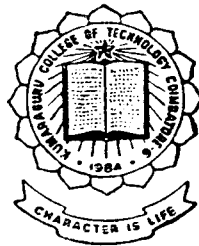


# Object Oriented Image Data Compression

P-297

DISSERTATION SUBMITTED IN PARTIAL FULFILMENT  
OF THE REQUIREMENTS FOR THE DEGREE OF  
MASTER OF COMPUTER APPLICATIONS  
OF BHARATIAR UNIVERSITY

By  
P. SENTHIL KUMAR  
9438MO204



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**Kumaraguru College of Technology**

COIMBATORE-641 006


June 1997


## CERTIFICATE

*This is to certify that this project work entitled*


**“OBJECT ORIENTED IMAGE DATA COMPRESSION”**

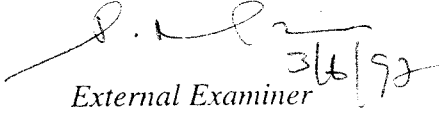
*Submitted to Kumaraguru college of Technology , Coimbatore (Affiliated to Bharathiar University ) in partial fulfillment of the requirements for the award of the Degree of Master Of Computer Applications is a record of original work done by Mr. P. Senthil Kumar, Reg No. 9438MO204 during his period of study in the Department of Computer Science and Engineering, Kumaraguru College Of Technology, Coimbatore under my guidance and this project work has not formed any basis for the award of any Degree/Diploma/Associateship/Fellowship of Similar title to any candidate of any University.*

  
Professor and Head

  
Staff in-charge

*Submitted for University Examination held on 02 / 06/97*

  
Internal Examiner

  
External Examiner

सरो उपग्रह केन्द्र

अन्तरिक्ष विभाग

भारत सरकार

वाई पत्तन मार्ग, विमानपुरा डाक घर,

गलूर-560 017, भारत

दूर: ' उपग्रह ' टेलिक्स: 0845-2325 और 2769

संख्या: 5266251



ISRO SATELLITE CENTRE

DEPARTMENT OF SPACE

GOVERNMENT OF INDIA

AIRPORT ROAD, VIMANAPURA P.O.,

BANGALORE-560 017. INDIA

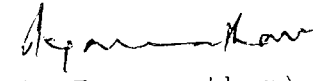
Telegram : UPAGRAH Telex : 0845 2325 & 2769

Telephone : 5 2 6 6 2 5 1

S.Jagannathan  
Scientist/Engineer  
Digital System Group

C E R T I F I C A T E

This is to certify that Mr.P.Senthil Kumar of Kumara Guru College of Technology, Coimbatore has successfully completed his project titled "Object Oriented Image Data Compression" under my guidance during the period from January 1997 to May 1997.

  
(S.Jagannathan)

Dedicated to my beloved parents

*Acknowledgement*

---

## ACKNOWLEDGEMENT

An endeavour over a long period can be successful only with the advice and support of many well wishers. I take this opportunity to express our gratitude and appreciation of all of them.

My heartfelt thanks to **Mr. Subramanian Head HRD, PPEG, Mr.U.N.Dass, Head DSG**, for having given us this opportunity to work in ISAC.

I express my gratitude to **Mr. R. Sesaiah Head Telemetry Division**, for providing me the facilities to carry out the project work. I also express my sincere thanks to **Smt. V.Nalanda Head Test and Simulation Section**, for her encouragement during the project.

I am bound to express my gratitude to **Dr. S.Subramaniam Principal K.C.T (Affiliated to Bharathiar University) Coimbatore**, for his constant encouragement throughout my course.

I wish to thank my Internal Guide **Prof. P. Shanmugham the Head, Department of Computer Science** for constantly encouraging me pursue new goals, ideas and being supportive throughout the tenure of my project.

I express my sincere thanks to **Mr.S.Jagannathan Engineer/Scientist, DSG** for providing me with the opportunity of working with a platform of my choice, for his inspiring advice, immensive help, whole-hearted support throughout the project. He is the brain behind this project who developed and maintained the schedules and ensured the missing pieces were found and put in place.

I would like to take this opportunity to thank all the Engineers and staff of DSG and in particular **Mr. Gangadhar Engineer/Scientist**, for providing the necessary softwares needed for the project and for his valuble technical advices.

Last but not the least I thank my friends for their comments and suggestions during the development of this project.

*P.Senthil Kumar.*

*Abstract*

---

## ABSTRACT

### *Existing System*

In Digital System Group (DSG) of ISRO Satellite Centre, a single package consisting of most widely used Image Compression algorithm has become a mandatory to study the compression ratios, quality, implementation risks on various sets of image data. Presently available compression algorithms are written C and Pascal in Unix/PC environment. Each algorithm is developed as separate project and working individually. The display and views of image facility are inadequate.

The existing compression algorithms, which are written in C and Pascal works as separately and there is no full pledged package. Which has feature of image manipulations. It is difficult to study the performance of more than one algorithm on any image at a time. The user has to run separately each algorithm and also presently the executable versions of algorithm occupy more space and not standardised.

### *Proposed System*

Keeping the above requirement, the proposed system is a complete package, which has the facility to implement all the algorithms in a single package. The new system must have friendly user interaction and user friendly. The provision for the display of the images, providing tools to study the image characteristics are the main aims of the proposed system.



*Contents*

---

# CONTENTS

Acknowledgement

Abstract

1 . Introduction

1.1 About the organisation

1.2 Problem specification

2 . Tools and Methodologies

2.1 OOPS

2.2 Microsoft Foundation Class

2.3 Visual C++

3 . Image Compression

3.1 Hauffman Coding

3.2 Arithmetic Coding

3.3 Differential Cosine Transformation

3.3 DCPM

4. System Design

5. System Flow

6 . Implementation

7 . Conclusion

7.1 Scope For Further Development

8 . Bibliography

Appendix

a) Screen Design

b) Sample Outputs

*Introduction*

---

## 1.1 About The Organisation

The Indian Space Programme was formally organised in 1972, when the Government of India setup the Space Commission with a view to promote the development and application of Space Technology and Space Science for the socio-economic benefits of the Nation.

Department of Space (DOS) is responsible for the execution of the activity in the country in Space applications, Space Technology and Space Sciences through the Indian Space Research Organisation (ISRO). The DOS/ISRO headquarters is located at Bangalore providing overall directions to the technical Scientific and Administrative functions of the eight centres given below:

- Vikram Sarabhai Space Centre (VSSC), Trivandrum, engaged in Design and Development of Satellite launch technology.
- Sriharikota Ranges (SHAR), Sriharikota, which is the main operational base of ISRO, fully equipped with sophisticated launching pad facilities.
- ISRO Satellite Centre (ISAC), Bangalore, engaged in design and development of communication, remote sensing and other satellite activities.
- Space Application Centre (SAC), Ahmedabad, which is ISRO's main Application R&D centre with activities in telecommunication and developing of pay loads for spacecraft's.
- Liquid Propulsion System Centre (LPSC), Mahendrigiri, engaged in launch vehicle propulsion systems and engine testing.

- ISRO Telemetry Tracking and command Network (ISTRAC), Bangalore, provide ground support for the tracking IRS data and satellite missions of ISRO.
- National Remote Sensing Agency(NRSA), Hyderabad, is engaged in developing and utilising modern remote sensing techniques and providing training and other operational supports to various end users.

### ***Digital System Group Profile***

The group has been organised into three divisions

1. Telemetry Division
2. Telecommand Division
3. Ground Systems Division

### ***Activities of group***

- Design, Development and fabrication of onboard digital electronics for TTC system for satellite.
- Development of Data handling system for space platforms and satellites specially for Remote sensing and Scientific satellites.
- Development of various ground support equipment's required for satellite missions including ground encoders, data processing system, etc.
- Advance R & D in the areas of Ttc, Data Compression Techniques, high density electronic circuitry like HMCs, ASICS, FPGAS etc.

## **1.2 Problem Specification**

### ***Existing System***

In digital System Group(DSG) of ISRO Satellite Centre, a single package consisting of most widely used Image Compression algorithm has become a mandatory to study the compression ratios, quality, implementation risks on various sets of image

data. Presently available compression algorithms are written in C and Pascal in Unix/PC environment. Each algorithm is developed as separate project and working individually. The display and views of image facility are inadequate.

The existing compression algorithms, which are written in C and Pascal works as separately and there is no full pledged package. Which has features of image manipulations. It is difficult to study the performance of more than one algorithm on any image at a time. The user has to run separately each algorithm and also presently the executable versions of algorithms occupy more space and not standardised.

### ***Proposed System***

Keeping the above requirement, the proposed system is a complete package, which has the facility to implement all the algorithms in a single package. The new system must have friendly user interaction and user friendly. The provision for the display of the images, providing tools to study the image characteristics are the main aims of the proposed system.

*Tools And Methodologies*

---

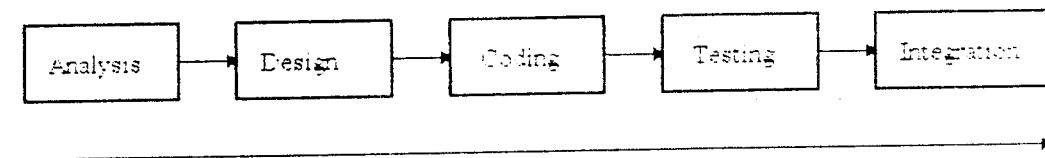
## 2 . Tools & Methodologies

### 2.1 Object Oriented Approach

In 1960's the generally poor state in software sparked a concentrated effort among scientists to develop a more disciplined, consistent style of programming in the result of the effort was the refinement of modular programming into an approach known as structured programming .

Structured programming relies on function decomposition, a topdown approach to program design in which a program is systematically broken down into components down to the level of individual subroutines. Eventhough structured programming has produced significant improvements in the quality of software over the years but its limitations are now painfully apparent .

One of the most serious problems is that it is rarely possible to anticipate the design of complicated design of completed systems before it is actually implemented . Here code and data are separate . Procedures define what is to happen to data but the two never become one . When addition or deletion must be made to the program code , often the entire program has to be reworked to include new routines.



*In the tradition structured design cycle , progress is linear*

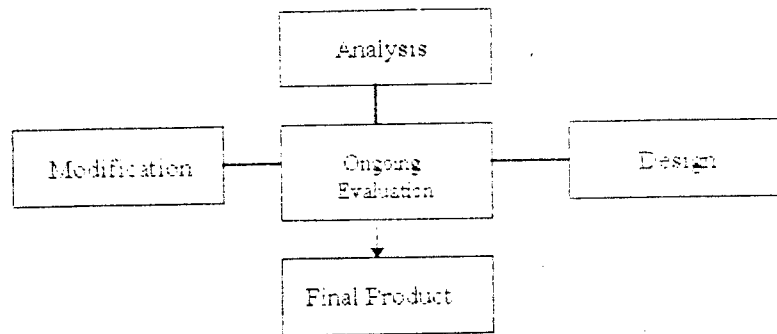
### Object Oriented Approach

Object Oriented technology is built upon a sound Engineering foundation whose elements were collectively call the object model . The object model encompasses



the principles of abstraction, encapsulation, modularity, hierarchy, typing, concurrency and persistence. In object model these elements are brought together in a synergic way .

The physical building block of the object model is module, which represents logical of classes and objects instead of subprograms as in earlier languages.



*Object Oriented design cycle involves an iterative approach throughout the design cycle  
( Booch Method )*

## OOA

Object Oriented analysis is a method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain.

Object oriented design is a method of design encompassing the process of object oriented decomposition and notation for deficiency both logical and physical as well as static and dynamic models of the and object abstraction to logically structure system.

OOP is a method of implementation in which programs are organised as co-operative of collections of objects, each of which represents an instance of some classes and whose classes are all members of a hierarchy of classes united inheritance relation ships.

Basically the products of OOA serve as the models from which use may start an OOP; the products of OOD can then be used as blue Print of completely implementing a system using OOP methods.

Major elements in object oriented Programming as follows:

- Abstraction
- Encapsulation
- Modularity
- Hierarchy
- Typing
- Concurrency
- Persistence

### *Abstraction*

An abstraction denotes the essential characteristics of an object that distinguish it from all kinds of an object and thus provide crisply defined conceptual boundaries relative to the perspective of the newer.

An abstraction focuses on the outside view of an object and so serves to separate an objects essential behaviour from its abstraction are

- *Entity Abstraction*

An object that represents a useful model of a problem domain or solution domain entity.

- *Action Abstraction*

An object that provide a generalized set of operations, all of which perform the same kind of function.

- *Virtual Machine Abstraction*

An object that groups together operations that are all and by some superior level of control, or operations that all one come junior level set of operations.

- *Coincidental Abstraction*

An object that packages a set of operations that have no relation to each other.

### *Encapsulation*

Abstraction and encapsulation are complementary concepts: abstraction focuses up on the desirable behaviour of an object, whereas encapsulation focuses up on the implementation that gives rise to its behaviour. Encapsulation is most often achieved through information hiding , which is the process of hiding all the secrets of an object that do not contribute to its essential characteristics, typically the structure o an object is hidden as well as the implementation of its methods.

Encapsulation is the process of compartment aligning the elements of an abstraction that constitutes its structure and behaviour, encapsulation serves to separate the contractual interface of an abstraction and its implementation.

### *Modularity*

Modularity is the property of a system that has been decomposed into a set of cohesive and loosely coupled modules.

In object oriented programming classes and objects form the logical structure of a system; we place these abstractions in the modules to produce the systems physical

architecture. In larger applications when many hundreds of classes are necessary. The use of modules essential to help manage completely.

### ***Hierarchy***

In most initial applications, we may find many more different abstractions that we can comprehend at one time. Encapsulation hides the view of abstractions, modularity gives a set of abstractions often from a hierarchy and by identifying these hierarchies make problem easy. Thus hierarchy is a ranking reordering of abstraction.

### ***Typing***

Typing is the enforcement of the class of an object, such that objects of different types may not be interchanged, or only in very restricted ways. Typing lets us express our abstraction so that the programming language can be made enforce design decisions.

### ***Concurrency***

Concurrency is the property that distinguishes an active object from one that is not active.

### ***Persistence***

Persistence is the property of an object through which its existence transcends time or space.

### ***Benefits Of OOPs***

- Encourages the reuse of software components.
- leads in systems that are more resilient to change.

- Reduces development risk.
- Appeals to the working of human coorganization

### ***Basic Concepts of OOPS***

- Objects
- Classes
- Inheritance
- Polymorphism
- Dynamic Binding
- Message passing

### ***Objects***

Objects are the basic run time entities on an object oriented system. They may represent a person, a place, a bank account, a table, of data or any item that the program must handle. They may also represent user defined data such as vectors, time and lists. An Object may include other Objects, for example a personal computer is an object . one of the object that makes up the computer CPU another object is its main memory and another is the I/O system. The I/O system subsystem also contains the objects, including the key board, the mouse and the display screen.

Objects Contain data and code to manipulate that data. Object contain data and member functions to manipulate that data. Object is an entity that can store data .send and receive messages. It is an instance of a class.

### ***Class***

Class is a group of Objects that share common properties and relationships. A class is the fundamental building block in C++. A class is a way to bind the data and its associated functions together. It allows the data to be hidden, if necessary.

from external use. When defining a class we are creating a new abstract data type that can be treated like any other built in data type. Generally a class specification has two parts

1. Class Declaration
2. Class Function definition

The class declaration describes the type and scope of it's member. It is similar to a struct declaration. The class function definitions Describe how the class functions are implemented.

The general form of a class declaration is

```
Class Classname
{
    Private:
variable declarations;
                function declarations;
    Public:
                variable declarations;
                function declarations;
};
```

The keyword class specifies that what follows is an abstract data of type classname. The functions and variables are collectively called members. The variables declared inside the class are known as data member and the functions are known as member functions. The member that have been declared as private can be accessed only from with in the class. The public members can be accessed from outside the class also.

### ***Inheritance***

A relationship between classes such that some of the member declared in one class are also present in the second class .The derived class inherits the members from the base class.

A base class is the class which has the overall outline of the domain of the problem. Inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification. In OOP, the concept of inheritance provides the idea of reusability. This means that we can add additional features to an existing class without modifying it. This is possible by deriving a new class from the existing one. The new class will have the combined features of both the classes. Here the old class is called the base class and new one is called the derived class.

The Types of the inheritance:

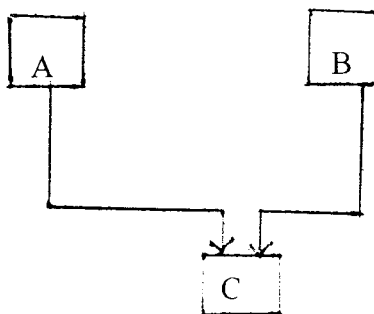
1. Single inheritance
2. Multiple inheritance
3. Hierarchical inheritance
4. Multi Level inheritance
5. Hybrid inheritance

**1. Single inheritance**



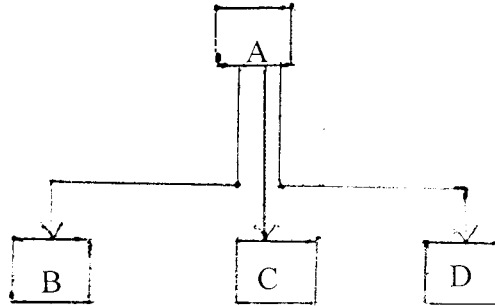
A derived class with only one base class is called Single Inheritance.

**2. Multiple Inheritance**



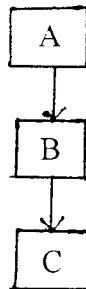
A derived class with several base class is called Multiple Inheritance.

### 3. Hierarchical Inheritance



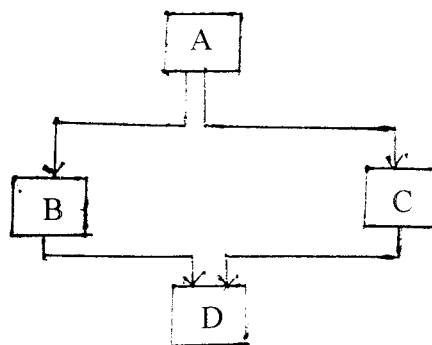
The traits of one class may be inherited by more than one class .this process is known as Hierarchical Inheritance.

### 4. Multilevel Inheritance



The mechanism of deriving a class from another derived class is known as Multilevel Inheritance.

### 5. Hybrid Inheritance





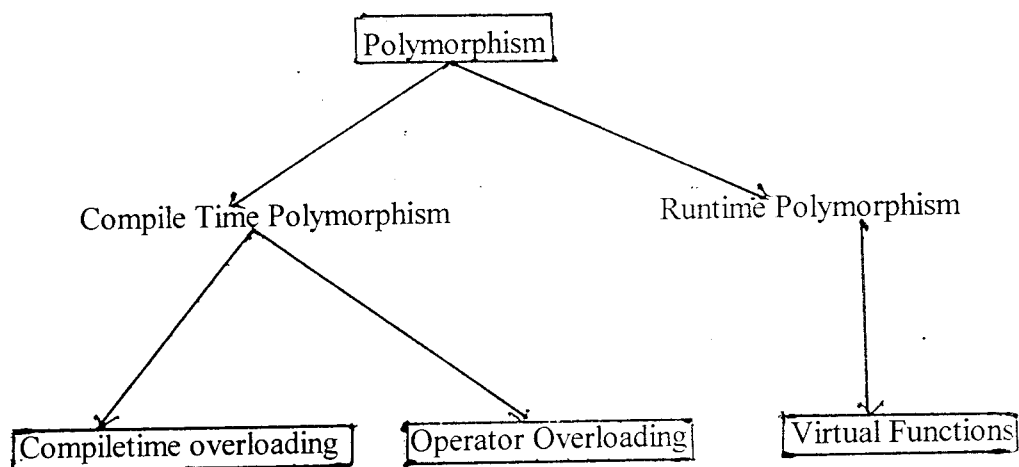
There could be a situations where we need to apply two or more type of inheritance to design a program i.e. the derived class using the two or more inheritance properties. This type of inheritance is called as Hybrid Inheritance.

### ***Polymorphism***

Polymorphism means the ability to make more than one form i.e. one name multiple forms. The ability to use an operator with different meanings .The concept of the polymorphism is also using in the function overloading and operator overloading.

Polymorphism is two types:

1. Compile time polymorphism
2. Runtime polymorphism.



### ***Compiletime Overloading***

Function Overloading and operator overloading are used compiled time overloading.

### ***Function Overloading***

we can assign the same name to two or more distinct functions .This process is called function overloading.

eg.    int Mul(int a, int b);  
      Float Mul(float x, float b);

These over loaded member functions are selected for invoking by matching arguments ,both type and number .This information is known to the compiler at the compile time and ,therefore ,compiler is able to select the appropriate function for a particular call at the compile time itself .This is called early binding or static binding or static linking . This is also known as compile time polymorphism.

### ***Operator Overloading***

It means assigning different meaning to an operations depending on the context.

e.g.,

The I/O operators << and >> are good examples of operator overloading .The built in definition of the << operator is for shifting of bits ,it is also used for displaying the values of various data types.

### ***Runtime polymorphism***

In runtime polymorphism the appropriate member function could be selected while the program is running. Let us consider an example that where the function name and prototype is the same in both the base and derived classes .In that case the function is not overloaded and therefore static binding does not apply .The compiler does not know what to do and defers the decision. At that time the appropriate member function could be selected while the program is running .This is known as runtime polymorphism. To achieve this the C++ supports the virtual functions.

### ***Virtual Functions***

When we use the same function name 'N' in both the base and derived classes the function in base class is declared as virtual using the keyword virtual

proceeding its normal declaration. When a function is made virtual C++ determines which function to use at run time based on the type of object pointed to by the base pointer, rather than the type of the pointer.

### *Dynamic binding*

The address of the function are determined at run time rather than compile time this is called dynamic binding. This is also known as late binding. It is associated with the polymorphism and inheritance.

### *Message Communication*

An object-oriented program consists of a set objects that communicate with each other. The process of programming in an object oriented language therefore involves the following basic steps.

- Creating Classes that define objects and their behaviour.
- Creating objects from class definitions.
- Establishing communication among objects.

Objects Communicate with one another by sending and receiving information much the same way as people pass messages to one another.

A message for an object is a request for execution of a procedure, and there fore will involve execution of a procedure, and there fore will involve a function in the receiving object that generates the desired result. Messages passing involves specifying the name of the object, the name of the function to be sent.

e.g.

Employee salary(name);

Information

Object

Message

### ***Applications of OOP***

- Real time systems
- Simulation and modelling
- Object oriented data body
- Hypertext, hypomedia and expertext
- AI and expert systems
- Neural networks and parallel programming
- Decision support and office automation systems
- CIM / CAM /CAD systems

### ***Constructors and Destructors***

#### ***Constructor***

A constructor is special member function whose task is to initialize the objects of its class. It is special because its name is the same as the class name. The constructor is invoked whenever an object of its associated class is created. It is called constructor because it construct the values of data members of the class.

A constructor that accepts no parameter is called the default constructor. The constructors that can take arguments are called parameterized constructors. The constructors can also be used to allocate memory while creating objects. Allocation of memory to objects at the time of their construction is known as dynamic construction of objects.

### ***Destructor***

A destructor, as the name implies is used to destroy the objects that have been created by a constructor. Like a constructor the destructor is a member function whose name is the same as the class name but is preceded by a tilde. For example the destructor for the class integer can be defined as shown below.

```
~integer() { }
```

A destructor never takes any argument nor does it return any value. It will be invoked implicitly by the compiler upon exit from the program to clean up storage that is no longer accessible.

A program typically consists of a number of objects, which communicate with each other by calling one another's member functions. We should mention that what are called member functions or methods. Also, data items which are referred to as instance variables. Calling an object's member function is referred to as sending a message to the object.

Keep in mind that object oriented Programming is not primarily concerned with the details of program operation. Instead, it deals with the overall organisation of the program. Instead of concentrating on tasks, you concentrate on representing concepts. Instead of taking a top-down approach, you take a bottom-up approach.

## **2.2 Microsoft Foundation Classes**

- MFC Overview
- The Framework
- SDI and MDI
- Documents, Views and the Framework
- CWinApp: The Application class
- Document Templates
- Document Template Creation
- Relationships Among MFC Objects

- Window Objects
- Derived Window Classes
- Registering Window “Classes”
- General Creation Sequence
- Destroying Windows
- Device Contexts
- Graphic Objects

### *MFC Overview*

Taken together, the classes in the Microsoft Foundation Class Library(MFC) make up an application framework, the framework on which you build an application for windows. At a very general level, the framework defines the skeleton of an application and supplies standard user interface implementations that can be placed onto the skeleton. Our job as a programmer is to fill in the rest of the skeleton, those things that are specific to your application.

Version 4.0 the MFC framework supports 32-bit programming for Win32 platforms including windows95, windows NT and later MFC win32 support includes multithreading.

### *The Framework*

Our work with the framework is based largely on a new major classes and several Visual C++ tools. Some of the classes encapsulate a large portion of the Win32 application programming (API). Other classes encapsulates application concepts such as Documents, Views and the application itself. Still other encapsulates OLE features and ODBC data-access functionality.

## ***SDI and MDI***

MFC makes it easy to work with both single document interface (SDI) and multi document interface (MDI) applications.

SDI applications allow only one open document frame window at a time. MDI applications allow multiple document frame windows to be open in the same instance of an application. An MDI application has a window which multiple MDI child windows, which are frame windows themselves, can be opened, each containing a separate document.

## ***Documents, Views and Framework***

At the heart of the framework are the concepts of document and view. A Document is a data object with which the user interacts in an editing section. It is created by the New or Open commands on the file menu and is typically saved in a file. A view is a window object which the user interacts with a document.

The Key Objects in a Running Application are:

### ***1. The Document***

Our document class(derived from CDocument) specifies application's data.

### ***2. The View(s)***

Our view class(derived from Cview) is the user's "Window on the data". The View class specifies how the user sees documents data and interacts with it. In some cases, you may want a document to have multiple views of the data. If we laid need scrolling, derive from CScroll View. If our view has a user interface that is out in a dialog

template resource derive from CFormView. For a form\_based data-access application, such as data-entry program derive from CRecordView.

### ***3 . The Frame Windows***

Views are displayed inside "document frame windows". In an SDI application, the document window is also the "main frame window" for the application. In an MDI application document windows are child windows displayed inside a mainframe window. Our derived mainframe window specifies the styles and other characteristics of the frame windows that contain our views. Derive firm CframeWnd to customise the mainframe window for MDI applications.

### ***4 . The Document Templates***

A document template orchestrates the creation of documents, views and frame windows. A particular document\_template class creates and manages all open documents of one type. Applications that support more than one type of document have multiple document templates. The class Csingle DocTemplate for SDI applications. Class CMultiDocTemplate for MDI applications.

### ***5 . Application Object***

Our application class(derived from CWinApp)controls all of the objects above and specifies application behaviour such as initialisation and cleanup. The applications one and only application object creates and manages the document templates for any document types the application supports.

### ***6 . ThreadObjects***

If our creates separate threads of example, to perform calculations in the background we will use the classes derived from Cwin Thread.



In a running application these objects co-operatively respond to user actions, bound together by commands and other messages. Each document template creates and manages one or more documents. The user views and manipulates a document through a view contained inside a frame window.

### ***CWinApp : The Application Class***

The main application class encapsulates the initialisation, running and termination of an application for windows. An application built on the frame work must have one (and only one) object items for use of a class derived from CWinApp. This object is constructed before the windows are created.

In a framework application, however, you do not write Winmain. It is supplied by the class library and is called when the application starts up. Winmain performs standard services such as registration window classes.

To initialise the application, WinMain calls our application object's Init Application and InitInstance member functions. To run the application's message loop, Winmain calls the Run member function. On termination WinMain calls the application object's Existence member function.

### ***Document Templates***

To manage the complex process of creating documents with their associated views and frame windows, the frame work uses two document templates classes: CSingle Doc Template for SDI applications and CMultiDoc Template For MDI applications. A CSingleDocTemplate can create and store one document at a time. A CMultiDocTemplate keeps a list of many open documents of one type.

Some applications support multiple document types. In such an application, when the user chooses the New command on the File menu, a dialogue box shows a list of possible new document template object.

### ***Document Template Creation***

While creating a new document in response to a new or open command from the file menu, the document also creates a new frame window through which to view the document.

The document-template constructor specifies what types of documents, windows and views the template will be able to create. This is determined by the arguments you pass to the document template\_constructor.

### ***Document/ViewCreation***

Creation of a new document and its associated view and the frame window is a co-operative effort among the application object, a document template, the newly created document and the newly created frame window.

The following table summarises which objects create what

<u>Creator</u>	<u>Creates</u>
Application Object	Document Template
Document Template	Document
Document Template	FrameWindow
FrameWindow	View

### ***Relationships Among MFC Objects***

A document, the frame window used to contain the view, and the view associated with the document. A document keeps a list of the views of that document and a pointer to the document template that created the document. A view keeps a pointer to its document and is a child of its parent framewindow. A document frame window keeps pointed to its current active view. A document template keeps a list of its open document. The application keeps a list of its document templates. Window keeps track of all open windows so it can send message to them.

### ***Window Objects***

A C++ window object is distinct from its corresponding window, but the two are tightly linked. The window object is an object of the C++ Cwnd class that your program creates directly. It comes and goes in response to your program's constructor and destructor calls. The windows window, on the other hand, is an opaque handle to an internal windows data structure that corresponds to a window and consumes system resources when present.

A windows Window is defined by a "window handle" and is created after the Cwnd. The window may be destroyed either by a program call or by user's action. The window handle is stored in the window object's m\_hwnd members variable.

### ***Derived Window Classes***

Although you can create windows directly from Cwnd or derive new window classes from Cwnd, most windows used in a frame work program are instead created from one of the Cwnd-derived frame-classed supplied by MFC.

### ***Cframe Wnd***

Used for SDI frame windows that frame a single document and its view. The frame window is both the main frame window for the application and the frame window for the current document.

### ***CMDI Frame Wnd***

Used for individual documents opened in an MDI main frame window. Each document and its view are framed by an MDI child frame window contained by the MDI main frame window.

### ***CMDIChildWnd***

Used for individual documents opened in an MDI main frame window. Each document and its view are framed by an MDI child frame window contained by the MDI main frame window.

### ***Views***

Views are created by using the Cwnd derived class Cview. A view is attached to a document and acts as an intermediary between the document and the user. A View is a child window that typically fills the client area of an SDI frame window or an MDI child frame window.

### ***Dialog Boxes***

Dialog boxes are created using the Cwnd derived class Cdialog.

### ***Forms***

Form views based on dialog-template resources, like dialog boxes, are created using classes CFormView and CRecordView .

### ***Controls***

Controls such as buttons, listboxes and combo boxes are created using other classes derived from Cwnd.

### ***Control Bars***

Child windows that contain controls. Examples include toolbars and statusBars.

### ***RegisteringWindow “classes”***

In a traditional program for windows, we process all messages to a window in its “window procedure” or “WndProc”. A WndProc is associated with a window by means of a “window class registration” process. The main window is registered in the WinMain function, but other classes of windows can be registered any where in the application.

In contrast, most window class registration activity is done automatically in a framework program. The framework still used traditional” registration classes”. and it provides several standard ones, registered for you in the standard application initialisation function. You can register additional registration classed by calling the AFXRegisterWndClass global function and then pass the registered class to the Create member function of Cwnd.

### ***General Creation Sequence***

All the window classes provided by the Microsoft Foundation Class Library employ two phases Construction. That is during an invocation of the C++ new operator, the Constructor allocates and initialises C++ object but does not create a corresponding Windows window. That is done afterwards by calling the Create member function of the window object.

The create member function makes the windows window and stores its HWND in the C++ objects data member M\_HWND. Create gives complete flexibility over the creation parameters.

### ***Destroying Windows***

In the frame work when the user closes the frame window, the windows default OnClose handler calls Destroy Window. The last member function called when the windows window is destroyed in OnNcDestroy, which does some cleanup, calls the Default member function to perform windows cleanup, and lastly calls the visual member function PostNcDestroy. The CframeWnd implementation of PostNcDestroy deletes C++ windows objects.

### ***Device Contexts***

A device context is a window data structure that contains information about the drawing attributes of a device such as a display or a printer. All drawing calls are layed through a device context object, which encapsulates the windows API's for drawing lines shapes and text. Device context allow device independent drawing in windows. Device context can be used to draw to the screen to the printer or to a metafile.

### ***CPaintDC***

CPaintDC objects encapsulates the common idiom of windows calling the BeginPaint function then drawing in the device context then calling the EndPaint function.

### ***CClientDC***

CClientDC objects encapsulate working with a device context that represents only the client area of the window. The CClientDC constructor calls the GetDC function and the destructor calls the ReleaseDC function.

### ***CWindowDC***

CWindowDC objects encapsulates the device context that represents the whole window including its frame .

### ***CMetafileDC***

CMetafileDC objects encapsulates the drawing into windows metafiles.

### ***Graphic Objects***

Windows provides a variety of drawing tools to use in device context. It provides Pens to draw lines braces to fill interiors and fonts to draw text. The MFC library provides graphic-object classes equivalent to the drawing tools in windows.

## **2.3 About Visual C++**

Visual C++ is an integrated developed environment for Windows using C/C++. Visual C++ development system supports development of DOS applications, Windows applications using C(APIs) and C++(MFC).Newer releases support cross-platform application development for Apple Macintosh. In addition to this, one can build

DLL&OCX control which can be used by developers of both VC++ and other Windows programming environment like PowerBuilder, VisualBasic, Delphi etc.

In VisualC++ IDE, one can program with APIs(C-Approach) or with C++ within the MFC class-library application frame work, which is part of VC++. In both cases one can use many visual tools like graphic resource editor, source browser and debugger to make the development faster, easier and intuitive.

### ***MFC***

MFC is packaged along with VC++. Like any other software MFC also has version numbers. The version of MFC which is packaged with VC++ 4.0 is 4.0. MFC is a collection of over 170 classes. These classes wrap the functionalities of APIs by grouping them context-wise. In addition to being wrapper classes they also define the program structure (hence MFC are also called as Application Frameworks(AFX) in addition to being class-library).

MFC simplifies Windows programming by taking care of basic chores like defining and registering window classes and maintaining message loops. Programmer can concentrate on application specific code and increase productivity. It also makes programming modular by replacing message to these handler functions. Though MFC is vast (over 170 classes), proper grouping of services makes it relatively easy to remember and work with, compared to win32 APIs. Even though MFC is built on existing APIs, this layer is accessible to the programmer. In the absence of any special services from MFC, the programmer can go for basic API services.

### ***WIZARDS***

VisualC++ provides two wizards, AppWizard and ClassWizard to make the development process faster and error free.



AppWizard is a code-generator which creates a working skeleton application based on MFC. This Wizard is capable of providing skeleton applications for DLL and OCX also in addition to regular EXEs. AppWizard doesn't generate all the code for an application. It gives a starter a set of files to get started quickly with a new application. The skeleton application can also provide ODBC and OLE support.

Although AppWizard which is a part of VC++ provides skeletons SDI, MDI and Dialog based applications, one can design AppWizard to build skeleton applications of specific needs. For example, if one needs to develop multiple projects, say telecommunication interface, one can build a special wizard that automates the process.

### ***CLASSWIZARD***

This utility helps in creating new classes, writing message handlers, virtual functions and DDX, DDV support to dialog box controls.

### ***DATABASE ACCESS***

MFC provides classes which help in connecting one's application to various databases for retrieving and updating information. It provides support to both ODBC (open data Base Connectivity) and DAQ (Data Access Objects based on OLE technology) standards for connecting to the databases. This, coupled with fast development cycles wizards, C++ power, runtime checking and debugging tools with VC++ makes it an obvious choice in many complex situations for relatively hassle free development.

A visual C++ application is an application for windows that you design and develop using MFC, the Microsoft visual C++ build tools and the visual C++ windows hosted development environment.

By using a totally integrated environment we can develop our application by focusing on the visual interface elements. Visual C++ calls these elements as "user interface objects". We first design for user interface objects and then use the visual C++ tools to create and manage the code to support them.

These tools automate the often tedious and error-prone process of deriving classes, creating member functions and mapping functions to messages. Using visual C++ we can concentrate on designing the resources for our application and writing the functional code to handle messages.

We can also use the visual C++ tools develop standard applications for windows SDK in C or in C++, since Visual C++ includes a text editor, project window, debugger and resource editors.

### ***APPSTUDIO***

APPSTUDIO is another utility available in VC++. It provides facility to include the windows resources to our application . The available resources in VC++ are

- Accelerator
- String table
- Dialog box
- Menu
- Bitmap
- Icon

### ***Key Accelerator***

This is one resource which makes the firing or executing certain functions in our application with one or two key strokes.

eg.,

When a function for display of image is available, this function when provided with key Accelerator of Alt + D and added in resources whenever this combination is pressed the function is executed.

### ***String Table***

The string table contains the identification numbers of each resource which is added in our application.

This also have the names correspondingly, which can be changed according to our wish.

### ***Dialog Box***

This resource is very important for an application to get input from the user in a very simple manner. Dialog box provides with very simple GUI tools like edit box, combobox, listbox & radio buttons etc.. to make the input as easy as possible.

### ***Dialog***

Open a new blank dialog box and add the necessary controls for the application using classwizard to create a new class for the dialog box and now add variables for each control box and add any function for controls provided.

When the application is rebuilt and when dialog class added as header wherever we want to call then the dialog box can be activated.

### ***Menu***

The another facility provided by APPSTUDIO is menu. The menu already created by the system is called IDR\_MANIFRAME. The user if necessary can added any item to the same menu or can create a new menu for our application and load the menu wherever necessary.

*Image Compression*

---

### 3.1 Introduction

Remote Sensing satellites take image of the earth resources using high spatial resolution multispectral CCD (Charge coupled Device) cameras. These cameras give out enormous data, as they represent the image in digital format. And thus, high data rates are required for transmitting these satellite images to the ground station in real time. It also results in larger money requirement for storing the images onboard the Satellite. But, by efficiently coding these image using data compression technique, it is possible to bring down the transmission channel bandwidth requirement and storage space requirement. This compression can be achieved either losing some information in the image or without losing any information in the image.

The data compression techniques can be classified as Lossy or Lossless based on the way they achieve compression. In the case of lossless coding, compression does not result in any loss of information in the signal. This is unlike lossy coding techniques that lose certain amount of information for achieving better compression ratio. These lossy coding can be used in applications where loss of information is tolerable and high compression is required. A Lossless coding technique is applied where the information content of the signal has to be retained. These techniques exploit only the redundancy in the signal for compression. This redundancy is characteristic which is related to predictability, randomness, smoothness in the data and reflects the information content. A quantitative measure of this information content is entropy. It is the minimum average number of bits required for representing a sample from a signal source.

Image compression methods can be classified as

1. Lossless coding
2. Lossy coding

based on the way they achieve compression.

In the case of lossless coding, compression is achieved by removing the 'redundancy' inherently present in the image. This redundancy is a characteristic which is

related to predictability, randomness, smoothness in the data. An image histogram, which plots the number of pixels against the pixel intensity, gives a statistical measure of this randomness in the image. For example, an image which has a flat histogram has a little redundancy and thus a little chance for compression. Satellite images are having their pixel values concentrated around a few values which means it is possible to compress these images. In lossless coding [1], the information content of the image is retained. A quantitative measure of this information content is entropy. It is the minimum average number of bits required for representing a sample from a signal source. For example, 8-bit PCM represents an image and thus possible to code with a lesser number of bits.

In general, lossless coding techniques such as Huffman, Arithmetic coding try to approach the entropy limit and hence called entropy coding techniques. The basic principle behind these entropy coding techniques is to allocate a fewer number of bytes for often occurring symbols and more number for symbols which occur very rarely. This generates a variable length code which requires a rate buffer for matching channel data rate. These are also called reversible coding methods because no information is lost during the coding process. It is possible to get back the original image by decoding an entropy coded image. But these methods can only give at the most a compression ratio of 1:2 for remote sensing images with reasonable implementation of complexity.

Onboard the Satellite, CCD camera gives out the data in PCM format which is a time discrete format, amplitude discrete, representation of the image which removing much statistical redundancy. This image can be considered as a two dimensional spatial array of picture (pixel) elements or discrete values. These pixels represent the intensity values of the image at the indexed spatial position. For instance in a gray scale spectral band image using 8-bit code for each pixel, the intensity value can range from 0 - 255. Such fixed length code is all right when space or transmission time is not an issue. This scheme is also efficient in a case where all the intensity values (symbols) ranging from 0 - 255 are equally likely to occur in the image. But, in practice the

frequency of the occurrence of the intensity values in the image may vary. Thus it may be redundant representation to allocate a fixed number of bits for all possible pixel values.

For efficient coding of satellite images, with less number of bits, the code bits allocated should conform to the frequency of occurrence of the pixel intensity values. An image histogram that plots number of pixels against the pixel intensity gives the idea of how efficiently it can be coded. For example an image which has a flat histogram has little redundancy and thus little chance for compression. In general satellite images are having their pixel values concentrated around a few values which means it is possible to compress these images. The idea behind this efficient coding is to allocate more number of bits for the symbol occurring rarely and less number of bits for the symbols occurring often.

### 3.2 HAUFFMAN CODING

Hauffman code takes advantage of the statistical redundancy present in the source. Instead of uniformly assigning the bits to all the symbols as the case in standard PCM (Pulse Code Modulation), here the bit allocation depends on the probability of the occurring symbols. Less bits are allotted to the symbols occurring more frequently and more bits are allocated for the bits occurring less frequently, thereby reducing the number of bits transmitted for a given message. Hauffman has devised an algorithm for generating variable length code words for a given source symbol probabilities. This algorithm is optimum in the sense that average number of binary digits required to represent the source symbols in a minimum. It also forms codewords that satisfy the prefix condition, i.e., no codeword is a prefix to another codeword. This condition ensures unique decodability of the received sequence.

#### *Description Of The Algorithm*

Hauffman algorithm uses a frequency table and data structure called binary tree for constructing a variable length code. Here, this binary tree is referred to as code

tree. In a code tree each node consists of two child nodes or none. The nodes with no child nodes are called leaves of the tree and they represent the symbols. These nodes are linked to one another based on the probabilities, i.e., frequencies of the symbols. The hauffman tree is so formed that the bits allotted for the most probable symbol is the least and is maximum for the least probable symbol. This results in shorter code for the message or the image to be coded. The procedure to construct the hauffman code tree is described below.

The first step is to calculate the frequencies of the given image. For example in an image represented by 8 bit PCM code the symbol set consists of intensity values ranging from 0 - 255. So the frequency table is formed by counting, how many times each intensity value is occurring in the image. These form the leaves of the code tree. It is convenient to arrange the symbols in ascending or descending order. The next step is to combine the two symbols with lowest frequencies to form a single node. This node is a parent node for these two symbols. A '0' is assigned to the left child and a '1' is assigned to the right child to differentiate these child nodes. Now this parent node will have the sum of the frequencies of the child nodes and the symbol set is less by one. The above step is carried out repeatedly on the resultant parent nodes and the symbols, while allocating one bit to the child nodes formed at every step, until all the symbols are reduced to a single super symbol. This is the root of the code tree which represents the message itself and will have a frequency value that is the sum of the frequencies of all the symbols in the given image.

To form the codeword for a given symbol, the tree is traversed from the root to the leaf having that symbol while reading off the bits assigned to the branches in the path. Since the symbol only occur at the leaves, no other symbol is encountered along the path from the root to the leaf. This ensures that no code word will have another symbol code word as its prefix. And also while constructing the tree, the lowest frequency nodes are combined to form a parent node by adding one level of branching. This makes the path length of the low frequency symbols longer and hence longer code word length for low probability symbols.



### 3.3 Arithmetic Coding

In Hauffman coding, each symbol of the message gets translated into a codeword with an integer number of bits. In some cases, based on entropy, a symbol may only need a fraction of a bit to represent it. But there is no provision for this in Hauffman code as bits are allocated per symbol basis. Hence, in such cases Hauffman coding still allocates one whole bit. This makes the code inefficient for messages with fractional entropy. Arithmetic coding does not have this drawback because the bits are not allocated per symbol basis. The whole message gets mapped to an interval in the range  $[0-1]$ .

Then the message can be represented by any single real number in this interval. The binary representation of this number is the compressed code for the whole message. This single number can be unambiguously decoded to create the exact stream of symbols in the message that were used to generate that number. In such code it is possible to effectively code each symbol of the message with a fractional bits on average. It is generally referred to as interval coding because it uses the concept of translating the message to an interval. The procedure to code a message using Arithmetic coding is described below.

#### *Description Of The Algorithm*

To start with, the main interval  $[0,1]$  is divided into subintervals according to the number of symbols in the message to be transmitted. The subinterval lengths are proportional to the probability of their occurrence in the message. But it does not matter to which portion of the main interval they are mapped.

While encoding the message, symbols are processed in the order they come in. The first symbol will be encoded by mapping it to the corresponding symbol subinterval in the main interval. This subinterval will have a lower bound and an upper bound and its range is proportional to the probability of that symbol. It means the interval length is

similar for symbols with less probability. This effectively results in allocation of more number of bits for representing less probable symbols. This is in accordance with the basic idea of efficient coding. Now, for the next symbol, this subinterval becomes the main interval. It is further divided by mapping again all the symbols onto this interval. This interval is used to encode the next symbol of the message. As the symbol get encoded, the interval narrows down. This process continues until the end of the message symbol occurs. Finally, the message gets mapped to an unique interval. Once the message is encoded in this way, any real number that lies in this interval can represent the message without any ambiguity. It can be observed that the final interval range depends on the message length and the probabilities of the symbols. The number of times the interval gets divided depends on the length of the message. A longer message results in a smaller interval which needs more number of bits for its representation. The individual symbols decide the quantum of reduction in the interval length after each symbol encoding.

The decoding process is just the reverse of the encoding process. First it is seen that in which symbols range does the incoming encoding number falls. By knowing the range the first symbol can be decoded. Then the effect of this symbol is removed by the reverse process i.e. subtracting the lower interval limit value of the symbol from the floating point number than dividing the number by the symbols range. And from this new interval limit value of the symbol from the floating point number than dividing the number by the symbols range. And from this new interval number, the next symbol can be decoded. The procedure can be repeated to extract the rest of the symbols. To indicate to the decoder that the end of the message has been reached, and end-of-message symbol is usually added to the message alphabet.

### **3.4 DCPM - Huffman / Arithmetic Coding**

The basic limitation of any lossless coding is the requirement of more bits or inability to compress to proper attractive levels. The best among the lossless coding techniques i.e., Arithmetic and Huffman coding by itself is usually not used. Hybrid

coding is a promising one where in two or more encoders are employed receiving end in the reverse order. Thus DCPM followed by Arithmetic encoder or Hauffman encoder should give better results. This is because DCPM memorises the previous symbols and thus effectively reduced the entropy by representing the source with a difference signal . Since the implemented DCPM is not having any Quantizer, the whole setup is a lossless one .

### ***Concept of predictive coding***

In DCPM system, the incoming sample is predicted based upon previously transmitted and decoded information. since, there is no quantization involved in the present system the previous samples can be used for prediction. Thus the samples are represented as the difference values between the predicted and the present value of the sample. These difference values are coded as binary words of variable word length and sent to the channel coder for transmission. Since the system uses a predictor which removes the redundancy present in the source .

### **3.5 Transform Coding**

Transform coding, also called block quantization, is an alternative to predictive coding. A block of data is unitarily transformed so that a large fraction of its total energy is packed in relatively a small number of transform coefficients which are quantized independently. The optimum quantizer is the one which minimizes the mean-square-error. The ideal transform is Karhuner Love (KL) transform. The other transforms are basically Fourier-Transform based. Examples are Sine, Cosine, Hadamard etc.

The cosine transform is widely known and chips are also available. The DCT belongs to a class of mathematical operations that includes the well-known Fast Fourier Transform (FFT), as well as many others. The basic operation performed by these tranforms is to take a signal and transform it from one type of representation to another.

The DCT is closely related to the Fourier Transform, and produces a similar result. It takes a set of points from spatial domain and transform them to an identical representation in frequency domain. In processing, the signal is a graphical image. The X and Y axes are two dimensions of the screen. The amplitude of the "signal" in this case is simply the value of a pixel at a particular point on the screen. The gray value is in general of 8 bits value. So a graphical image display on the screen can be thought of as a complex three-dimensional signal, with the value on the Z axis denoted by the color on the screen at a given point. This is the spatial representation of the signal. The DCT can be used to convert spatial information into "frequency" or "spectral" information, with the X and Y axes representing frequencies of the signal in two different dimensions. And like the FFT, there is an Inverse DCT (IDCT) function that can convert the spectral representation of the signal back to a spatial one.

The formula for DCT is given below

$$DCT(i,j) = (1/2N) * \sum C(i) * C(j) * pixel(x,y) * \cos[(2x+1) * i * \pi / 2 * N] * \cos[(2y+1) * j * \pi / 2 * N]$$

$$C(x) = 1/2 \text{ if } x \text{ is } 0, \text{ else if } x > 0$$

$$Pixel(x,y) = (1/2N) * \sum C(i) * C(j) * DCT(i,j) * \cos[(2x+1) * i * \pi / 2 * N] * \cos[(2y+1) * j * \pi / 2 * N]$$

$$C(x) = 1/2 \text{ if } x \text{ is } 0, \text{ else if } x > 0$$

One of the first observations shows up when examining the DCT algorithm is that the calculation time required for each element in the DCT is heavily dependent on the size of the matrix. Since a doubly nested loop is used, the number of calculations is  $O(N^2)$ . As  $N$  goes up, the amount of time required to process each element in the DCT output array will go up dramatically. One of the consequences of this is that it is virtually impossible to perform a DCT on an entire image. To get around this, DCT implementations typically break the image down into smaller, more manageable blocks. The JPEG group selected an 8-by-8 block for the size of their DCT calculation. While

increasing the size of the DCT block would probably give better compression, it doesn't take long to reach a point of diminishing returns. Research shows that the connections between pixels tend to diminish quickly, such that pixels even fifteen or twenty positions away are of very little use as predictors. This means that a DCT block of 64-by-64 might not compress much better than if we broke it down into four 16-by-16 blocks. And to make matters worse, the computation time would be much longer. While there is probably a good argument for using 16-by-16 blocks as the basis for DCT computations, the JPEG committee elected to stick with 8-by-8.

*System Design*

---

## 4. SYSTEM DESIGN

The package is developed to bring all the mostly used algorithms into a single package. The package is developed based on object oriented approach. Classes are created for every algorithm that is implemented and dynamically loaded when the user called particular algorithm.

The images are available in the binary format and any image can be loaded to compress on any algorithm, when the compression is completed the resulting image is saved in the name given by the user.

The system also provides certain image processing facilities like display the image, zoom the image, editing operations on the image to study about the image characteristics. The system also provides tool like, Histograms; Statistical Methods to study about the pixel intensities, picture features of the image.

The various classes, its member functions and the major modules that are designed are described below

The classes

- Chauff
- Cdct
- Carithmetic
- Cdcpm

### ***Chauff***

The Chauff class is created to implement the hauffman coding algorithm.

The private variables that are declared in this class are

- Infile
- Outfile
- In, out (*Refer Figure 2.1*)

### ***Infile***

The infile is a file pointer which is opened in binary mode for input to the algorithm when the algorithm is called for compression and is used for output. When the algorithm is called for decompression.

### ***Outfile***

The outfile is a bitfile pointer which is opened as input file when the algorithm is called for decompression and as output file when it is called for compression.

### ***In,out***

The in, out are the Cstring variable names that are used to store the input, output file names respectively for compression and decompression.

### ***Member Functions***

The major modules which are used as public member function are described below.

1. Count Bytes
2. Scale Counts
3. Output Counts
4. Convert Tree to Code

### ***Count Bytes***

This member function is to read the size of the image. The image size is of a rectangular matrix and is in the multiples of 16. The image frequency table is also formed here i.e. the no of pixel in each intensity is taken and stored in an array.



### ***BuildTree***

This function is generated to develop a binary tree to connect the frequency table. The intensities are arranged in descending order and the leaf nodes are the symbols that are added to the binary tree.

### ***Convert Tree To Code***

This member function traverses the binary tree and assigns code to each leaf.

### ***CompressData***

This function is to map the corresponding pixel intensity to the code generated by the tree to code function and are written to the output file name.

### ***Expand Data***

This function is called to expand the compressed code to convert into original image. This is called when an image is to be decompressed.

### ***Compress & Expand File***

These two are the main member functions which are main functions for compression and decompression. They in turn call the corresponding related functions to achieve the task.

### ***DCT***

The private variables that are declared in this class are

- Input
- Output
- In, out (*Refer Figure 2.2*)

### *Input*

The infile is a file pointer which is opened in binary mode for input to the algorithm when the algorithm is called for compression and is used for output. When the algorithm is called for decompression.

### *Output*

The outfile is a bitfile pointer which is opened as input file when the algorithm is called for decompression and as output file when it is called for compression.

### *Zigzag*

This is a structured array which holds the row , column of every pixel that is being selected and can hold a size upto 1024 by 1024 matrix.

The Public member functions that are used for this class is explained below

### *Initialise*

This member function initialises the any element of the matrix value by the Cosine values that has to be multiplied with original image pixels depending upon the quantity given as input.

The Cosine value is calculated by

$$C(i , j) = \text{sqrt}(2/ n) * \text{Cos}(\text{pi} * (2* j+1) * I( 2* N));$$

### *Forward Dct*

This function is to multiply the coefficient matrix with the cosine matrix to get the cosine transformation matrix.

### ***Inverse Dct***

This Function is to convert the compressed matrix from the compressed file and then do the inverse of the Dct process to get the original file.

### ***ReadDct***

This function is to called while decomposition to read the Dct matrix from the five unit Dct data. This function is called while compression will the transform matrix into the file.

### ***Major Supporting Modules***

Apart from the Image Compression techniques the other modules which has to be designed are the processing of the images like display, Zoom, Histogram, Statistical analysis. Separate modules are developed and linked.

### ***Display***

This module is developed for the display of the images on the screen. This helps to know the algorithm's efficiency, after compression and decompression.

### ***Zoom***

This function is to Zoom the image to required size to know the nature of the images. The Zoom out procedure is carried out by repetition of pixel depending upon the image size and required Zoom size. Zoom in procedure is carried out by applying some techniques. These are as follows

- Mean Method
- Maximum Pixel Method
- Least Square method

### ***Histogram***

This function is to generate the image histogram, for the study of repetition of pixels. The histogram is done by taking pixel intensity by no pixels.

### ***Statistical Analysis***

This Module is developed for the study of the image characteristics. This module is used to calculate the maximum/minimum intensity pixel in a particular image mean, standard deviation of the images.

### ***Manipulation***

This module is designed to cut, paste, copy operations on the image. This operations are carried out by using clipboard as the buffer for the temporary storing.

*System Flow*

---

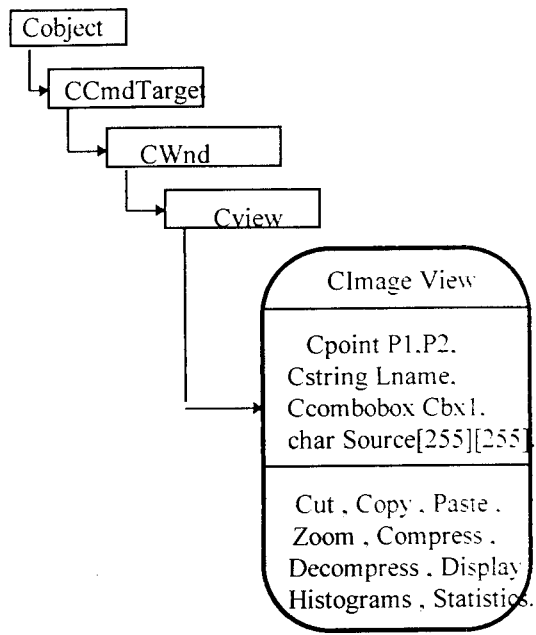


Fig 1.1

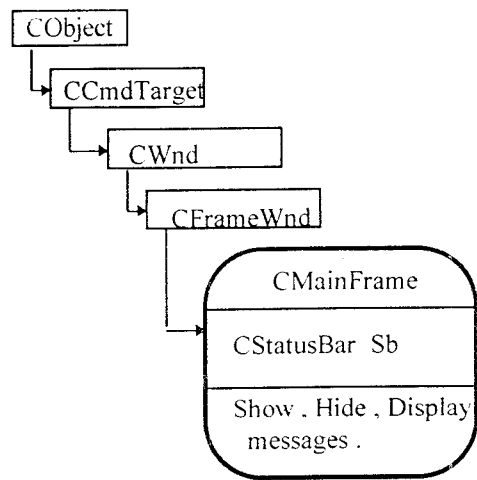


Fig 1.2

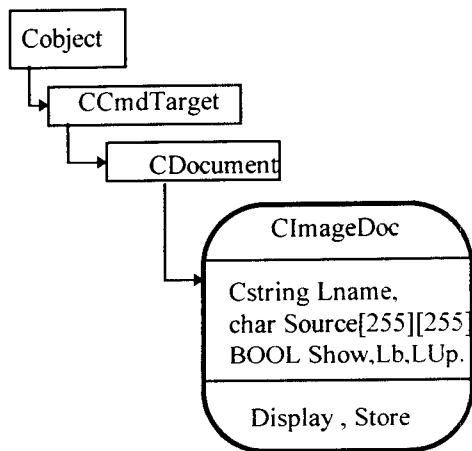


Fig 1.3

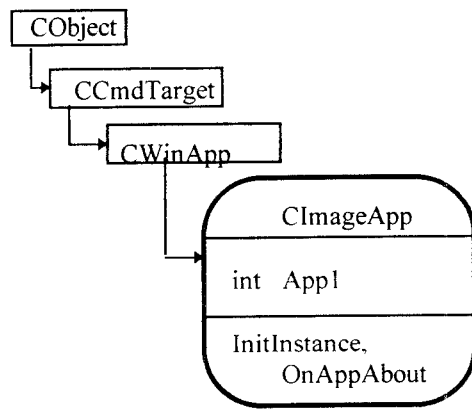


Fig 1.4

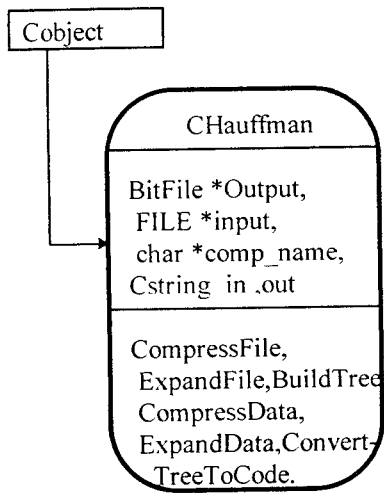


Fig 2.1

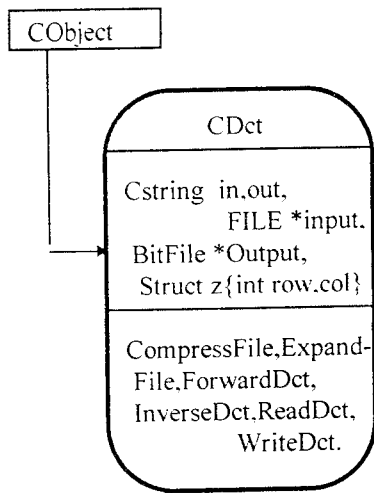


Fig 2.2

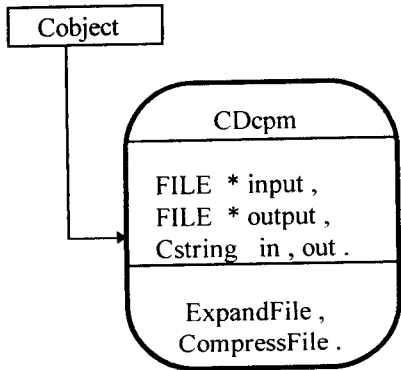


Fig 2.3

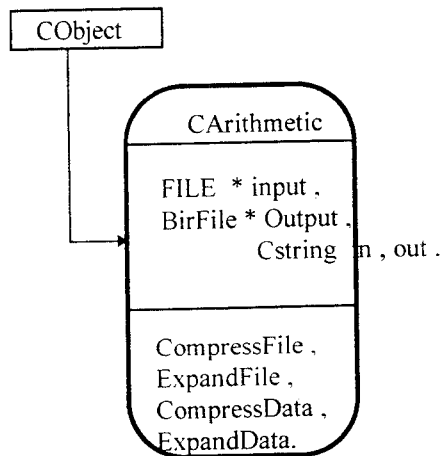
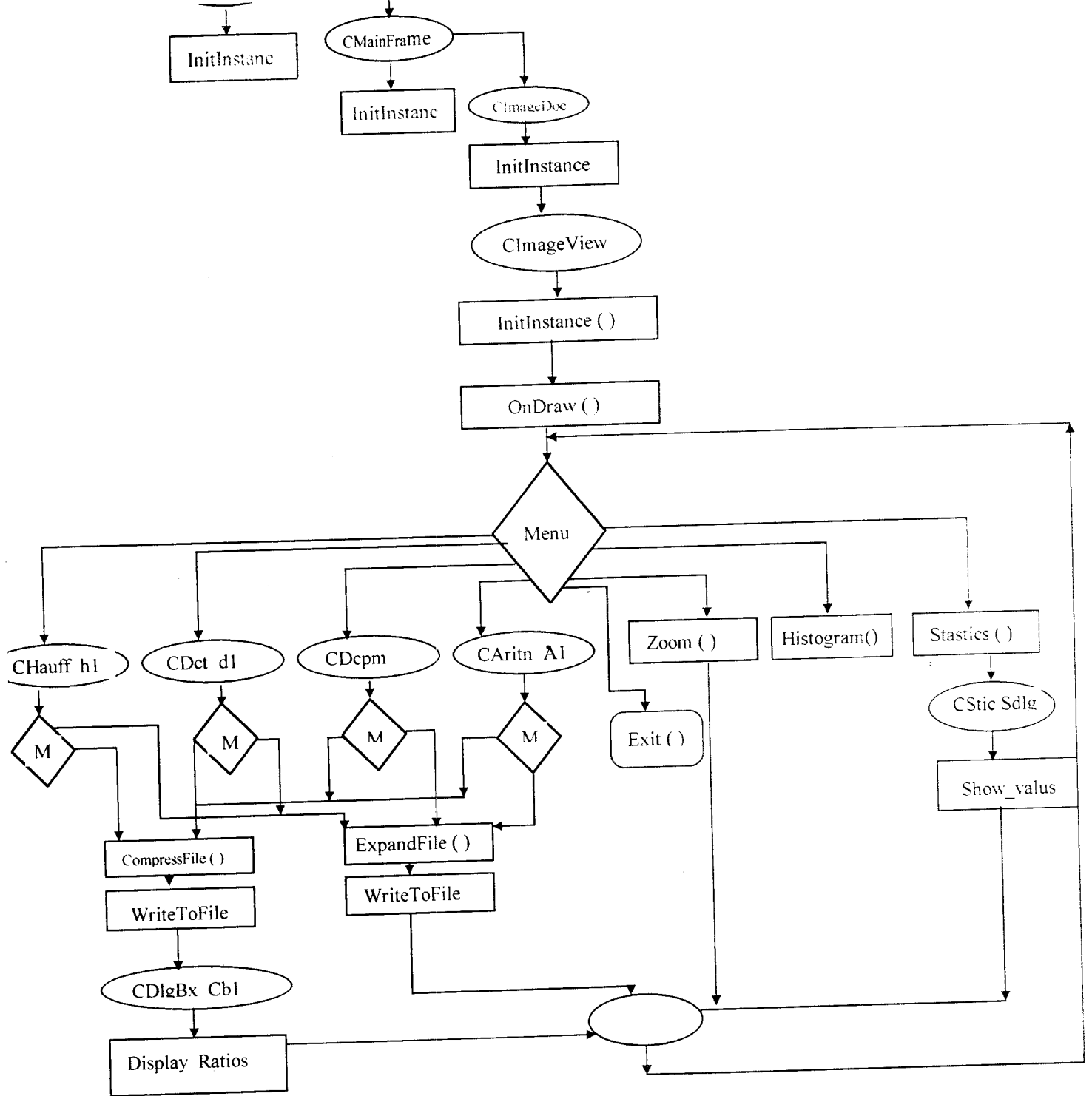


Fig 2.4



System Flow Chart

Fig 3.1



*Implementation*

---

## 6 . Implementation

### 6 . 1 Hardware Support

- ◆ Intel Pentium@100 MHz
- ◆ 16 MB RAM
- ◆ 540 MB HDD Mirrored
- ◆ One 1.44 MB 3.5" FDD
- ◆ Running DOS 6.22 and Windows95

### 6.2 Software Support

- ◆ Visual C ++
- ◆ Ver 4.0

### 6 . 3 Implementation of Algorithms:

#### *In VC++*

The package is developed in visual C++ to provide the GUI facilities and to use the available MFC libraries.

The algorithms which are developed are taken and developed as separate class. These class are derived from MFC base class CObject to get the windows programming rights to over own member functions. It is menu driven package the user can select any item from the menu and do necessary operations. The over all flow diagram is shown in the figure 3.1.

The main features that are included in the package are:

- Compression
- Manipulation
- Image Stastics
- Zooming

- Loading & display of images
- Online Help

### ***Compression***

The menu options are provided for the user to select any particular algorithm. When selected then an instance of that class is created and a dialog box is opened to load the image to be compressed and also gets the output name of the file is got. Then a radio button is provided to select the Compression or DeCompression. Then if the user selects any one of them and clicks the ok button, then corresponding task will be evaluated and it shows the what ratio is compressed for particular image.

### ***Manipulation***

It is the one more object of the menu, it contains the options of cut, paste, copy operations. Whenever cut option is selected from the menu, the selected part of the image in the screen will be deleted, then that cutted part will be converted in to a bitmap and will be storing in the buffer. Whenever the paste option is selected from the menu, the user has to point out where it has to paste, after pointing the point the cut image in the buffer will be pasted at the point.

### ***Tools***

This tool menu consists of two options. These are as follows

- Histogram
- Statistics

### ***Histograms***

This function draws the Graph according to the no.of pixels by pixel intensity of images. This Graphs are used for checking the redundancy of pixels of image. If the graph spreaded wide then the redundancy of pixel in the image is less, otherwise the redundancy of pixels in the image is high. By showing graph itself, whether the image having the redundancy of pixels or not.

we developed the histogram in VC++ using the different pen widths of the histogram is drawn by taking one bar of width 3 is one pixel intensity along x axis and each pixel in y axis represents 60 pixels.

### ***Statistics***

This tool is used to calculate the mean, standard deviation, maximum pixel and minimum pixel. It will calculate the mean, standard deviation, maximum and minimum pixels of the image.

### ***Loading of an image***

The images stored in a particular directory depending up on the type of images like (Lansat, Spot, Geologica, etc..). Whenever the user wants to load a image, we should load the image by selecting image from the menu, then combo box will appear. In that combo box we can select the necessary image by double clicking on that name of the image. After double clicking the file, the corresponding image is read and copied to an array in the application.

### ***Display***

This option is to display the image that has been already loaded.

### ***Zoom***

This is another image processing facility provided in the package. The Zooming can be of two types. These are compressing and enlarging. For Zoomin (compression), two methods are provided as options.

- Maximum Pixel Method
- Average Method
- Least Square Method

These methods are available in a dialog box radio button. When maximum pixel method is selected, then the required region is taken for resulting image. The required region is calculated by original size/required size in square matrix. The zoom out procedure is carried out by repetition of pixels.

### *Help*

The help manual is provided in the application, which is created in MS Word and compiled in MS Explorer.

The help provides topics to help the users in using the package, tips about the image compression etc. The help can be activated by pressing F1 key wherever help is requested through help menu selection.

### *In C++*

A Similar system is developed in IBM machine in C++ to implement all the algorithms. The classes which were developed are loaded and a main program with menu selection is provided to run any particular algorithm.

*Conclusion*

---

## 6 . Conclusion

### 6 . 1 Scope For Further Development:

The system is now capable of applying now only four algorithms. When the remaining algorithms, if converted and added as a class then they can be easily implemented.

The image processing facilities can be still improved by adding still more routines for filtering edge cutting, the image clarity can be improved.

The statistical analysis can be elaborated by adding some more routines to have a complete image compression package.

When classes were called using DLL's, the memory efficient can be still improved and the execution time can be reduced.

In C++ system, the user interface is not provided, when programs were written in X/Motif for providing GUI facilities and the classes when called in background, then a complete package can be done in the UNIX environment also.

*Bibliography*

---



**Inside Visual C++**

- David J. Kruglinski

**Heavy Metal Visual C++ Programming**

- Steve Holzner

**Digital Image Processing**

- Gonzalez & Woods

**Fundamentals of Digital Image Processing**

- Jain

*Screen Design*

---



LanSatllite Image : lena.bin of 256X256 Matrix

OSG Lab Image Compression

File Manipulation Image Type Algorithm Display Compression Tools Help

- Agricultural
- Geological
- Lan Sat
- Spot

- lena.bin
- ez.bin
- lena.bin
- ar.bin
- girl.bin

Image Not Loaded Yet

DSG Lab: Image Compression

\_ | 5 | X

File Manipulation Image Type Algorithm Display Compression Tools Help



Lena Satellite Image : lena.bin of 256X256 Matrix

HUFFMAN CODING



Input File Name

lena.bin

Output File Name

ez.bin

To

Compress

Expand

Cancel

OK

File Name Size X

File Name Size

Input

Output

Compression Ratio