A Study and Implementation of Image Processing Algorithms & Applications On Embedded Platform

Technical Report · August 2008

CITATIONS
0

READS
93

1 author:

Debaleena Chattopadhyay
University of Illinois at Chicago
33 PUBLICATIONS **296** CITATIONS

SEE PROFILE

A Study and Implementation of Image Processing Algorithms & Applications On Embedded Platform

A Technical Project Report

By

Debaleena Chattopadhyay
Computer Science and Engineering Dept.
Institute of Engineering & management
West Bengal University of Technology
Kolkata, India

At

TATA CONSULTANCY SERVICES

Abstract

Digital Image processing has gradually expanded and diversified into several branches based on mathematical tools as well as applications. It is also successfully encroaching the embedded platform. This project report delineates four different image processing implementations. In spite of the fact, that they belong to the same genre, they are way apart in their domain. So, each of the implementations have been presented in a section of its own as an independent sub-project.

SECTION I: Contrast Enhancement by Changing Distribution and Laplacian Transform

This work aims to enhance the image quality of video transmissions, which are susceptible to various degrading agents. Here, we have implemented some aspects of image enhancement to get a desired quality output. The enhancement techniques modified only the gray level of the images.

SECTION II: Shot Detection by Feature Extraction of Co-occurrence matrixes

This work proposes a threshold-independent methodology for abrupt scene cut detection. The proposed study is based on feature computation from RGB co-occurrence matrices' statistics, defined at various pixel displacement distances. It integrates the statistical find-outs in a training set and implements a learning paradigm.

SECTION III: XDAIS-compliant API Implementation & the Multimedia Toolbox

The TMS320 DSP Algorithm Standard™, also known as XDAIS, is part of TI's eXpressDSP™ initiative. The purpose of the standard is to reduce those factors that prohibit an algorithm from being easily integrated into a system without significant re-engineering. The Multimedia Toolbox is an advanced video processing toolbox that has been designed as a modularized application (a dynamic library) so that its components may be used as plug-ins to other applications.

SECTION IV: Missing Medical Instrument Identification

Missing Medical Instrument Identification is an automated process to enumerate, identify and locate missing medical instruments within an Instrument Case. The idea is to primarily build a database consisting of enlisted medical instrument boxes keyed by an identification mark like an RFID tag. Now, based on the existent database, we are supposed to test live Instrument Cases and determine whether or not they are consistent with the master templates.

Acknowledgements

I would like to thank the members of Multimedia Group, TCS Innovation Lab, Kolkata, namely Mr.Aniruddha Sinha, Mr. Ayan Chaki, Mr. Brojeshwar Bhowmick, Mr. Biswanath Saha, Mr. K. S. Chidanand, Mr. Kaustav Goswami and Mr. Tanushyam Chattopadhyay for giving me a fantastic opportunity to work with them and guiding me thoroughly in my endeavors.

Table of Contents

SECTION I

Contrast Enhancement by Changing Distribution and Laplacian Transform

SECTION II

Shot Detection by Feature Extraction of Co-occurrence matrixes

SECTION III

XDAIS-compliant API Implementation & the Multimedia Toolbox

SECTION IV

Missing Medical Instrument Identification

SECTION I

Contrast Enhancement by Changing Distribution and Laplacian Transform

1. Introduction

The work we aimed at was to enhance the contrast quality of a transmitted video over television. Due to additive noise, the videos get degrade, and out here we have made a solemn effort to improve its quality. Since, videos for television broadcasting are in YUV format, we have worked with algorithms that shall modify the graylevels of the picture. The results marshaled here are images. But, we have also tested with full length videos and got expected improvements. Other than the graylevel enhancement algorithms, we are also looking forward to intensify specific colors (RGB) within an image. The parameter variance for changing distribution is a little eccentric, with more change at the mid region of the scale than the extremes. The laplacian transform however gives less visibility at the beginning of the scale but brightens the image up the scale. The algorithms have added advantage of minimal processing and hence minimal time complexity.

2. Image Degradation

Whenever an image is digitized, transmitted or scanned, i.e. converted from one form to another, some form of degradation occurs at the output. The noise added to an image produces some distortions, which can be to some extent minimized.

3. Image Enhancement

Improvement in the quality of these degraded images can be achieved by the application of restoration or/and enhancement technique. Restoration may be defined as an attempt to estimate the original image by applying effective inversion of the degrading phenomenon. This requires a priori model of the degradation process. With no such knowledge available we can improve the quality of the image for some specific application by using an ad hoc process called *image enhancement*.

In image enhancement, essentially any technique can be used, provided the resulting image provides additional information that was not readily apparent in the observed image.

The term *image enhancement* essentially means improvement of the appearance of an image by increasing dominance of some features, or by decreasing ambiguity between different regions of the image. So, notably certain features are being enhanced at the cost of suppressing other parts. The enhancement techniques can be divided into three categories:

1. Contrast intensification
2. Noise cleaning or smoothing and
3. Edge sharpening or crispening

The algorithms for image enhancement are basically developed using one of the basic two approaches.

- Spatial domain techniques
- Frequency domain techniques

The term ‘spatial-domain’ refers to the discrete image domain defined as

$$\{(r, c)|r = 0, 1, 2, \ldots\ldots, M-1; c = 0,1, 2,\ldots\ldots,N-1\} \quad \textit{Eq.(1)}$$

Spatial-domain operators directly operate on the pixel values g(r, c) of the image and can be expressed as

$$\sim g(r, c) = T(g(r, c), Q(r, c)) , \quad \textit{Eq.(2)}$$

for all r and c, where Q(r, c) is the setoff graylevels of the neighboring pixels. T is the operator in general, defined over some neighborhood of (r, c), and the operation is the convolution. Some of these operators are also called *filters*.

4. Histogram Specification

Among the various image enhancement techniques, this one emphasizes on the shape of the histogram. The shape of graylevel histogram gives an idea about the overall appearance of an image. For example, an image with a positively skewed graylevel histogram looks brighter than an image with negatively skewed graylevel histogram. The most common of these techniques is Histogram Equalization.

4.1 Changing the Distribution of the Image

4.1.1 Background Theory

It can be noted that if we modify the gray level of an image that has uniform Probability Distribution Function using inverse of the transformation:

$$l = T(m) = \int_0^m p_m(x)dx \qquad \textit{Eq.(3)}$$

Then, we will get an image that will have a p.d.f. similar to $p_m(m)$. Using this, we can obtain any shape of the gray level distribution by processing the given image in the following way. Suppose $p_m(m)$ and $p_l(l)$ represent the gray level p.d.fs. of input and output images and *m* and *l* are respective gray levels. Also, let *k* represents gray level of some intermediate image result, i.e.

$$k = T_1(m) = \int_0^m p_m(x)dx \qquad \textit{Eq.(4)}$$

and

$$k = T_2(l) = \int_0^l p_l(x)dx \qquad \textit{Eq.(5)}$$

Here we have taken the output probability distribution as

$$p_l(x) = \alpha \times e^{(-\alpha(x-x_{\min}))} \qquad \textit{Eq.(6)}$$

And in both the cases, either for $k = T_1(m)$ or for $k = T_2(l)$, $p_k(k)$ is uniform. Hence, the transformation

$$l = T_2^{-1}(\int_0^m p_m(x)dx) \qquad \textit{Eq.(7)}$$

achieves the desired result.

4.2 Unsharp Masking

4.2.1 Background theory

A well-celebrated process used to sharpen images consists of subtracting a blurred version of an image from the image itself. This process, called *unsharp masking*, is expressed as:

$$f_s(x,y) = f(x,y) - \bar{f}(x,y) \qquad \textit{Eq.(8)}$$

where *f*s(x, y) denotes the sharpened image obtained by unsharp masking, and is a blurred version of f(x, y).The origin of unsharp masking is in darkroom photography, where it consists of clamping together a blurred negative to a corresponding positive film and then developing this combination to produce a sharper image.
The unsharp masking can be applied on the original image by applying the following mask:

–1	–1	–1
–1	$A + 8$	–1
–1	–1	–1

Figure 1: Mask for Unsharp Masking

where A is the amplification factor.

When A=1, unsharp masking becomes "standard" Laplacian sharpening. As the value of *A* increases past 1, the contribution of the sharpening process becomes less and less important. Eventually, if *A* is large enough, the masked image will be approximately equal to the original image multiplied by a constant.
One of the principal applications of unsharp masking (with boost filtering) is when the input image is darker than desired. By varying the boost coefficient, it generally is possible to obtain an overall increase in average gray level of the image, thus helping to brighten the final result.

4.3 Changing Pixel Based on Laplacian

The laplacian can be represented as

$$\nabla f(x) = \frac{\partial f}{\partial x} = f(x+1) - f(x) \qquad \textit{Eq.(9)}$$

for a single variate function. But image is a bi-variate one. Now, let *f(x,y)* be the function. The double partial differential form (laplacian form) of image is

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial x^2} \qquad \textit{Eq.(10)}$$

which conveys the high frequency of the scene. Incorporating an amplification factor A can enhance those pixels.
Now, amplification can be applied as:

$$f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + A.f \ , \qquad \textit{Eq.(11)}$$

where *A* is the amplification factor.

5. Results

5.1 Results for changing Distribution:

We have taken the range of α as (–1, +1).
α=0 gives complete black image (absolute no visibility of the image attributes). As α→ +1 from 0, the image gets brighter. While as α→ -1 from 0, the image gets darker. When α reaches the boundary values i.e. α=+1 or α=-1, we are getting back the original image.

Figure 2: Input image 1

Figure 3(a): [α = +1]

Figure 3(b): [α = +0.1]

Figure 3(c): [α = +0.01]

Figure 3(d) [α = +0.001]

Figure 3(e): [α = -1]

Figure 3(f): [α = -0.1]

Figure 3(g): [α = -.01]

Figure 3(h) [α = -.001]

Figure 3: Output Image Set 1

5.2 Results for Laplacian Transform:

The amplification factor A is varied along the following range [8, 11.5]. When A=8, only the high frequency regions survive and the image is not clearly visible. As A→ 11.5, the image gets gradually brighter.

Figure 4: Input image 2

Figure 5(a): [A=8]

Figure 5(b): [A=8.5]

Figure 5(c): [A=9]

Figure 5(d): [A=9.5]


Figure 5(e): [A=10]


Figure 5(f): [A=10.5]


Figure 5(g): [A=11]

Figure 5: Output Image set 2

6. Video Processing

The above mentioned image enhancement is used in video processing. The videos we worked with were in the YUV (4:2:2) format. So, the Y-component of the image was only varied keeping others unchanged.
The subsampling scheme is commonly expressed as a three part ratio (e.g. 4:2:2), although sometimes expressed as four parts (e.g. 4:2:2:4). The parts are (in their respective order):

- *Luma* →horizontal sampling reference (originally, as a multiple of 3.579 MHz in the NTSC television system)
- *Cr* →horizontal factor (relative to first digit)
- *Cb*→ horizontal factor (relative to first digit), except when zero.

7. Application

This kind of *Image Enhancement* can be used in various fields like medical, surveillance, media etc.
Some of the applications of these concepts can be in:

- Low Vision Aid
- Medical Imaging
- Security Surveillance

SECTION II

Shot Detection by Feature Extraction of Co-occurrence matrixes

1. Introduction

Keeping in pace with the recent internet boom, the ever-expanding assemblage of multimedia material is on the brink of inundating us with a ponderous accumulation of unorganized digital video data. This final hour seems to be approaching us with inexorable certainty, when browsing a digital video library for video retrieval will become a more tedious work..To assuage this mammoth task of finding a needle in a haystack, significant research efforts are being devoted in effective retrieval and management of visual information. Each video sequence can be considered as a set of still images and the number of images in such a set is in a range of hundreds or thousands or even more. With such a huge volume of data in hand organising them effectively and retrieving information from them efficiently requires better representational formats. Much ongoing research is centered to locate meaningful information and extract knowledge from video documents. It is well-understood that better the representation scheme of video contents, faster and accurate will be the retrieval of data. One of the good representation schems is to index the video data i.e. look for information from the indexed data. Indexing videos manually is a time consuming venture and is also impractical. However utilizing various image processing techniques, computer scientists came up with the idea of shot detection which is the convention to group frames into shots. Thus, a shot designates a contiguous sequence of temporal video frames recorded by an uninterrupted camera operation. Video shot boundary detection algorithms need to perform under the constraint of camera motion, object motion and deviant lighting conditions. Again, video shot boundaries vary in appearances; they can be abrupt temporal change (hard cut), smooth temporal change (fade and dissolve) and wipe.

This project deals with detection of abrupt temporal changes (hard cuts) within a video data. In contrast to the existing threshold-dependent algorithms for this purpose, our methodology first extracts a set of features using co-occurrence matrices and then implements a training paradigm.

2. Co-occurrence Matrix

A co-occurrence matrix, also referred to as a co-occurrence distribution, is defined over an image to be the distribution of co-occurring values at a given offset. Mathematically, a co-occurrence matrix **C** is defined over an **n x m** image **I**, parameterized by an offset **(Δx,Δy)**, as:

$$C(i,j) = \sum_{p=1}^{n} \sum_{q=1}^{m} \begin{cases} 1, if\ I(p,q) = i\ and\ I(p+\Delta x, q+\Delta y) = j \\ 0, \qquad otherwise \end{cases}$$

The 'value' of the image originally referred to the grayscale value of the specified pixel. The value could be anything, from a binary on/off value to 32-bit color and beyond. It is also possible to define the matrix across two different images. Really any matrix or pair of matrices can be used to generate a co-occurrence matrix, though their main applicability has been in the measuring of texture in images, so the typical definition, as above, assumes that the matrix is in fact an image.
It is notable, that the **(Δx,Δy)** parameterization makes the co-occurrence matrix sensitive to rotation. We choose one offset vector, so a rotation of the image not equal to 180 degrees will result in a different co-occurrence distribution for the same (rotated) image. This is rarely desirable in the applications co-occurrence matrices are used in, so the co-occurrence matrix is often formed using a set of offsets sweeping through 180 degrees (i.e. 0, 45, 90, and 135 degrees) at the same distance to achieve a degree of rotational invariance.

3. K-Means Clustering

K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a clever way as different locations causes different results. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest

centroid. When no point is pending, the first step is completed and an early groupage is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more.
Finally, this algorithm aims at minimizing an *objective function*, in this case a squared error function. The objective function is:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} |x_i^{(j)} - c_j|^2$$

Where $|x_i^{(j)} - c_j|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j, is an indicator of the distance of the *n* data points from their respective cluster centres.

4. Proposed Methodology

Within a video, first a feature window is sought consisting of three consecutive frames $n_1, n_2 \; and \; n_3$, where n_3 is the pivot frame. Then there intermediate pixel-wise difference is calculated as $d_{1,2} \; and \; d_{2,3}$. These differences are calculated in the R, G, B planes separately. Now, with these differences in hand, we will calculate some features for determining whether the pivot frame is a shot or not. But before that, to make the determination more accurate, we compute co-occurrence matrixes at different pixel distances of the two difference matrixes→ $d_{1,2} \; and \; d_{2,3}$. The co-occurrence matrixes are calculated at $0°, 45°, 90° \; and \; 135°$. The features calculated were namely difference, energy and contrast.

- Moment → $\sum_i \sum_j i \times j \times f(i,j)$
- Energy → $\sum_i \sum_j f^2(i,j)$
- Contrast → $\sum_i \sum_j (i-j)^2 \times f(i,j)$

After feature calculation of each co-occurrence matrix of the two difference matrixes we get a two vectors of $4 \times 3 = 12$ values for each difference matrix, $d_{1,2} \; and \; d_{2,3}$ in each plane R,G and B. The vectors are averaged and mean vector is calculated. Now, the distance between these two vectors will give us the feature vector consisting of two values. When there is a shot, $d_{1,2}$ and $d_{2,3}$ will differ with a much larger value than, when there is no shot. This is because, if there is a shot at n_3, n_2 and n_3 will be way different from each other. So, gliding the created window through all the frames of the video gives a matrix $m \times n$ where *n=12 and m= number of frames.* In this matrix, the shot frames should have higher values than the non-shot frames. So, plotting this matrix will give peaks at the shot-frames.
One such plot for moment of 0° co-occurrence matrix for a particular video is as follows:

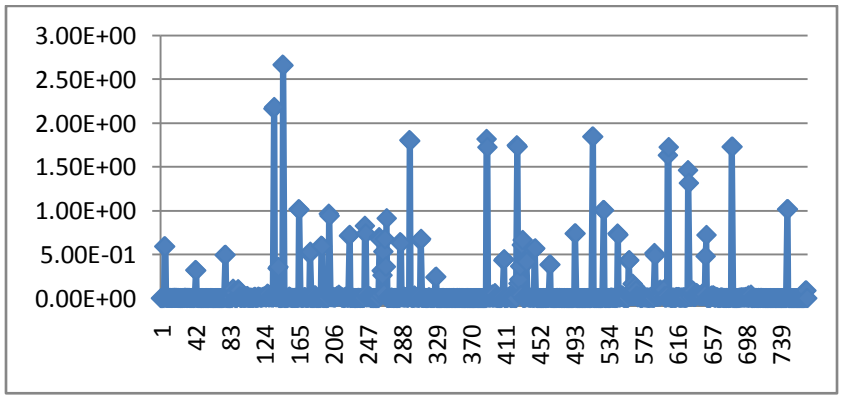

Figure 6: Graphical representation of shot and non-shot frames

After the feature vector for shot as well non-shot frames are obtained, we can separate them. Now, using KMEANS algorithm, we have made two clusters of shot frames and non–shot frames. After that, testing with the rest of data with the created cluster gives an average accuracy of 96.54%.

5. Improvements

With, the results satisfactory, there were some aspects of the algorithm that needed certain attention. First of all the feature, Energy was giving much redundant data, than what can be tolerated. Though local peaks were generated, globally, some peaks could not distinguish themselves from non peaks (or non-shot frames). Another fatal error, that came up was: for the window, $n_1, n_2\ and\ n_3$, we are getting a high value vector or a peak, but we are also getting a peak in the window $n_2, n_3\ and\ n_4$ where n_4 is supposed to be the corresponding frame for which checking is going on. n_4 is not a shot-frame, but due to the high differences, we are also getting peaks for the frames following the shot-frames. One of the additional features we experimented with was the Sobel Edge Detector.

We implemented edge detection in our algorithm in a certain modified way. Since, in some shots, we observed that, not whole but regions of images is changing (in reference to RGB components); we divide the images into quads and then applied Sobel edge detector on the quads.

Mathematically, the operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes, and one for vertical. If we define *A* as the source image, and G_x and G_y are two images which at each point contain the horizontal and vertical derivative approximations, the computations are as follows:

$$G_y = \begin{matrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{matrix} * A \qquad \text{and} \qquad G_x = \begin{matrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix} * A,$$

where $*$ here denotes the 2-dimensional convolution operation.
The *x*-coordinate is here defined as increasing in the "right"-direction, and the *y*-coordinate is defined as increasing in the "down"-direction. At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using:

$$G = \sqrt{G_x^2 + G_y^2}$$

Using this information, we can also calculate the gradient's direction:

$$\theta = arctan\left(\frac{G_y}{G_x}\right)$$

where, for example, θ is 0 for a vertical edge which is darker on the left side.

According to our experiments we saw, the strength gave more reliable results than direction.

SECTION III

XDAIS-compliant API Implementation & the Multimedia Toolbox

1. Introduction

The TMS320 DSP Algorithm Standard™, also known as XDAIS, is part of TI's eXpressDSP™ initiative. The purpose of the standard is to reduce those factors that prohibit an algorithm from being easily integrated into a system without significant re-engineering. Many of the unknowns in such a situation relate to resource allocation and consumption on a DSP. Bugs often occur during system integration as a result of the algorithm designer's unfounded assumptions about the system into which the algorithm is to be integrated. The standard, therefore, focuses on a set of general rules and guidelines that should be applied to all algorithms. For those algorithms utilizing DMA, the IDMA interface must be implemented. In addition, all algorithms must comply with a memory management API, called IALG. xDM is an extension to the IALG interface standard. It defines and standardizes interfaces for multimedia codecs to ease integration and ensure interoperability. xDM is built over TI's well proven eXpress DSP Algorithm Interoperability Standard (xDAIS) specification.

The Multimedia Toolbox is an advanced video processing toolbox that has been designed as a modularized application (a dynamic library) so that its components may be used as plug-ins to other applications. The modules can be used for still images as well as videos (considering the video to be comprised of individual frames which are images). The project began with the implementation of XDAIS algorithm standard and later those standardized algorithms were integrated into the above-mentioned Multimedia Toolbox.

2. Algorithms

There were basically two sets of algorithms that were made xDAIS-compliant. The first set was for a DSP platform that utilizes the xDAIS as well as the xDM standard while the other set was only utilizing the XDAIS standard. The primary task was to decouple the procedures from any memory management operations, and port those memory related operations into the XDAIS module. Then, all these memory management operations are complied with the IALG interface. The xDM interface bridges between the xDAIS interface(IALG) and the algorithm specific interface(IMOD).

The algorithms worked upon are:

- Contrast Enhancement by Unsharp masking and Laplacian Transform
- Find Connected Component
- Sobel Edge Detection
- Median Filter

2.1 Contrast Enhancement by Unsharp masking and Laplacian Transform

This algorithm is the one depicted in SECTION 1(the Development Phase). The work aimed at enhancing the contrast quality of a transmitted video over television. Due to additive noise, the videos get degrade, and a solemn effort was made to improve its quality. Since, videos for television broadcasting are in YUV format, we have worked with algorithms that shall modify the graylevels of the picture. Other than the graylevel enhancement algorithms, we are also looking forward to intensify specific colors (RGB) within an image. The parameter variance for changing distribution is a little eccentric, with more change at the mid region of the scale than the extremes. The laplacian transform however gives less visibility at the beginning of the scale but brightens the image up the scale. The algorithms have added advantage of minimal processing and hence minimal time complexity.

2.2 Find Connected Component

A pixel p at coordinates (x, y) has four *horizontal* and *vertical* neighbors whose coordinates are given by: (x+1, y), (x-1, y), (x, y+1), (x, y-1). This set of pixels, called the 4-*neighbors* of p, is denoted by $N4$ (p). Each pixel is a unit distance from (x, y), and some of the neighbors of p lie outside the digital image if (x, y) is on the border of the image. The four *diagonal* neighbors of p have coordinates: (x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1) and are denoted by ND (p). These points, together with the 4-neighbors, are called the 8-*neighbors* of p, denoted by $N8$ (p).

To establish if two pixels are connected, it must be determined if they are neighbors and if their gray levels satisfy a specified criterion of similarity (say, if their gray levels are equal).For instance, in a binary image with values 0 and 1, two pixels may be 4-neighbors, but they are said to be connected only if they have the same value.

Let *V* be the set of gray-level values used to define adjacency. In a binary image, V= {1} if we are referring to adjacency of pixels with value 1. In a grayscale image, the idea is the same, but set *V* typically contains more elements. For example, in the adjacency of pixels with a range of possible gray-level values 0 to 255, set *V* could be any subset of these 256 values. We consider three types of adjacency:

- 4-*adjacency*. Two pixels p and q with values from *V* are 4-adjacent if q is in the set *N*4 (p).
- 8-*adjacency*. Two pixels p and q with values from *V* are 8-adjacent if q is in the set *N*8 (p).
- *m-adjacency* (mixed adjacency).Two pixels p and q with values from *V* are m-adjacent if
  - q is in *N*4(p), *or*
  - q is in *N*D(p) *and* the set has no pixels whose values are from *V*.

A (*digital*) *path* (or *curve*) from pixel p with coordinates (x, y) to pixel q with coordinates (s, t) is a sequence of distinct pixels with coordinates

$$(x_0 , y_0), (x_1 , y_1), \dots \dots . (x_n , y_n)$$

where $(x_0 , y_0) = (x, y)$, $(x_n , y_n) = (s, t)$,and pixels (x_i , y_i) and (x_{i+1} , y_{i+1}) are adjacent for $1 \leq i \leq n$. In this case, n is the *length* of the path. If $(x_0 , y_0) = (x_n , y_n)$, the path is a *closed* path.

Let *S* represent a subset of pixels in an image. Two pixels p and q are said to be *connected* in *S* if there exists a path between them consisting entirely of pixels in *S*. For any pixel p in *S*, the *set* of pixels that are connected to it in *S* is called a *connected component* of *S*.

2.3 Sobel Edge Detection

The operator calculates the *gradient* of the image intensity at each point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. The result therefore shows how "abruptly" or "smoothly" the image changes at that point , and therefore how likely it is that the part of the image represents an *edge*, as well as how that edge is likely to be oriented. In practice, the magnitude (likelihood of an edge) calculation is more reliable and easier to interpret than the direction calculation. Mathematically, the gradient of a two-variable function (here the image intensity function) is at each image point a 2D vector with the components given by the derivatives in the horizontal and vertical directions. At each image point, the gradient vector points in the direction of largest possible intensity increase, and the length of the gradient vector corresponds to the rate of change in that direction. This implies that the result of the Sobel operator at an image point which is in a region of constant image intensity is a zero vector and at a point on an edge is a vector which points across the edge, from darker to brighter values.

Since the intensity function of a digital image is only known at discrete points, derivatives of this function cannot be defined unless we assume that there is an underlying continuous intensity function which has been sampled at the image points. With some additional assumptions, the derivative of the continuous intensity function can be computed as a function on the sampled intensity function, i.e. the digital image. It turns out that the derivatives at any particular point are functions of the intensity values at virtually all image points. However, approximations of these derivative functions can be defined at lesser or larger degrees of accuracy.

Mathematically, the operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes, and one for vertical. If we define *A* as the source image, and G_x and G_y are two images which at each point contain the horizontal and vertical derivative approximations, the computations are as follows:

$$G_y = \begin{matrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{matrix} * A \quad \text{and} \quad G_x = \begin{matrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix} * A ,$$

where $*$ here denotes the 2-dimensional convolution operation.

The *x*-coordinate is here defined as increasing in the "right"-direction, and the *y*-coordinate is defined as increasing in the "down"-direction. At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using:

$$G = \sqrt{G_x^2 + G_y^2}$$

Using this information, we can also calculate the gradient's direction:

$$\theta = arctan\left(\frac{G_y}{G_x}\right)$$

where, for example, θ is 0 for a vertical edge which is darker on the left side.

2.4 Median Filter

Order-statistics filters are nonlinear spatial filters whose response is based on ordering (ranking) the pixels contained in the image area encompassed by the filter, and then replacing the value of the center pixel with the value determined by the ranking result. The best-known example in this category is the *median filter*, which, as its name implies, replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel (the original value of the pixel is included in the computation of the median). Median filters are quite popular because, for certain types of random noise, they provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters of similar size. Median filters are particularly effective in the presence of *impulse noise*, also called *salt-and-pepper noise* because of its appearance as white and black dots superimposed on an image.

The median, j, of a set of values is such that half the values in the set are less than or equal to j, and half are greater than or equal to j. In order to perform median filtering at a point in an image, we first sort the values of the pixel in question and its neighbors, determine their median, and assign this value to that pixel. For example, in a 3×3 neighborhood the median is the 5th largest value, in a 5×5 neighborhood the 13^{th} largest value, and so on. When several values in a neighborhood are the same, all equal values are grouped. Thus, the principal function of median filters is to force points with distinct gray levels to be more like their neighbors. In fact, isolated clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than $\frac{n^2}{2}$ (one-half the filter area), are eliminated by a $n \times n$ median filter. In this case “eliminated” means forced to the median intensity of the neighbors. Larger clusters are affected considerably less.

3. The TMS320 DSP Algorithm Standard

With the advent of many telecommunication, imaging and video standards, there has been a considerable growth in developing, marketing, and distributing commercial off-the-shelf (COTS) algorithms. Since today's applications often require the use of several such COTS algorithms, it is feasible to conceive of DSP applications that can provide the infrastructure that enable multiple standard algorithms to operate on a single platform. To support this type of environment, TI has put in place an infrastructure namely, the TMS320 DSP Algorithm Standard.

3.1 Terminology

Here we have defined some of the most common terms that we encounter when discussing the TMS320 DSP Algorithm Standard.

- Algorithm: A module of code that consumes a data stream, processes it, and outputs a resultant stream. Examples include vocoders, modems, audio compression, video decompression, etc. Reference Framework: The "glue" code that holds together the drivers, the algorithms, resource managers, and DSP kernel. Reference Frameworks start out as application-agnostic. Upon the addition of application-specific algorithms, the Framework takes on an application-specific nature.
- DSP Kernel: A low-level software layer that provides hardware abstraction and manages low level physical resources. It provides threading, interrupt support, pipes, signals, and several other functions. In addition, DSP/BIOS offers data logging and statistical accumulation that enable real-time analysis of the system.
- Application: The definition depends upon the use of some or all of the other components. If a customer writes all the code from scratch including a kernel, algorithms, and a framework, then the entire software system may be described as the application. However, in an environment where DSP/BIOS, a reference framework, and COTS algorithms have been deployed, the application programmer may see no further down into the system than the APIs to the controlling framework.

3.2 Description of the eXpressDSP Elements

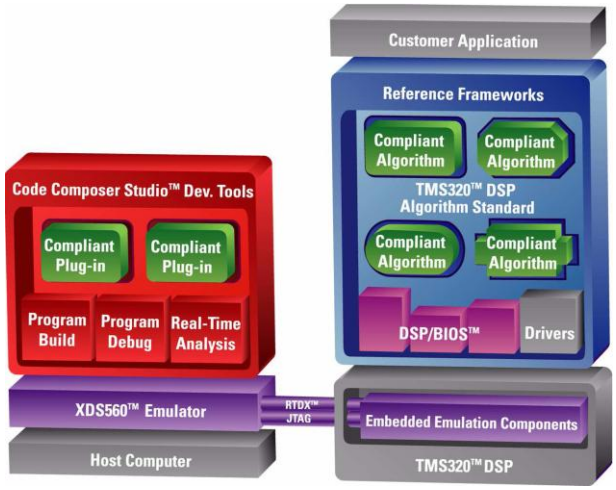

Figure 7: eXpressDSP Elements

The left-hand side of the diagram shows the Code Composer Studio Development Tools environment. The right-hand side of the diagram represents the target DSP in the eXpressDSP environment. It assumes the use of the DSP/BIOS real-time kernel (shown as purple components) with the possible inclusion of threads and tasks as required by the system designer. It also assumes that the system is built upon a Reference Framework (shown as the surrounding blue box).

Algorithms that have been written to comply with XDAIS, referred to as eXpressDSP-compliant, are shown here as the green shapes "plugging" into the Reference Framework. The key here is the "sockets" that the algorithms plug into. It is the separation between the plugs and sockets that defines XDAIS.

XDAIS focuses on the interfaces between the algorithm and the rest of the system, rather than the ability of algorithm writers to exploit their individual talents to achieve their goals of fastest, smallest, and cheapest. Essentially, the core of the standard focuses on an abstraction of DSP resource management away from the algorithms themselves. Typically, resources on a DSP refer to memory usage and placement, along with I/O control such as the use of DMA channels, and possibly the use of key control registers. When only a single algorithm runs on the DSP, one can make broad assumptions about the use of the DSP resources. Even in the case of a multiple-channel instantiation of that single algorithm, provided basic re-entrancy exists, the system should run just fine. However, when several algorithms are combined, problems may occur; for example, when one algorithm assumes the use of certain resources that are then "stolen" or "borrowed" by another algorithm in real time. It can be very difficult to pinpoint the source of a problem if the first algorithm was never designed to run in such a way and then doesn't perform to its specification, or worse, performs sporadically.

3.3 Implementation

All XDAIS algorithms **must** implement the IALG interface. XDAIS algorithms that **want** to utilize the DMA resource must implement the IDMA2 interface and use the ACPY2 interface provided by the client to request DMA services. Modern component programming models support the ability of a single component to implement more than one interface. This allows a single component to be used concurrently by a variety of different clients. In addition to a component's concrete interface (defined by its header) a component can, for example, support a trace interface that allows an in-field diagnostics subsystem to start, stop, and control the acquisition of data showing the component's execution trace. If all traceable components implement a common abstract trace interface, tools and libraries can be written that can uniformly control the generation and display of trace information for all components in a system. Support for multiple interfaces is often incorporated into the development environment using code wizards, the programming language itself, or both. Since the standard only requires the C language, the ability of a module to support multiple interfaces is awkward.

However, several significant benefits make this approach worthwhile. A vendor may opt to not implement certain interfaces for some components. New interfaces can be defined without affecting existing components, and partitioning a large interface into multiple simpler interfaces makes it easier to understand the component as a whole.

3.4 The IALG Interface

All algorithms in order to define their memory resource requirements and enable efficient use of on-chip data memories by client applications are required to implement the IALG interface. This means that the algorithm must implement every function defined in the IALG_Fxns structure (and assigned to the appropriate field in this structure). The functions defined in IALG_Fxns fall into several categories. Removing memory allocation from the algorithm complicates instance object creation. In order for an algorithm to be used in a variety of applications, decisions about memory overlays and preemption must be made by the client rather than the algorithm. Thus, it is important to give the client as much control over memory management as possible.
Some of the memories managing functions are:

- algAlloc() → let the algorithm to communicate its memory requirements to the client
- algInit()→let the algorithm initialize the memory allocated by the client
- algFree( ) → let the algorithm communicate the memory to be freed when an instance is no longer required.

This enhancement provides a simple mechanism for sharing run-time relocatable read-only look-up tables. Once an algorithm instance object is created, it can be used to process data in real-time.

Some other functions of the IALG template are:

- algActivate( )→ provides a notification to the algorithm instance that one or more algorithm processing methods is about to be run zero or more times in succession.
- AlgDeactivate( ) → provides a notification before reusing any of the instance's scratch memory
- algControl( ) → provides a standard way to control an algorithm instance and receive status information from the algorithm in real-time.
- algMoved( ) → allows the client to move an algorithm instance to physically different memory.

Algorithm Parameters and Status

When algorithm instances are created, the client can pass algorithm-specific parameters to the algAlloc( ) and the algInit( ) methods. The following figure summarizes the only valid sequences of execution of the IALG_Fxns functions for a particular algorithm instance.

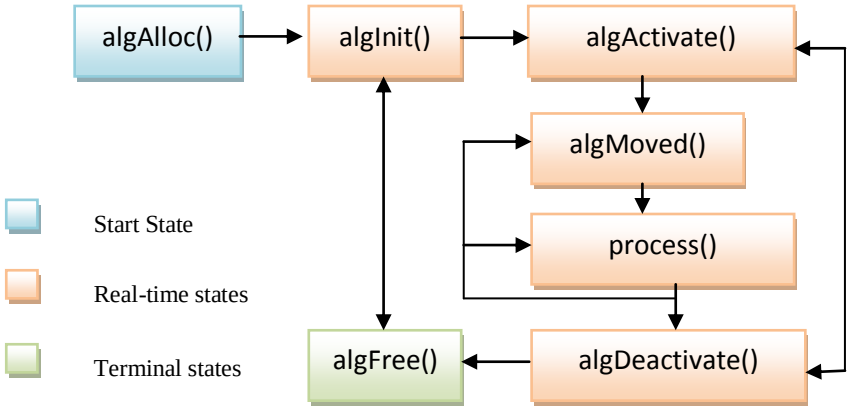

Figure 8: IALG Interface Function Call Order

4. The xDAIS-DM (Digital Media) standard

4.1 Goals of the standard

The goals of this standard include:

- Enable plug and play architecture for multimedia codecs across various classes of algorithms and vendors.
- Enable faster time to market for multimedia products such as, digital cameras, cell phones, set-top boxes, and portable multimedia players.

- Provide a standard interface based on given class of multimedia codecs (for example, audio, video, image, and speech).
- Define common status and parameters based on given class of multimedia codecs.
- Flexibility of extension of custom functionality.
- Low overhead of interface.
- Reduce integration time for system developers.

4.2 Relationship Between xDM and xDAIS

xDM is an extension to the IALG interface standard. It defines and standardizes interfaces for multimedia codecs. The relationship between xDM and xDAIS is depicted

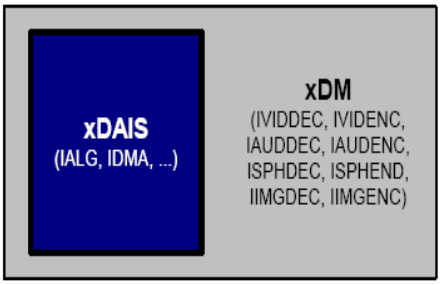

Figure 9: xDM Interface

As shown in the figure, xDM extends the xDAIS standard. xDM defines eight generic interfaces for the following categories.

- IVIDENCx - Generic interface for video encoders
- IVIDDECx - Generic interface for video decoders
- IAUDENCx - Generic interface for audio encoders
- IAUDDECx - Generic interface for audio decoders
- ISPHENCx - Generic interface for speech encoders
- ISPHDECx - Generic interface for speech decoders
- IIMGENCx - Generic interface for image encoders
- IIMGDECx - Generic interface for image decoders

4.3 xDAIS Interfaces

As per xDAIS, the given algorithm extends the standard IALG interface to the IMOD interface (algorithm specific). The IMOD interface provides basic functionality of the algorithm, while the IALG interface takes care of the memory management. xDAIS does not define the IMOD interface. The algorithm implementer must define the IMOD interface based on his requirements. For example, in case of MP3, the algorithm will implement the IMP3 interface. This interface is kept totally open. The application talks to the codec library via the IMOD interface. Optionally, an application can directly talk to the codec library via the MOD interface, which provides high level functionality. The current xDAIS interface from algorithm to application is depicted belowin Figure 4. The xDM standard adds a new interface as in Figure 5. The algorithm uses one of eight predefined standard interface called xDM. This interface is a superset of the IALG interface.

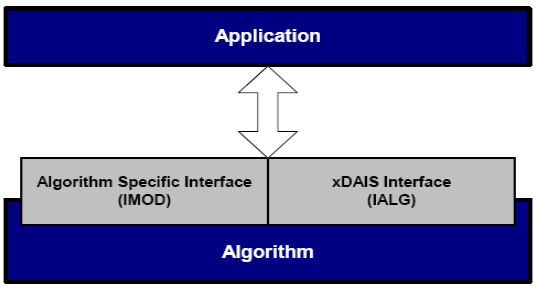

Figure10: xDAIS Standard

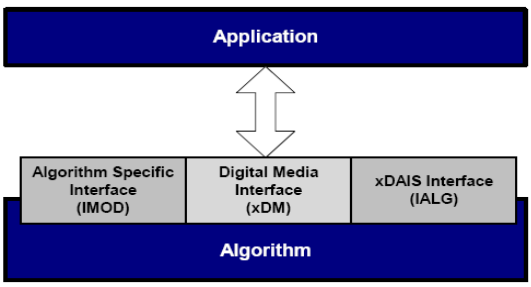

Figure 11: xDM Standard

5. The Multimedia Toolbox

The advanced video processing toolbox has been designed as a modularized application (a static library) so that components of it may be used as plug-ins to other applications. The modules can be used for still images and videos also (considering the video to be comprised of individual frames which are images).

The basic features of this multimedia toolbox are:

- An image format converter that will take an image in a specified format as input (e.g. JPEG) and save the image in any other specified format (e.g. BMP).
- An image color converter that will take an image in any specified color format as input (e.g. RGB), and convert and save the image in any other specified format (e.g. any of the three YUV formats viz. 4:2:0, 4:2:2 and 4:4:4).
- Image processing utilities which can be used in different image and video processing applications. Some of this utilities can be:
    - Noise Removal
    - Edge Detection
    - Morphology
    - Region Growing
    - Connected Component Analysis
    - Histogram Equalization

The system has a three tier structure:

a. **Top layer**: Decision making system
b. **Middle layer**: Decision support system
c. **Lower layer**: Video processing toolbox

The underlying assumptions for this system are:

- The object to be identified should occupy at least 10% of the image resolution
- Video toolbox will work as Lowest layer of the proposed 3-tier architecture
- Middle layer will provide the target video sequence as well as the object to be identified to the lower layer

In order to achieve the plug and play nature of the multimedia toolbox, the algorithms are converted into XDAIS- complaint APIs. This results in better memory management and performance on an embedded platform.

The Multimedia Toolbox can have the following field of applications:

- Identification of human object
- Identification of Specialized object
- Identification of geographical objects

SECTION IV

Missing Medical Instrument Identification

1. Introduction

1.1 Scope

The scope of the work consists of building a demo application for detecting missing instruments in an instrument case. The purpose of this demo is to develop an automated image processing software system to detect missing instruments from a instrument case.
The solution will consist of the following features:

- Reading the RFID Tag
- Taking an image of the instrument case under test
- Automated inspection of the instrument case
- Storing results in a database for future reference

1.2 Business Perspective

Identifying the misplaced or missing instruments have been a challenge for Medical Device Manufacturer at the following location/points

- When instruments are kept in a case and the case is ready to ship to Hospital or Distributor from Manufacture's warehouse.
- Verifying and checking the instruments before and after surgery.
- Analyzing the instruments health (if possible) if instruments are used many times
- Verifying and checking the case with instruments when damaged instruments are replaced

The proposed image processing technique with RFID would help to identify the instruments and improve the process related to instruments tracking and management.

1.3 Assumptions

The following assumptions are taken into account while designing the proposed system:

- Image of the individual instrument will have a distinct contrast difference with the background
- Image of empty case is available
- Image of case with all desired instruments placed properly is available
- Instruments in the case will have a distinct contrast difference with the background
- Prior knowledge of the different type of instruments that should be ideally available in a case is available to the system
- Instrument case will have a fixed position with reference to the camera
- Every instruments have a fixed position inside the case

1.4 Overview of the System

MMII mainly deals with scanning a real time object an identifying any discrepancies with the previously stored template. It mainly contains the following sections, namely the Image capture Section, the RFID tag & sensor section and the Image processing section. Apart from these, the main principle in place here is MACHINE VISION.
The Machine Vision operation can be viewed as a conglomeration of the following:→

- Sensor
- Light (Optics)
- Signal (OUT) } Hardware
- Lens
- Image Processing → Software

The Image Capture Section takes care of the Light and Lens. The particular importance out here is the ambient light atmosphere and the overall illumination of the object. Since the software processing entirely depends upon the captured image, its quality, whatever is should be consistent. For better results, however, the live object needs to be properly illuminated. The image of the live object (the one that is under examination) is processed in the image identification software. A master template for each kind of object is present in the software. The image capturing conditions for the live object and the master template should be identical.

The RFID sensor section works in a twofold way: The sensor first senses the presence of a live object at the image capture arena, where it is to be imaged, and signals out to the Image Capture setup, when the object is at place; Now the Image Capture Section captures the image of the live object and sends it for processing to the Image Processing Software. Meanwhile the RFID tag associated with that object is also passed to the image processing software. The instruments can be placed in a static arrangement or in a dynamic one. While the static arrangement requires manual labor for complete simulation, the dynamic arrangement, also known as Conveyer belt can complement the automation.

The Image Processing Section carries out the software tasks required. Main idea being matching the live object's image to the corresponding template, other subsidiary tasks such as designing the GUI, enumerating missing instruments as well as locating them are also taken care of.

The overall MII process sequence:→

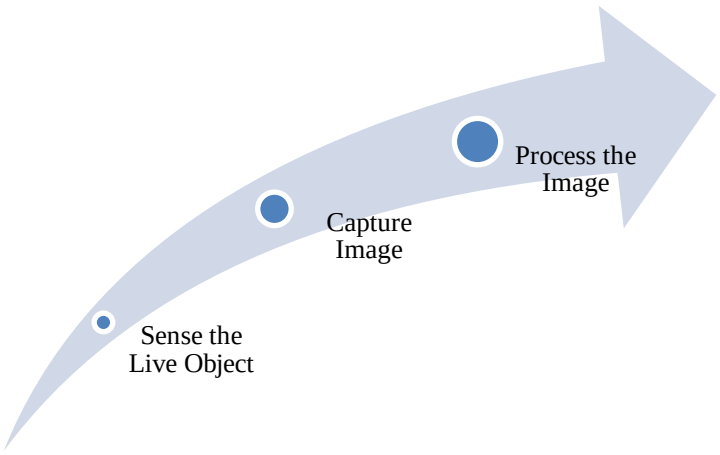

Figure 12: MMII Process Sequence

2. Overall Architecture

2.1 High Level Architecture of the system

This proposed demo software will have the following components:

- Instrument Inspection module: This module shall perform the inspection of the instrument case by image processing and pattern matching algorithms and will detect the missing components (if any) from the case.
- GUI: This module will have interactive user interface for user verification and for instrument inspection.
- Database: There will be three database for the demo application:
  - Database for storing instrument case number
  - Database for storing the region of interest (ROI) and other features of the instruments to be searched
  - Database for storing the inspection results as part of the management information system

The high level architecture of the proposed system is explained below:

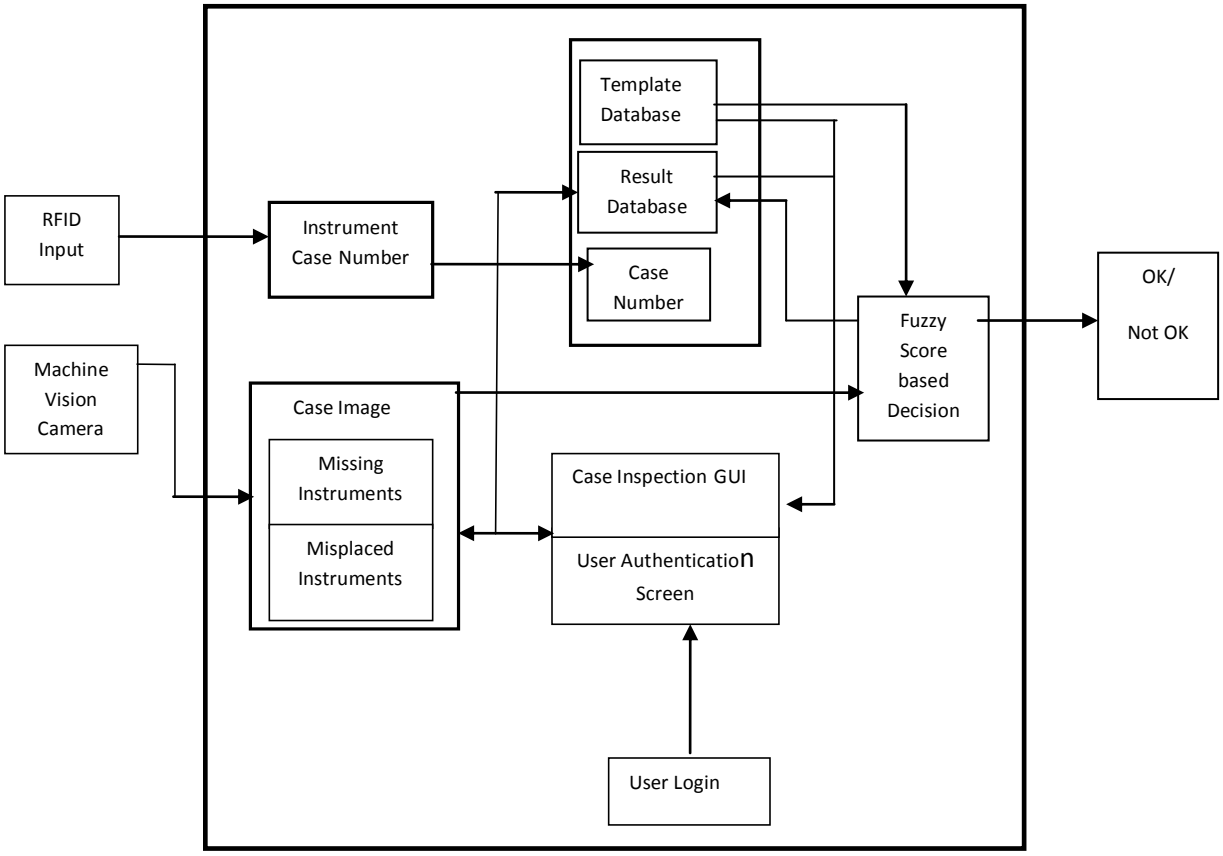

Figure 13: High Level Architecture

3. Functionality Description

3.1 Instrument Inspection Module

This module shall perform the following operations:
Once the start button is pressed, the input for the instrument case shall be obtained from the RFID database. It is assumed that the instrument cases shall have unique identification number. For the demo purpose, the identification number is retrieved from a text database. However, in future it can be integrated online directly with RFID output.

3.2 MMII Software Outlook

After fixing the resolution at which the complete procedure will be undertaken the following sequence is intended to follow. The Master Template view (ideally full case) of the instrument box already existing in the database should be opened with the IRFANVIEW. Following this, each and every instrument present in the box is to be selected with the mouse pointer; Care should be taken that the selected view gives minimal coverage i.e. anything unnecessary is excluded but everything necessary is included. The following screenshot shows precisely what we are trying to achieve out here:

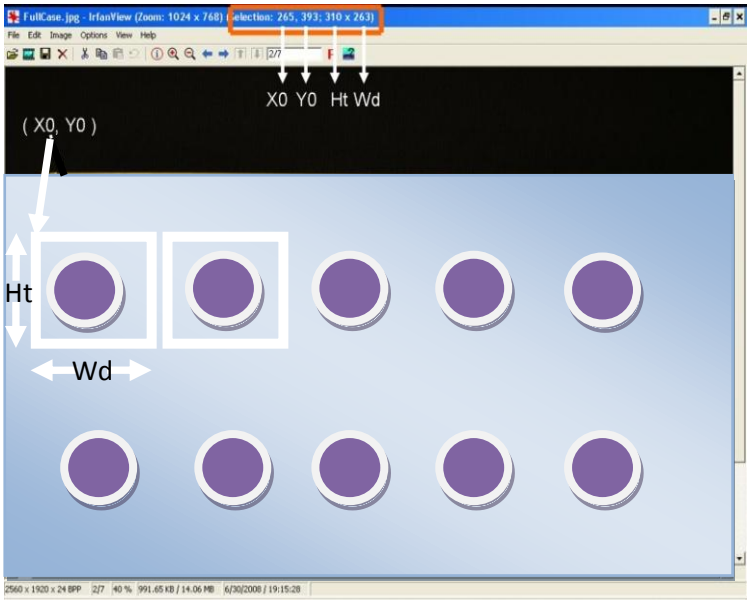

Figure 14: MMII Master Template

In the screenshot, there are two white-line bounded regions. They highlight only two of the ten instruments present in the instrument box. Now, narrowing down to the first bounded region, we can see 4 discrete data and they can be defined like this:

- X0—the x-coordinate of the 1st instrument,
- Y0—the y-coordinate of the 2nd instrument,
- Ht—the height of the bounded region and
- Wd—the width of the bounded region.

The specific values for these entities can be seen in the top center, bounded by a red rectangle. So, from this instrument box, we shall get 10 different data sets, each set having 4 different data entries. They are to be noted down, and being wary will pay off in avoiding future recalculations.

Now, we are done with the first phase. The next phase is basically about calculating normalized coordinates that will help us in pattern recognition between the Master template and the Live view. For this we have the ROI EXCEL SHEET. Here goes the screenshot:

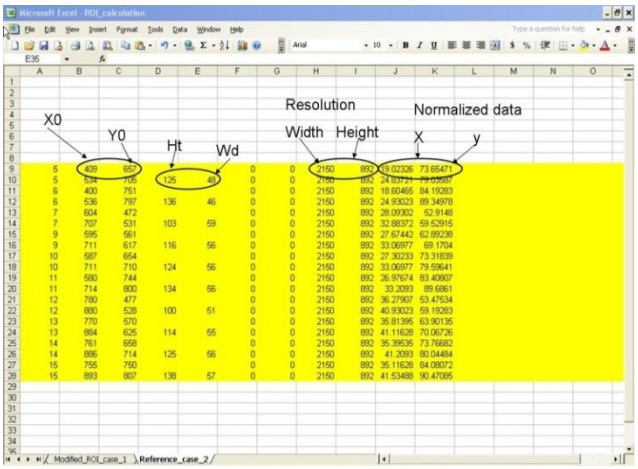

Figure 15: ROI Calculation Sheet

We are already aware of X0, Y0, Ht, and Wd. Now let us see what the rest of the tagged values mean. The resolution as you will recall is the first thing we fixed during the whole procedure. And the normalized data are the ones we are working for. We will write the corresponding values in every field and note down the normalized coordinates from the sheet.
Armed with these, we are ready to update the CASE_DATABASE file. We will be updating the normalized coordinate values out here corresponding to their RFID id numbers. Here goes the screenshot:

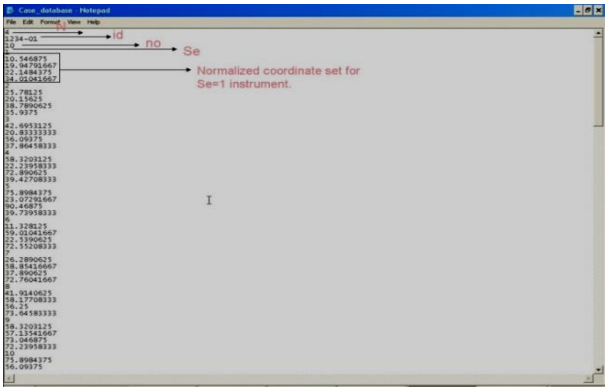

Figure 16: Case Database file

Now, let us browse the data identifiers present in the above screenshot. They can be defined as followed:

- N—the number of instruments in the database
- id—the RFID tag for the particular instrument set
- no—the number of instruments present in the set (e.g. the instrument set considered here holds 10)
- Se—the position id of the instrument whose normalized coordinates are to follow.

With this database updated and the Master template accordingly in place, we are ready to check the live view hereafter.

4. Working Principle

4.1 Overview

Images of the inspection case in any well known format come as the input to the system.
There will be two stages in the project:

1. Offline feature extraction of the individual instruments and storing in a template database as described in Figure 16
2. Online pattern matching of the instruments in the case with the features already stored as described in

3. Figure 17

In the first stage, the features of the instrument images extracted and are stored in a template database. Overview of the first stage of the system is given in Figure 16:

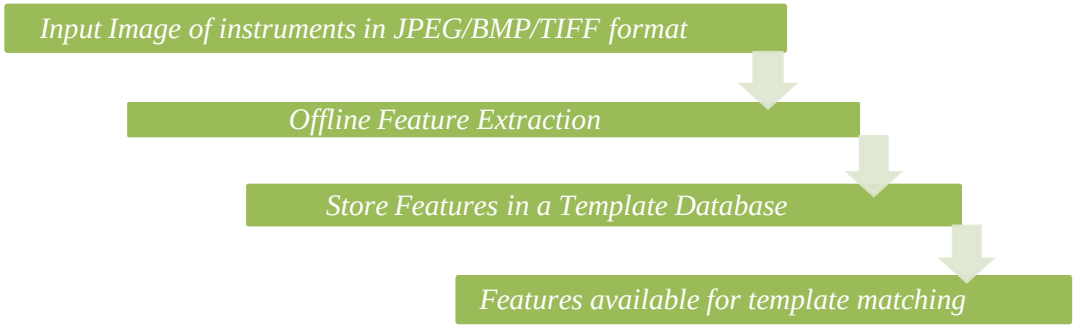

Figure 17: Offline Feature Extraction of Reference Images

The second stage is the feature/template matching based on the already stored features in the template database and the features extracted from the instruments in the case. As discussed in the assumption section, it is assumed that prior knowledge of the instruments available in a particular case is available and also each instrument is kept in a fixed position inside the case. Based on the above two assumptions, the instrument case number is available from the RFID output and based on the instrument case number, the features for the instruments are retrieved from the template database. After performing the template matching and based on pre-fixed criteria for OK or not OK, result is stored in the result database and also displayed in the GUI.

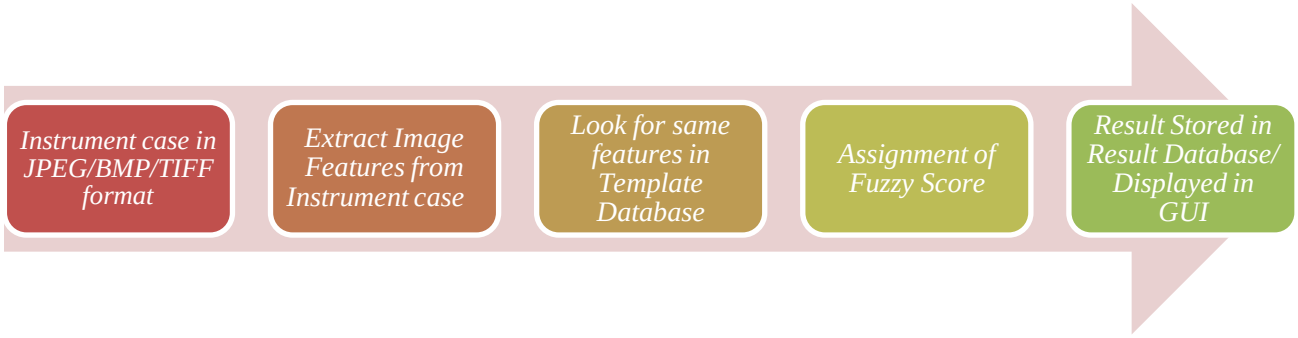

Figure 18: Online Inspection of Instrument Cases

4.2 Logical Conclusion

4.2.1 RFID

Radio-frequency identification (RFID) is an automatic identification method, relying on storing and remotely retrieving data using devices called RFID tags or transponders.
An RFID tag is an object that can be applied to or incorporated into a product, animal, or person for the purpose of identification using radio waves. Most RFID tags contain at least two parts. One is an integrated circuit for storing and processing information, modulating and demodulating a (RF) signal, and other specialized functions. The second is an antenna for receiving and transmitting the signal.

4.2.2 Processing

Here the processing corresponds to the comparison between the master template and the live object. The master template contains the complete set of instruments while the live object is analyzed and missing instruments (if any) are identified. The luminance property of instruments is utilized to distinguish between an empty holder and an instrument (Segmentation). Another algorithm called edge linking method (Hough Transform) is also implemented. This utilizes the fact that the instrument is supposed to have a smoother terrain than the case.

4.2.3 Histogram Similarity

The probability of occurrence of gray level r_k in an image is approximated by Eq.(1)

$$p_r(r_k) = \frac{n_k}{n}, k = 0,1,2,\ldots\ldots,L-1 \qquad \textit{Eq.(1)}$$

where, *n is* the total number of pixels in the image, n_k is the number of pixels that have gray level r_k, and *L* is the total number of possible gray levels in the image. A plot of probability p_r (k) versus r_k is called a *histogram*. Two set of

histograms, one for the Master Template and the other for the Live Object are compared out here, the threshold for comparison being their absolute difference.

$$diff = |\sum p_{r_1} - \sum p_{r_2}|$$

4.2.4 Hough Transform

This method of edge linking determines whether or not a set of edges lie on a line (or a specified curve). This method then links the edges by producing the line or curve. For this edge detection, it uses a parameter space called the Hough space. Each point (d, T) in Hough space corresponds to a line at angle T and distance d from the origin in the original data space.

4.2.5 Integration

The MII software simply implements the two above mentioned algorithms to compare the two sets of images. The threshold in the segmentation algorithm is so fixed; that the two clusters correspond to either the ideal case or the missing one. The Hough Transform implementation is primarily used to check the average number of edges detected in both the ideal cases and the missing cases. Then a threshold is fixed based on experimental outcomes.

Appendix 1

List of Figures