical directions are considered neighbors.

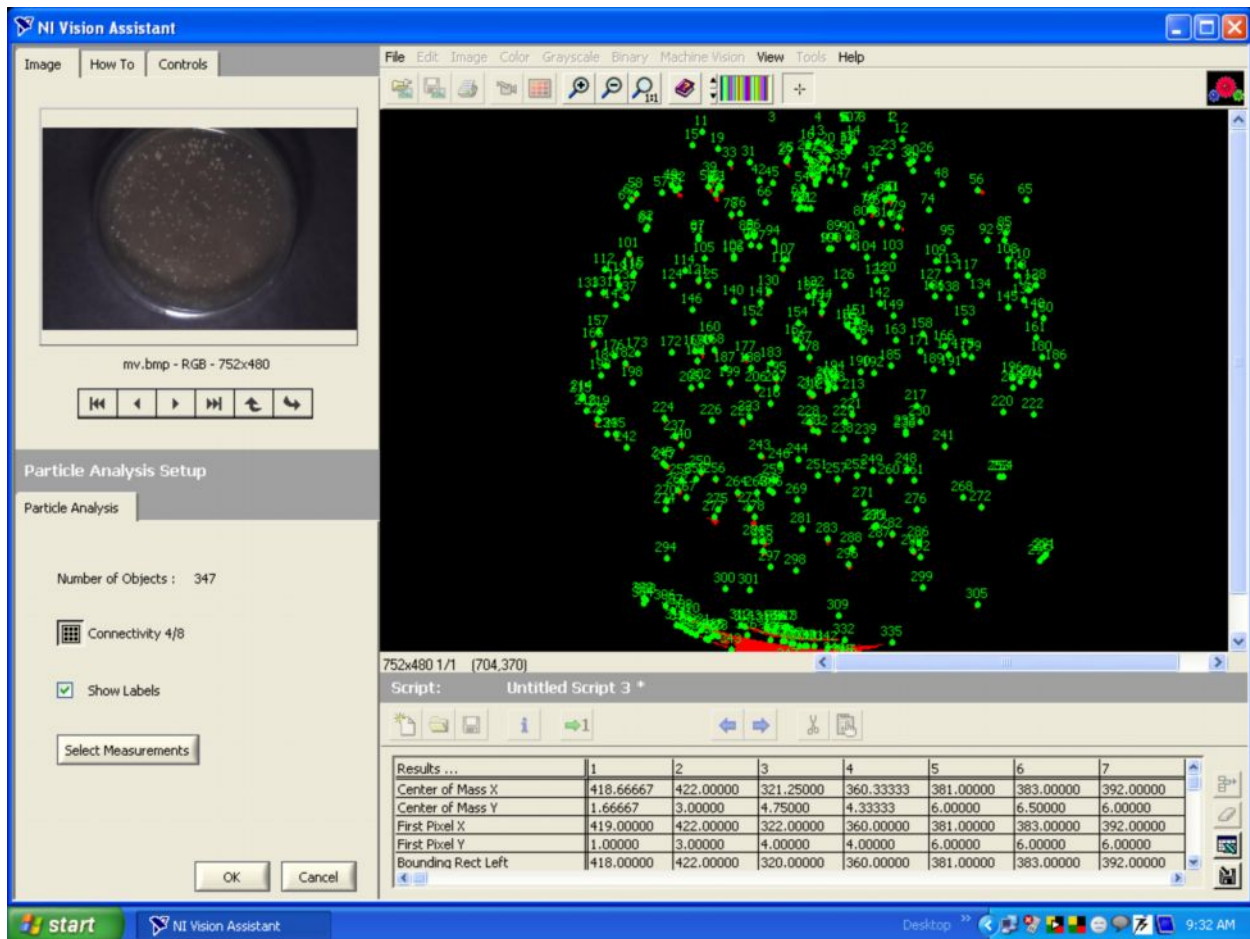
**Select Measurements**—displays a list of object measurements that can be calculated and displayed.

**Show Labels**—Vision Assistant assigns numeric labels to objects that it analyzes.

Now that we have isolated circular particles, find the Number of Bacteria by the following steps:

1. Select **Particle Analysis** from the **Binary** tab of the Processing Functions palette, or select **Binary»Particle Analysis**. A results table displays all of the measurement results. Vision Assistant assigns numerical labels to each particle. The first row of the results table lists the numerical label associated with each particle
2. Select **Show Labels** to view the labels
3. When we click a particle, the measurement results for that particle are highlighted in blue. When we click the results for a particle, the particle is highlighted in green in the processing view
4. Click **OK** to record the particle analysis and add the step to the script. The image of analyzing circular particles is shown in figure 5.14:





**Figure 5.14: Counting Circular Particles**

### 5.5.9 The Complete Processing Script

The image acquired by camera is stored under appropriate file name mv.bmp. This image is processed by nine distinct steps namely acquiring image, opening an image, extracting color planes from an image, filtering the image, separating particles from background with thresholding, proper close, fill holes, remove border objects, isolating circular particles and counting circular particles. It is very important to note that these nine steps do not require nine manual interventions and the entire process from opening an image to counting circular particles in an image is done in one script as shown in figure 5.15 and figure 5.16:





Figure 5.15: Processing Script

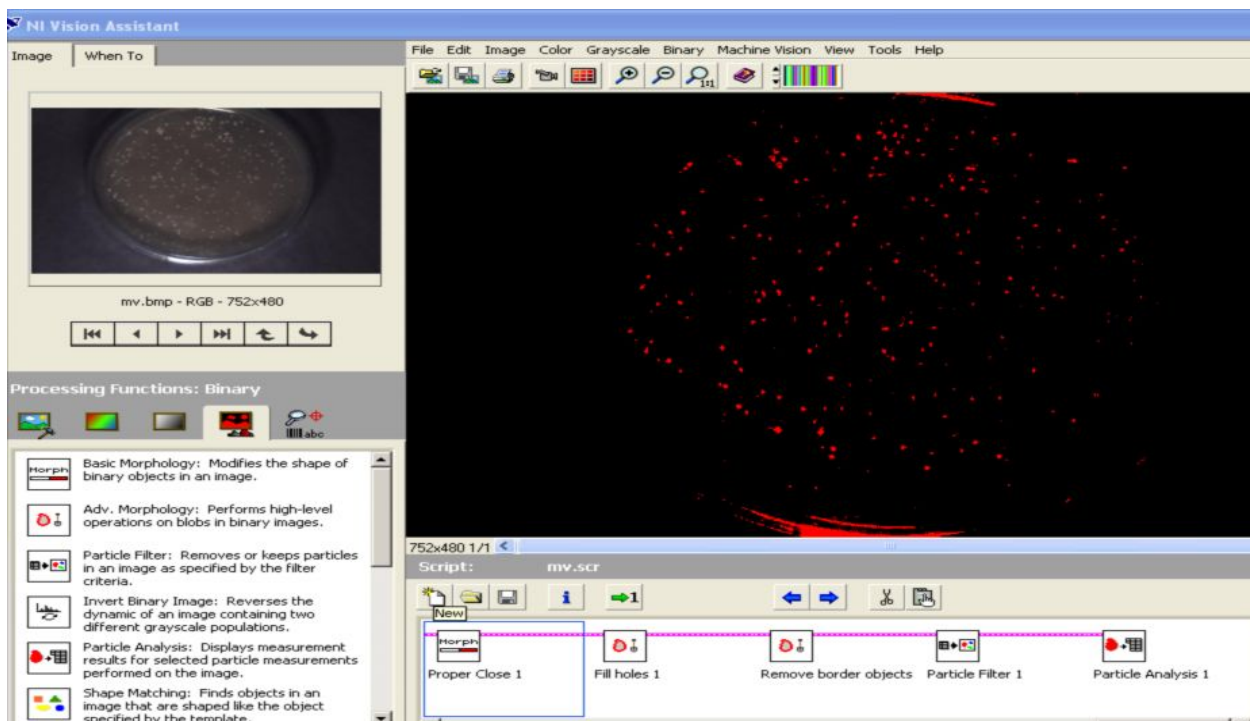


Figure 5.16: Processing Script (Continued)

The user simply has to select the file name of image and click once for obtaining the final count.

### 5.5.10 Estimating Processing Time

Vision Assistant can estimate the time, in milliseconds, that IMAQ Vision will take to process the active image with the open script. The Performance Meter gives both an estimate of the total time IMAQ Vision will take to process the image and an estimate of the time each function within the script will require as shown in figure 5.17. We estimate how many milliseconds IMAQ Vision will use to process mv.jpg with mv.scr by the following steps:

1. Select **Tools»Performance Meter**. The Performance Meter estimates the total time IMAQ Vision will take to run the script
2. Click **Details** to view an itemized list of the time IMAQ Vision will take to perform each function in the script
3. Click **OK** to close the Performance Meter

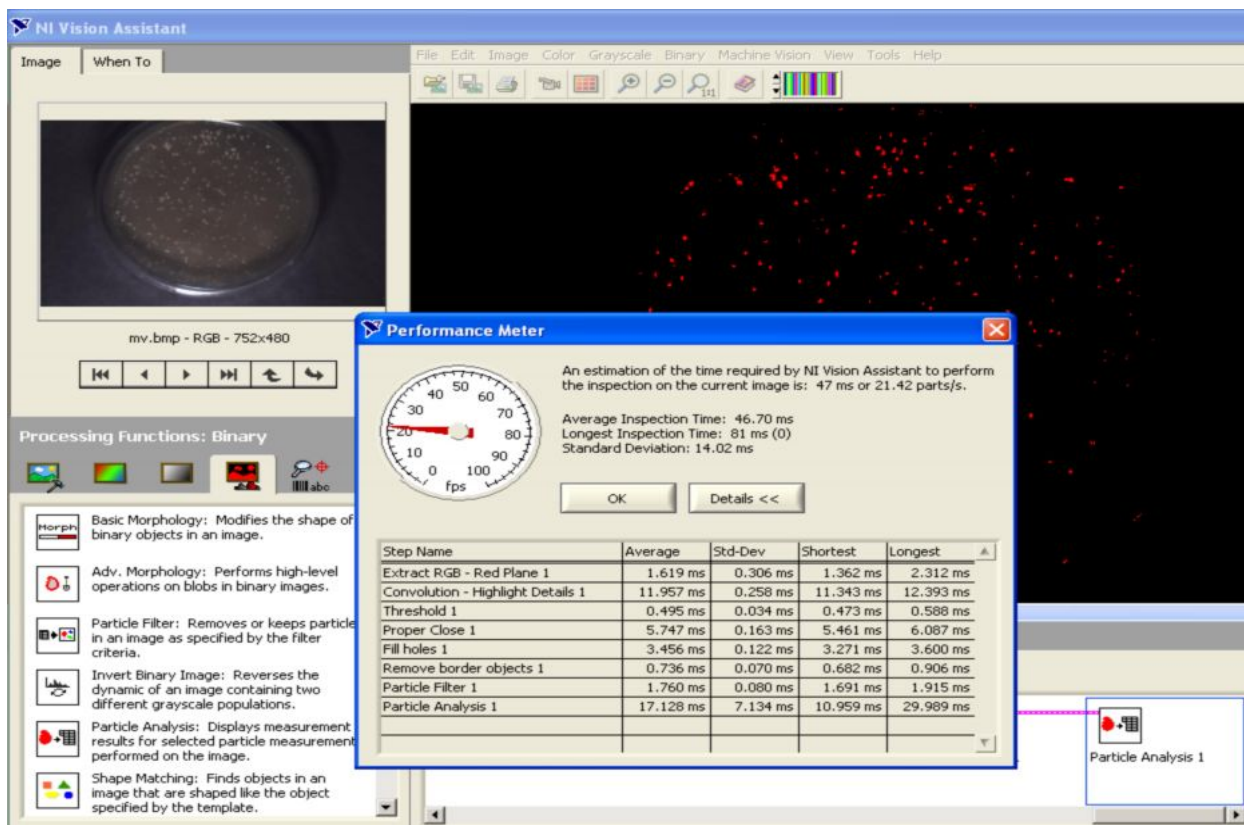


Figure 5.17: Estimating Processing Time of an image

The system shows the number of microorganisms present in processed image. From our analysis, we have come to the conclusion that colonies in a Petri dish can be easily counted by Particle Analysis. We perform a particle analysis to detect connected regions or, groupings of pixels in an image and then make selected measurements of those regions. Average Processing time of script for 72 cases is 47ms which is very less as compared to manual counting of bacteria.

## 5.6 Results and discussion

To validate our script we executed our program on 72 numbers of Petri dishes of culture bacteria like E.coli, Pseudomonas, and Lactobacillus. The colonies were counted to cross check the count given by one script. Two or more overlapping colonies were counted as one in both manual and automatic mode. The result of this count is given in Table 5.2:

S. No.	Manual counting	Automatic counting	Error	Percentage error
1	104	106	+2	1.92%
2	272	268	-4	-1.47%
3	162	156	-6	-3.70%
4	183	187	+4	2.19%
5	277	280	+3	1.08%
6	120	116	-4	-3.33%
7	192	190	-2	-1.04%
8	192	189	-3	-1.56%
9	193	189	-4	-2.07%
10 *	218	235	+17	7.80%
11	459	461	+2	0.44%
12	347	350	+3	0.86%
13	350	352	+2	0.57%
14	220	218	-2	-0.91%
15	240	243	+3	1.25%
16	447	449	+2	0.45%

17	305	303	-2	-0.66%
18	333	330	-3	-0.90%
19	247	250	+3	1.21%
20	266	269	+3	1.13%
21	278	280	+2	0.72%
22	314	310	-4	-1.27%
23	323	325	+2	0.62%
24	115	117	+2	1.74%
25	236	233	-3	-1.27%
26	178	174	-4	-2.25%
27	166	169	+3	1.81%
28	489	491	+2	0.41%
29	179	181	+2	1.12%
30	144	147	+3	2.08%
31	456	454	-2	-0.44%
32	378	376	-2	-0.53%
33	431	435	+4	0.93%
34	401	405	+4	1.00%
35	279	275	-4	-1.43%
36	258	260	+2	0.78%
37 *	175	200	+25	14.29%
38	177	180	+3	1.69%
39 *	233	210	-23	-9.87%
40	105	103	-2	-1.90%
41	90	92	+2	2.22%
42	165	162	-3	-1.82%
43	77	75	-2	-2.60%
44	48	50	+2	4.17%
45	296	300	+4	1.35%
46	368	366	-2	-0.54%

47	407	410	+3	0.74%
48	374	376	+2	0.53%
49	348	345	-3	-0.86%
50	263	267	+4	1.52%
51	136	134	-2	-1.47%
52	439	441	+2	0.46%
53	295	300	+5	1.69%
54 *	75	88	+13	17.33%
55	187	190	+3	1.60%
56	170	168	-2	-1.18%
57	387	385	-2	-0.52%
58	261	265	+4	1.53%
59	438	440	+2	0.46%
60	98	101	+2	3.06%
61	179	180	+1	0.56%
62	184	179	-5	-2.72%
63	138	142	+4	2.90%
64	483	487	+4	0.83%
65 *	356	386	+30	8.43%
66	496	493	-3	-0.60%
67	186	185	-1	-0.54%
68	48	50	+2	4.17%
69	57	55	-2	-3.51%
70 *	260	246	-14	-5.38%
71	435	436	+1	0.23%
72	484	482	-2	-0.41%

**Table 5.1: Results and discussion**

Barring six readings marked \* the remaining 66 readings were very much within the tolerance limit of error of  $\pm 5\%$ .

The percentage Root Mean Squares of the remaining 66 readings were

$$= \sqrt{\frac{\text{Sum of (Percentage error * Percentage error)}}{\text{Number of readings}}}$$

$$= \sqrt{\frac{0.019212}{66}}$$

$$= \pm 1.71 \%$$

### Conclusions & Future scope

---

#### 6.1 Conclusions

The proposed Machine Vision based Method is a robust yet effective method for bacterial counter. It has the ability to detect the dish plate regions, isolate colonies on the dish plate and further separate the clustered colonies for accurate counting of colonies. Machine vision is an emerging area of technology in automation and control, wherein the images captured by camera or the images previously taken are processed and analyzed in real time. There are different software tools available for image processing and analysis. Vision Assistant 7.1 is one of these tools and is useful for doing real time operations. In our work, we have made use of IEEE 1394 digital camera for image acquisition and National Instruments Vision Assistant 7.1 software for image processing and analysis. The performance of the proposed method is promising, especially when processing the dish plate with color medium. The manual counting of colonies by a lab worker results in inaccurate counts of the bacteria colonies. Machine vision based bacteria Colony Counter would allow each plate to be counted with equal efficiency. From our analysis, we have come to the conclusion that bacteria colonies in a Petri dish can be easily counted by Particle Analysis.

#### 6.2 Future Scope

The present work is a first attempt to count bacteria colonies by machine vision. We find that after optimizing our script for various values of parameters like acquiring image, opening an image, extracting color planes from an image, filtering the image, separating particles from background with thresholding, proper close, fill holes, remove border objects, isolating circular particles and counting circular particles, 66 out of 72 Petri dishes so presented were within counting error of  $\pm 5\%$ . However the process needs to be modified /optimized to include 100% of population. Moreover the present work was restricted to strains of bacteria like E.coli,

Pseudomonas and Lactobacillus. This may also be extended to cover the entire range of bacteria so available for laboratory analysis. The present work restricted itself to counting only neat colonies that is those which could be counted by manual visual inspection. However, advanced image process techniques may enable to count the colonies so far as possible by manual visual inspection. The scope for further work in this area is thus limited only by the imagination of research.



## References

---

1. Bang Wei, Zhang and Chengcui Chen (2007) “An Effective and Robust Method for Automatic Bacterial Colony Enumeration” Proceedings of the International Workshop on Semantic Computing and Multimedia Systems, in conjunction with the International Conference on Semantic Computing, Volume 17, Page No. 581-588
2. Coyne M. B., Forage A. J. and Paice F. (1974) “Bacterial Colony Counting Using the M.R.C. Image Analyser” Journal of Radio pathology, Volume 19, Page No. 708-715
3. Caldwell R. Daniel and Marvin P. Bryant (1966) “Medium Without Rumen Fluid for Nonselective Enumeration and Isolation of Rumen Bacteria” Journal of Bacteriology, Volume 14, Page No. 794-801
4. Dubuisson Marie-Pierre, Jain Anil K. and Jain Mahendra K. (1994) “Segmentation and Classification of Bacterial Culture Images” Journal of Microbiological Methods Volume 19, Issue 4, Page No.279-295
5. Gilchrist J. E., Campbell J. E., Donnelly C. B., Peeler J. T. and Delaney J. M. (1973) “Spiral Plate Method for Bacterial Determination”, Volume 25, Page No. 244-252.
6. [http: Colony] “Colony” is available at <http://www.ruf.rice.edu/~bioslabs/bios318/colony.htm>
7. [http:Functions] “Functions and Applications of Machine Vision” is available at [http://www.autovis.com/courses/Fundamentals\\_syllabus.html](http://www.autovis.com/courses/Fundamentals_syllabus.html)
8. [http:Growth-1] “Growth and Reproduction of Bacteria” is available at <http://www.textbookofbacteriology.net/growth.html>
9. [http:Growth-2] “Phases of Growth” is available at <http://biology.clc.uc.edu/fankhauser/Labs/Microbiology/GrowthCurve/GrowthCurve.htm>
10. [http:Machine-a] “Anatomy of Machine Vision” is available at <http://www.mellegriot.com/products/machinevision/machinetutorial.asp>
11. [http:Machine-b] “Components of Machine Vision” is available at <http://en.wikipedia.org/wiki/Machinevision>
12. [http:Wiki-a] “Agar Plate” is available at [http://en.wikipedia.org/wiki/Agar\\_plate](http://en.wikipedia.org/wiki/Agar_plate)

13. [http:Wiki-b] “Bacteria” is available at <http://en.wikipedia.org/wiki/Bacteria>
14. [http:Wiki-c] “Colony Counter” is available at [http://en.wikipedia.org/wiki/Colony\\_counter](http://en.wikipedia.org/wiki/Colony_counter)
15. John M. Kramer, Margaret Kendall and Richard J. Gilbert (1978) “Evaluation of the Spiral Plate and Laser Colony Counting techniques for the Enumeration of Bacteria in foods”, *Journal of Applied Microbiology and Biotechnology*, Volume 6, Page No. 289-299
16. Kawai M., Yamaguchin. and Nasu M. (1999) “Rapid Enumeration of Physiologically active Bacteria in purified water used in the Pharmaceutical Manufacturing Process” *Journal of Applied Microbiology*, Volume 86, Page No. 496-504
17. Khatipov Emir (2001) is available at <http://www.bio.net/bionet/mm/methods/2001-October/090763.html>
18. Mukherjee Dipti, Pal Amita, Sharma S. Eswara and Majumder D. Dutta (1995) “Water Quality Analysis: A Pattern Recognition Approach” Volume 28, Issue 2, Pages No. 269-281
19. [NI-a] IMAQ NI CVS-1450 series user manual is available at <http://www.ni.com/pdf/manuals/373610d.pdf>
20. [NI-b] NI VA tutorial manual
21. [NI-c] NI concepts manual
22. Paul C. Olsztyn, James H. Beyer and David Sullivan (1998) “Bacteria Colony Counter & Classifier” Volume 83, Page No. 382-400
23. Putman M., Burton R. and Nahm M. H. (2005) “Simplified Method to Automatically Count Bacterial Colony Forming Unit,” *Journal of Immunological Methods*, Volume 302, Page No. 99-102
24. Whelan F. (2001) “Machine Vision Algorithms”, Volume 34, Page No. 114-119