CONVERTING A HAND DRAWN DIAGRAM TO A DIA DIAGRAM

Mini Project Report

Submitted by

GEORGE MATHEW

ARCHANA NAIR

LEAH ABRAHAM NALONNIL

In partial fulfilment of the requirements for award of the degree of ***Bachelor of Technology*** *in Computer Science & Engineering*

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT) TM

Angamaly-683577, Ernakulam

Affiliated to

MAHATMA GANDHI UNIVERSITY

Kottayam-686560

May 2010

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)TM

Mookkannoor (P.O), Angamaly-683577

CERTIFICATE

This is to certify that the mini project report titled **CONVERTING HAND DRAWN DIAGRAM TO A DIA DIAGRAM** submitted by Archana Nair, George Mathew and Leah Abraham Nalonnil, towards partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in Computer Science & Engineering is a record of bonafide work carried out by them during the academic year 2009-2010.

Staff Incharge   Head of the Department

Place:

Date:

Internal Examiner   External Examiner

ACKNOWLEDGEMENT

If words are considered as symbols of approval and tokens of acknowledgement, then let the words play the heralding role of expressing our gratitude. First and foremost, we praise the God almighty for the grace he showered on us during our studies as well as our day to day activities.

We are extremely thankful to the Chairman, **Mr. PV Mathew**, who provided us with all the vital facilities required for the successful completion of the project.

We also thank our Principal, **Dr. K Sunderessan,** for the amenities he provided, which helped us in the fulfilment of our project. We would like to express our sincere thanks to our beloved **Prof. Prasad JC (H.O.D)** who always guided us and rendered his help in all the phases of our project.

We would like to pay our gratitude to our project in charge, **Mr. Mahesh C**, for his constant support and encouragement and for guiding us with patience and enthusiasm in all the stages. We are extremely thankful to **Mr. Jyotish K. John**, **Mrs. Anjali G Nair** who always extended his help and support from the beginning for the successful completion of our project.

We would like to express our sincere gratitude to all the faculties of the Computer Science Department, FISAT. We would like to express special thanks to our Lab instructors for providing us with all the necessary Lab facilities and helping us throughout this project to make it a success.

We also sincerely thank our parents and friends for giving us moral support and encouragement in all possible ways.

ABSTRACT

The project aims at converting a rough hand drawn diagram to a neat diagram. This is done using a free open source software tool, dia. The figures of a flow chart are detected using OpenCV library. OpenCV is an open source computer vision library. The library is written in C and C++ and runs under Linux, Windows and Mac OS X. According to the dimensions found, the XML code for the corresponding diagrams is updated.

CONTENTS

1. PREAMBLE

1.1 Introduction to the Project

OpenCV is a computer vision library. It is a collection of C functions and few C++ classes that implement some popular algorithms of Image Processing and Computer Vision.

OpenCV library supports basic image treatment such as brightness, contrast, threshold of the image. It also supports object detection with which the figures in the flow chart are detected.

Once the figures are detected they are interfaced with the dia. Dia is an open source software that can be used to draw many different kinds of diagrams. It currently has special objects to help draw entity relationship diagrams, flowcharts, and many other diagrams. It is also possible to add support for new shapes by writing simple XML files. It can load and save diagrams to a custom XML format, can export diagrams to a number of formats, including EPS, SVG, XFIG, WMF and PNG, and can print diagrams.

XML (eXtensible Markup Language) is a World Wide Web Consortium (W3C) endorsed standard for semi-structured data. XML data are surrounded by textual markup that serves to describe the semantics of the data. XML allows the author to define his own tags and his own document structure.

1.2 Objectives of the Project

The objective of the project is to convert hand drawn figures to a neat sketch.

The straight lines in a figure are detected using OpenCV functions. This is accomplished by first passing the figure through a function called canny, which detects all the edges. The output of the canny function is applied with hough transform to detect the straight lines.

The detected straight lines are adjusted a bit for ease of application. Then the intersection points are found out using simple line equations. From this the triangles and four sided figures are found out.

Any detected four or three sided figures have to be adjusted for dia. Any detected triangle has to be adjusted to the nearest possible inverted or vertical isosceles triangle. The four sided figures has to be adjusted to the nearest rhombus or rectangle or parallelogram

The detected and adjusted figures are finally drawn in dia to get the neat dia sketch.

1.3 Scope of the Project

The project could be helpful in the education field. Presently there exists no effective system to directly transform a diagram in a book to digital format. This could be easily done using our project. So digitalising book becomes easy

The project gives an additional input option. A simple paper and pen could be used to draw diagrams and input into the system

2. SYSTEM STUDY

2.1 Proof of Concept

Proof of concept is a short and/or incomplete realization of a certain method or idea to demonstrate its feasibility, or a demonstration in principle whose purpose is to verify that some concept or theory is probably capable of being useful.

In software development, proof of concept (abbreviated PoC) is often incorrectly used to describe three processes with different objectives and different participant roles. These uses of the phrase proof of concept are therefore not synonymous and are delineated below.

A proof of concept can refer to a partial solution that involves a relatively small number of users acting in business roles to establish whether the system satisfies some aspect of the requirements.

The PoC development gives the opportunity to touch, feel and have a closer look at the feasibility of innovative idea instead of using long product design sessions, extensive documenting and reviews.

2.1.1 Need of proof of concept

By contrast, the objective of a proof of technology is to determine the solution to some technical problem, such as how two systems might be integrated or that a certain throughput can be achieved with a given configuration. No business users need be involved in a proof of technology.

A pilot project refers to an initial roll out of a system into production, targeting a limited scope of the intended final solution. The scope may be limited by the number of users who can access the system, the business processes affected, the business partners involved, or other restrictions as appropriate to the domain. The purpose of a pilot project is to test, often in a production environment, whether the system is working as it was designed while limiting business exposure.

The key benefits in PoC are:

- Proof feasibility and potential architecture of your idea before investing in full-scale development.
- Demo specific functionality with future solution.
- Validate solution concept.
- Reduction of amount of uncertainty involved with the new project.
- Identification of functional unforeseen problems, risks, gaps and defects prior full-scale development efforts.
- Gain a solid risk-free base for main production stage.

2.1.2 **Proof of concept design**

All the straight lines in the input figure were identified and their coordinates were obtained. . The successful implementation of these actions would prove the feasibility of the project.

2.1.3 **Result of proof of concept**

A screenshot of the result of proof of concept is given in section 7.4.1.

2.2 System Specification

2.2.1 Software specification for development, implementation

A system with the following configuration is needed for implementing open GL Graphics in live video:

Language used	: C++, XML language
Operating system	: Linux OS

Compiler	: GNU C++ compiler
Editor	: Emacs and Gedit text editor
Application Program	: Open CV and dia

2.2.1.1 About C++ Programming Language

C++ is one of the most popular programming languages and there are very few computer architectures for which a C++ compiler does not exist. It is an object oriented programming language.

The initial development of C occurred at AT&T Bell Labs in the early 1980s by Bjarne Stroutstrup. The name C++ was coined by Rick Mascitti where '++' is the C increment operator. The major reason behind the success and popularity of C++ is that it supports the object oriented technology, the latest in the software development and the most near to the real world.

2.2.1.2 About XML

XML, the eXtensible Markup Language, is a current key topic in nearly all aspects of information technology. It is a way of describing semi-structured data besides the data itself, its semantics are encoded as well.

The idea behind the development of XML through the World Wide Web consortium (W3C) was to create a universal way of exchanging data across system and vendor boundaries. So far this design goal seems to be achievable as more and more applications are developed using XML as format for data exchange

2.2.1.3 About Linux Operating System

Linux is a generic term referring to Unix-like computer operating systems based on the Linux kernel. Their development is one of the most prominent examples of free and open source software collaboration; typically all the underlying source code can be used, freely modified, and redistributed, both commercially and non-commercially, by anyone under licenses such as the GNU General Public License.

Linux can be installed on a wide variety of computer hardware, ranging from embedded devices such as mobile phones, smart phones and wristwatches to mainframes and supercomputers. Linux is predominantly known for its use in servers;

in 2007 Linux's overall share of the server market was estimated at 12.7%, while a 2008 estimate suggested that 60% of all web servers ran Linux.

The name "Linux" comes from the Linux kernel, originally written in 1991 by Linus Torvalds. The man supporting Userland in the form of system tools and libraries from the GNU Project (announced in 1983 by Richard Stallman) is the basis for the Free Software Foundation's preferred name *GNU/Linux.*

The primary difference between Linux and many other popular contemporary operating systems is that the Linux kernel and other components are free and open source software. Linux is not the only such operating system, although it is by far the most widely used. Some free and open source software licenses are based on the principle of copy left, a kind of reciprocity: any work derived from a copy left piece of software must also be copy left it. The most common free software license, the GNU GPL, is a form of copy left, and is used for the Linux kernel and many of the components from the GNU project.

Free software projects, although developed in a collaborative fashion, are often produced independently of each other. The fact that the software licenses explicitly permit redistribution, however, provides a basis for larger scale projects that collect the software produced by stand-alone projects and make it available all at once in the form of a Linux distribution.

A Linux distribution, commonly called a "distro", is a project that manages a remote collection of system software and application software packages available for download and installation through a network connection. This allows the user to adapt the operating system to his/her specific needs. Distributions are maintained by individuals, loose-knit teams, volunteer organizations, and commercial entities. A distribution is responsible for the default configuration of the installed Linux kernel, general system security, and more generally integration of the different software packages into a coherent whole.

2.2.1.4 About GNU compiler

The GNU compiler collection includes front ends for C, C++, Objective-C as well as libraries for these languages. GCC was originally written as the compiler for the GNU operating system.

G++ is the traditional nickname of GNU C++, a freely redistributable C++ compiler. It is part of GCC, the GNU compiler suite, and is currently part of that distribution.

2.2.1.5 About EMACS Editor

It is a powerful and efficient editor. Variants exist for both test and graphical interface (XEMACS) EMACS is very configurable and has a rich macro language. EMACS is controlled by some commands.

2.2.1.7About gedit editor

Text Editor (gedit) is the default GUI text editor in the Ubuntu operating system. It is UTF-8 compatible and supports most standard text editor features s well as many advanced features. These include Multilanguage spell checking, extensive support of syntax highlighting, and a large number of official and third party plug-ins. Gedit is suited for both basic and more advanced text editing and is released under the GNU General Public License.

2.2.1.8 About Open CV

Open CV is a computer vision library originally developed by Intel. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on *real-time* image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these commercial optimized routines to accelerate itself.

Open CV runs under FreeBSD, Linux, Mac OS and Windows. The user can get official releases from source forge, or take the current snapshot under SVN from there. Open CV now uses Cmake.

The library is mainly written in C, which makes it portable to some specific platforms such as Digital signal processor. But wrappers for languages such as C# and Python have been developed to encourage adoption by a wider audience.

Open CV's application areas include:

- 2D and 3D feature toolkits
- Ego motion estimation
- Facial recognition system
- Gesture recognition
- Human-Computer Interface (HCI)
- Mobile robotics

2.2.1.6 About Dia

Dia is a free and open source general purpose diagramming software, developed as part of the GNOME projects office suite and was originally created by Alexander Larsson. Dia uses a controlled single document interphase (SCDI) similar to GIMP. Dia has a

modular design with several shape packages available for different needs: flowchart, network diagrams, circuit diagrams, and more.

It does not restrict symbols and connectors from various categories from being placed together. Dia can be used to draw many different kinds of diagrams.it has special objects to help draw entity relationship models, Unified Modelling Language(UML) diagrams, flowcharts etc. it is also possible to add support for new shapes by writing simple XML files.

2.2.2 **Hardware specification for development, implementation**

Processor	:	Pentium IV or above.
Motherboard	:	Suitable for Processor.
RAM	:	512 MB.
Hard Disk	:	80 GB.
Keyboard	:	Multimedia keyboard.
Monitor	:	Non-Interrelated 14' ' colour monitor

3. SYSTEM DESIGN AND MODELLING

3.1 Design Methodologies

The purpose of system design phase is to plan a solution of the problem specified by the requirements document. The design activity results in three separate outputs- architecture design, high level design and detailed design. Architecture focuses on looking at a system as a combination of many different components and how they interact with each other to produce the desired results. The high level design identifies the modules that should be built for developing the system and specification of these modules. At the end of system design, all the data structures, file formats are all fixed. In detailed design the internal logic of each module is specified.

The approach used for the development of this project is top-down approach. A good plan of attack for designing the algorithm is to break down the task to be accomplished into a few sub tasks, decompose each of these sub tasks into smaller sub tasks and so forth. One of the main advantages of dividing a programming task into sub task is efficiency and easiness of algorithm design. Also different people can work on different task.

3.1.1 Requirement analysis

Requirement analysis is done in order to understand the problem a software system is to solve. The emphasis in requirement analysis is on identifying what is needed from the system, not how the system will achieve its goals.

We refined the project goals so that the various functions needed were identified and its expected operations analysed. We also analysed the user needs.

3.1.2 Architecture

Architectural design provides the blueprint for design with necessary specifications of hardware, software, people or other resources. Multiple architectures are evaluated before one is selected. Architecture is evaluated considering the difference between architectural design values and intentions.

We were initially required to select an operating platform. Considering the benefits of Open Source Software, the environment we selected was G++ (The GNU Compiler Collection) running on Ubuntu 10.10.

3.1.3 High-level design

Our next task was to identify the modules required to accomplish our goals:

3.1.3.1 Loading an image

This module is used to load an image that is already there in the

3.1.3.2 Detecting the lines and other 3 sided and four sided figures

This module detects the triangles, rectangles, squares and rhombus

3.1.3.3 Finding the dimensions and coordinates of each figure

The dimensions such as the length, width, height, angle etc are calculated and the coordinates are found out

3.1.3.4 Adjusting the figures according to the coordinate system of dia

The detected figures are adjusted to the ones that are supported by dia

3.1.3.5 Generate the corresponding dia diagrams

From these adjusted figures the dia diagrams are generated

3.1.4 Detailed design

3.1.3.1 Loading an image

A 3-channel image is loaded into the program by using the OpenCv function cvLoadImage(). The image is then converted into a 1-channel image for computation purpose. This 1-channel image is displayed in a new window in the output screen.

3.1.3.2 Detecting the lines and other 3 sided and four sided figures

The edges of each figure are detected using cvCanny() function. These edges are used to detect lines. This is done using Hough Transforms. The coordinate of the lines are found.

Depending upon the intersection of the lines, three sided or four sided figures are detected. If there are 3 intersection points then it’s a triangle.

Similarly if there are four intersection points it can be a rectangle, rhombus, parallelogram or square

3.1.3.3 Finding the dimensions and coordinates of each figure

The length of each line and it's orientation is calculated and depending upon the dimensions the four sided figures are identified. If all sides are equal and the two angles are perpendicular to the x-axis then it's a square.

If the lines are at an angle of 45 degrees with respect to the x-axis then it's a rhombus.

If only the opposite sides are equal then it's a rectangle.

3.1.3.4 Adjusting the figures according to the coordinate system of dia

Dia has a limited set of figures. In order to generate the corresponding dia diagram the detected figure are adjusted. The triangles are adjusted as isosceles triangle.

3.1.3.5 Generate the corresponding dia diagrams

Each figure has a corresponding xml code. The dimensions of the figures calculated are passed to these xml codes and the respective dia diagrams are created.

3.2 System Architecture

The system architecture basically explains the architecture used to solve the problem. The operating system used is Linux, which is more secure than operating systems. The image which is already loaded in the system is processed by the operating system. The operating system used here is linux which is more secure than the other operating systems. The processed image is then used for the further development by using OpenCV library. The developed image is then processed in dia. The resulting image is displayed on the monitor.

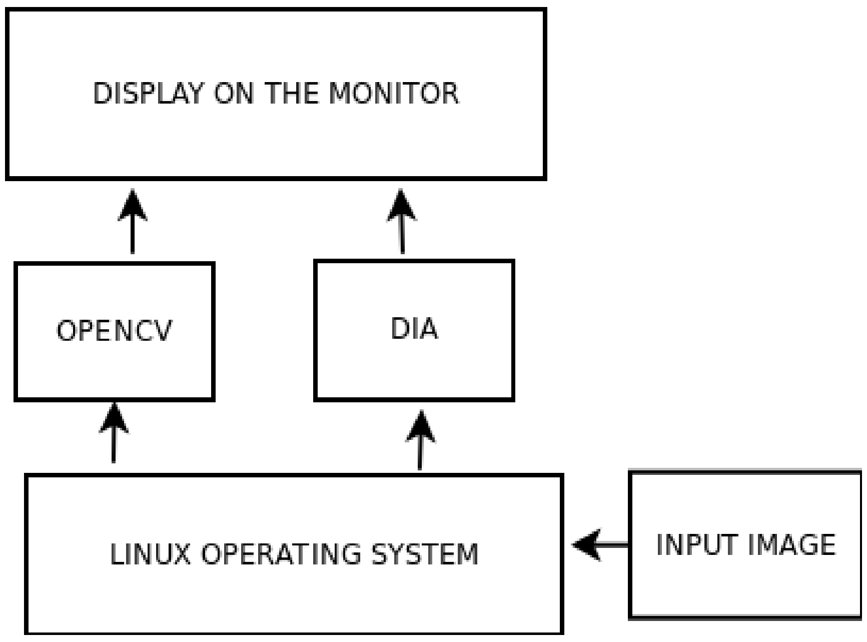


Fig: 3.2: System Architecture

3.3 Control Flow Diagram

Information about a system or program being developed is often communicated from the customer to the developer in graphic form. The diagram types described here are not the only ones that may be used. However, most diagrams in vogue offer one or more of the following concepts to accentuate a particular aspect of a system or computer program under development.

Control flow diagrams break a process down to a finite number of steps that get executed one at a time. In a single-threaded system, only one of the steps is being executed. The control flow governs how the next step to be executed is determined.

Control flow does not say anything about what the inputs and outputs of steps A and B are. It does not address why we might want to perform step A and then step B. It does not address how steps A and B get performed internally. In general, when a step completes, only a single arrow emerges from the box. A special case step called a “decision” will cause one of two or more paths to be followed based upon some condition that exists when the decision is executed. These diagrams start and stop with terminal steps.

There are several types of control flow diagrams, for example:

- Change control flow diagram, used in project management.
- Configuration decision control flow diagrams are used.
- Process control flow diagrams.
- Quality control flow diagrams.

In software and systems development control flow diagrams can be used in control flow analysis, data flow analysis, algorithm analysis, and simulation. Control and data flow analysis are most applicable for real time and data driven systems. These flow analyses transform logic and data requirements text into graphic flows which are easier to analyze than the text. PERT, state transition and transaction diagrams are examples of control flow diagram

3.3.1 Types of control flow diagrams

3.3.1.1 Process Control Flow Diagram

A flow diagram can be developed for the process control system for each critical activity. Process control is normally a closed cycle in which a sensor provides information to a process control software application through a communications system. The application determines if the sensor information is within the predetermined (or calculated) data parameters and constraints. The results of this comparison are fed to an actuator, which controls the critical component. This feedback may control the component electronically or may indicate the need for a manual action.

3.3.1.2 Performance seeking control flow diagram

The control law consists of estimation, modelling, and optimization processes. In this estimator, the inputs, outputs, and residuals were recorded. At the compact propulsion system modelling stage, all the estimated inlet and engine parameters were recorded. In the optimization phase, the operating condition constraints, optimal solution, and linear programming health status condition codes were recorded. Finally, the actual commands that were sent to the engine through the DEEC were recorded.

Control flow diagram:

The figure 3.3 explains the control flow of converting a rough diagram to a neat diagram. The rough diagram is loaded from a file. It is then processed in Opencv and Dia to generate the neat sketch.

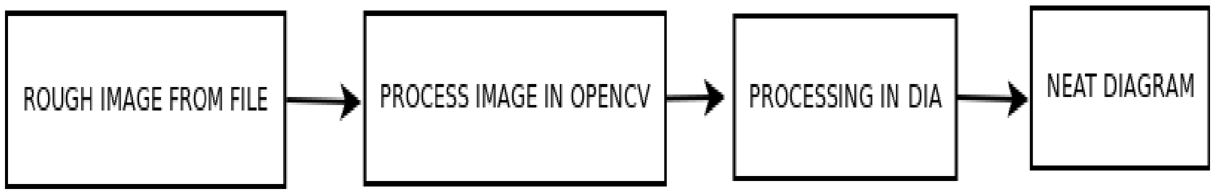


Fig: 3.3: Control Flow Diagram

3.4 Data Flow Diagram

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

A DFD provides no information about the timing or ordering of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD).It is common practice to draw a context-level data flow diagram first, which shows the interaction between the system and external agents which act as data sources and data sinks. This context-level DFD is next "exploded", to show more detail of the system being modelled.

Data-flow diagrams (DFDs) are one of the three essential perspectives of the structured-systems analysis and design method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data-flow diagram, users are able to visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's dataflow diagrams can be drawn up and compared with the new system's data-flow diagrams to draw comparisons to implement a more efficient system. Data-flow diagrams can be used to provide the end user with a physical idea of where the data they input ultimately has an effect upon the structure of the whole system from order to dispatch to report. How any system is developed can be determined through a data-flow diagram.

There are only four symbols:

1. Squares representing external entities, which are sources or destinations of data.
2. Rounded rectangles representing processes, which take data as input, do something to it, and output it.
3. Arrows representing the data flows, which can either be electronic data or physical items.
4. Open-ended rectangles representing data stores, including electronic stores such as databases or XML files and physical stores such as or filing cabinets or stacks of paper.

The general rules to follow while drawing a DFD are :

1. All processes must have at least one data flow in and one data flow out.
2. All processes should modify the incoming data, producing new forms of outgoing data.
3. Each data store must be involved with at least one data flow.
4. Each external entity must be involved with at least one data flow.
5. A data flow must be attached to at least one process.

The four key elements in DFD are processes, data flows, data stores & external entities.

External Entity

An external entity is a source or destination of a data flow which is outside the area of study.

Process

A process shows a transformation or manipulation of data flows within the system. The symbol used is a rectangular box.

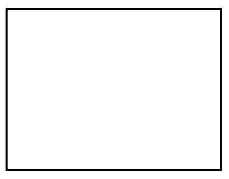
Data Flow

A data flow shows the flow of information from its source to its destination. A data flow is represented by a line, with arrowheads showing the direction of flow. Information always flows to or from a process .

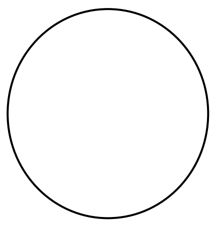
Data Store

A data store is a holding place for information within the system: It is represented by an open ended narrow rectangle. Data stores may be long-term files such as sales ledgers, or may be short-term accumulations.

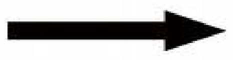
DFD is used to define the flow of the system and its resources such as information. DFDs are a way of expressing system requirements in a graphical manner.It has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. In normal convention, logical DFD can be completed using only 4 notations.



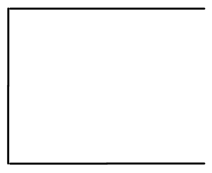
A square represents a data source or destination



A closed figure represent process that transform the control.



A directed line represents process that transform the data stream



A rectangle representing data stores, including electronic stores such as databases or XML files and physical stores

Fig 3.4.1: Data Flow Diagram Symbols

In the figure 3.4.2 a rough diagram is loaded as a 3-channel image in OpenCV. It is then converted to a single channel image and processed thereafter. The lines are detected which is further used to detect three sided and four sided figures. The figures are adjusted according to the needs of the dia software. The XML code is generated for each figure detected. This gives a neat dia diagram.

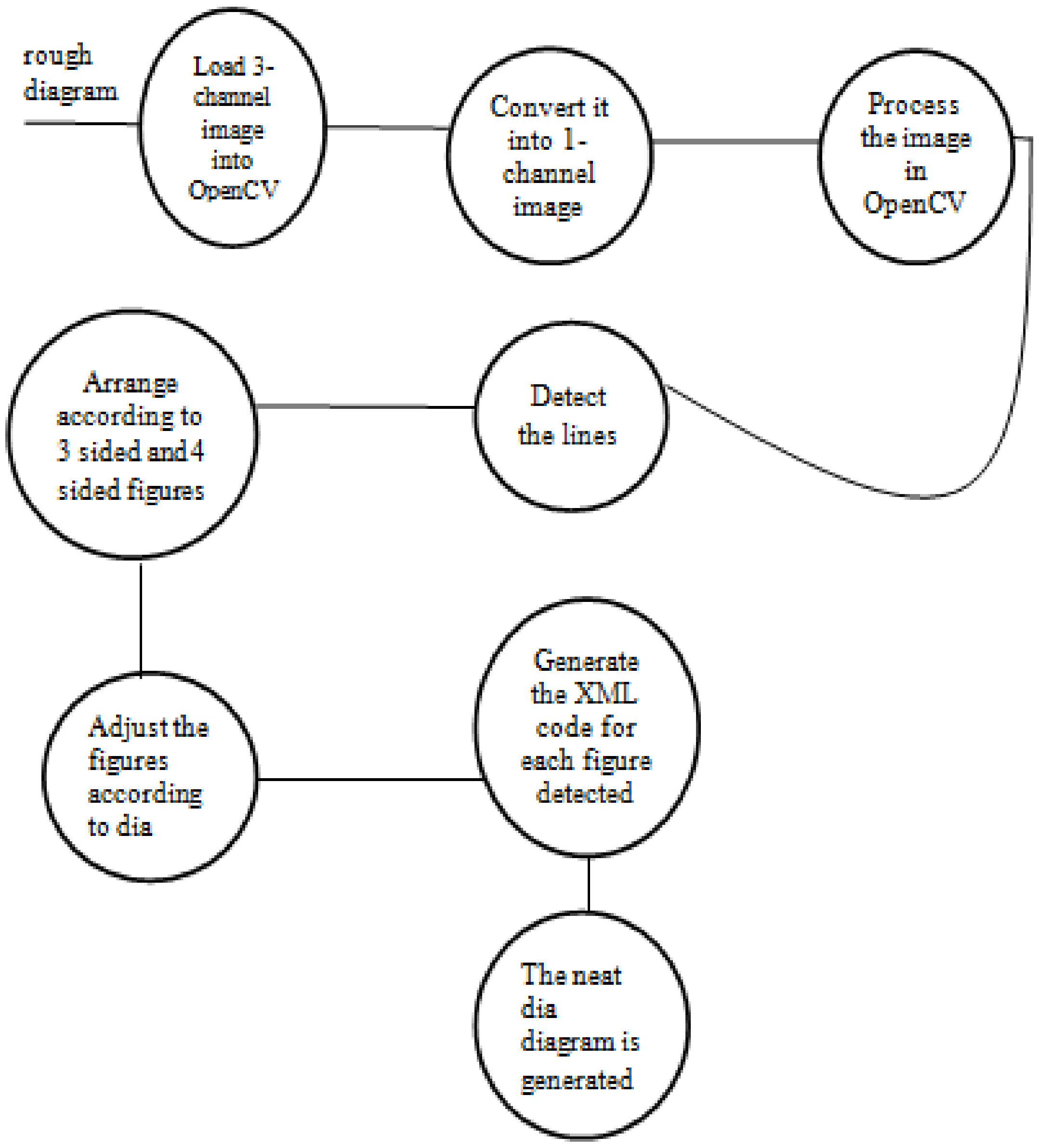

rough diagram
Load 3-channel image into OpenCV
Convert it into 1-channel image
Process the image in OpenCV
Detect the lines
Arrange according to 3 sided and 4 sided figures
Adjust the figures according to dia
Generate the XML code for each figure detected
The neat dia diagram is generated

Fig 3.4.2: Data Flow Diagram

3.5 Layer Explanation Diagram

The layer explanation diagram describes the different layers involved in the solving of the problem. It shows the different layers through which it passes to finally give the solution.

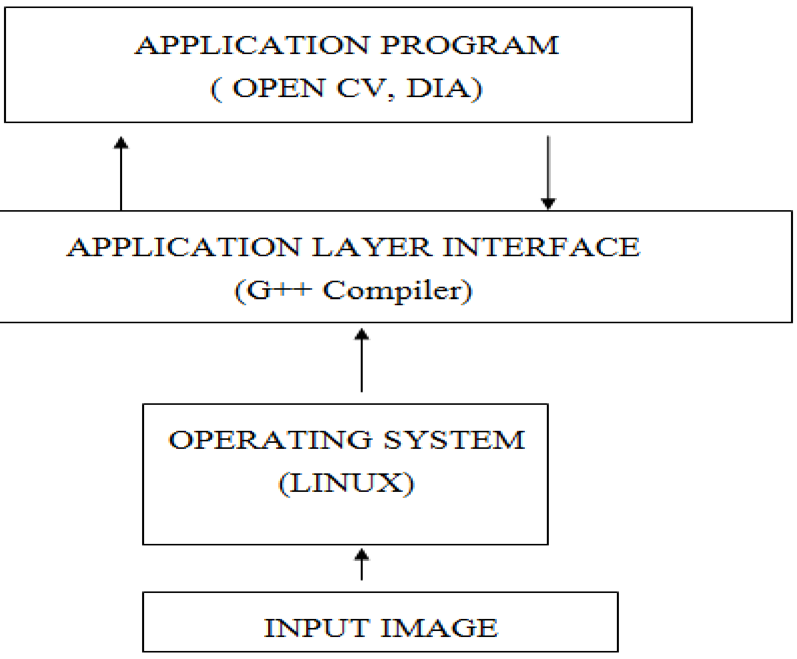


Fig 3.5: Layer Explanation

3.6 Application Design Diagram

3.6.1 General definition

Application design is a process of problem-solving and planning for a software solution. An application design may be platform-independent or platform-specific, depending on the availability of the technology called for by the design. The design concepts provide the software designer with a foundation from which more sophisticated methods can be applied. A set of fundamental design concepts has evolved. They are:

3.6.1.1 *Abstraction*- Abstraction is the process or result of generalization by reducing the information content of a concept or an observable phenomenon, typically in order to retain only information which is relevant for a particular purpose.

3.6.1.2 *Refinement*- It is the process of elaboration. A hierarchy is developed by decomposing a macroscopic statement of function in a stepwise fashion until programming language statements are reached. In each step, one or several instructions of a given program are decomposed into more detailed instructions. Abstraction and Refinement are complementary concepts.

3.6.1.3 *Modularity*- Software architecture is divided into components called modules.

3.6.1.4 *Software Architecture*- It refers to the overall structure of the software and the ways in which that structure provides conceptual integrity for a system. A software architecture is the development work product that gives the highest return on investment with respect to quality, schedule and cost.

3.6.1.5 *Control Hierarchy*- A program structure that represent the organization of a program component and implies a hierarchy of control.

3.6.1.6 *Structural Partitioning*- The program structure can be divided both horizontally and vertically. Horizontal partitions define separate branches of modular hierarchy for each major program function. Vertical partitioning suggests that control and work should be distributed top down in the program structure.

3.6.1.7 *Data Structure*- It is a representation of the logical relationship among individual elements of data.

3.6.1.8 *Software Procedure*- It focuses on the processing of each modules individually

3.6.1.9 *Information Hiding*- Modules should be specified and designed so that information contained within a module is inaccessible to other modules that have no need for such information.

3.6.2 Project design

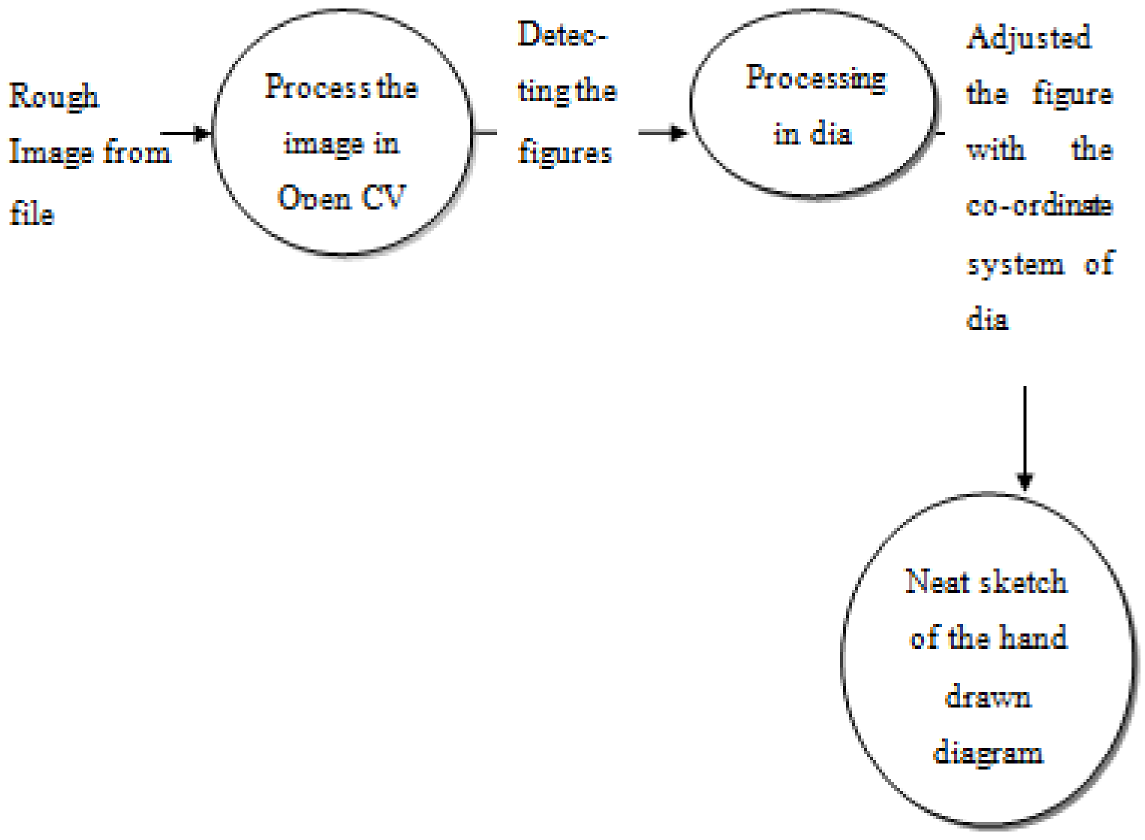

Rough Image from file
Process the image in Open CV
Detec-ting the figures
Processing in dia
Adjusted the figure with the co-ordinate system of dia
Neat sketch of the hand drawn diagram

Fig 3.6.2: Project Design

3.7 Flowchart

3.7.1 General Definition

A flowchart is a common type of diagram that represents an algorithm or process showing the steps as boxes of various kinds, and their order by connecting these with arrows. This diagrammatic representation can give a step-by-step solution to a given problem. Data is represented in these boxes, and arrows connecting them represent flow / direction of flow of data. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields.

3.7.1.1 *Symbols*

A typical flowchart may have the following kinds of symbols:

- Start and end symbols

  Represented as circles, ovals or rounded rectangles, usually containing the word "Start" or "End", or another phrase signalling the start or end of a process, such as "submit enquiry" or "receive product".
- Arrows

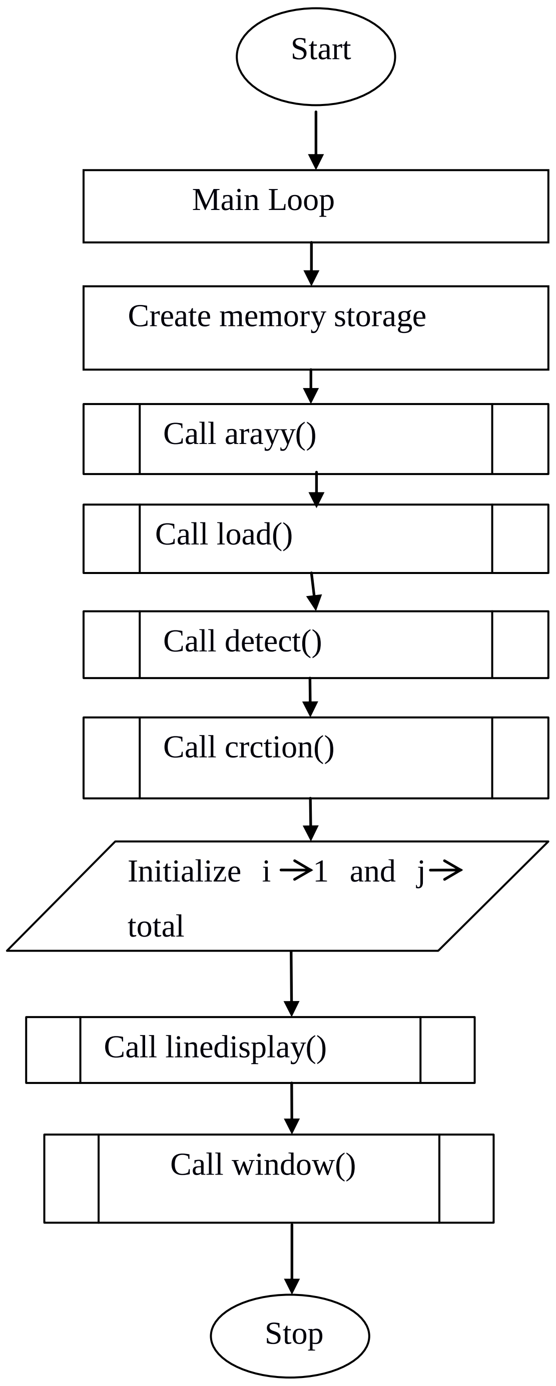
  Showing what's called "flow of control" in computer science. An arrow coming from one symbol and ending at another symbol represents that control passes to the symbol the arrow points to.
- Processing steps

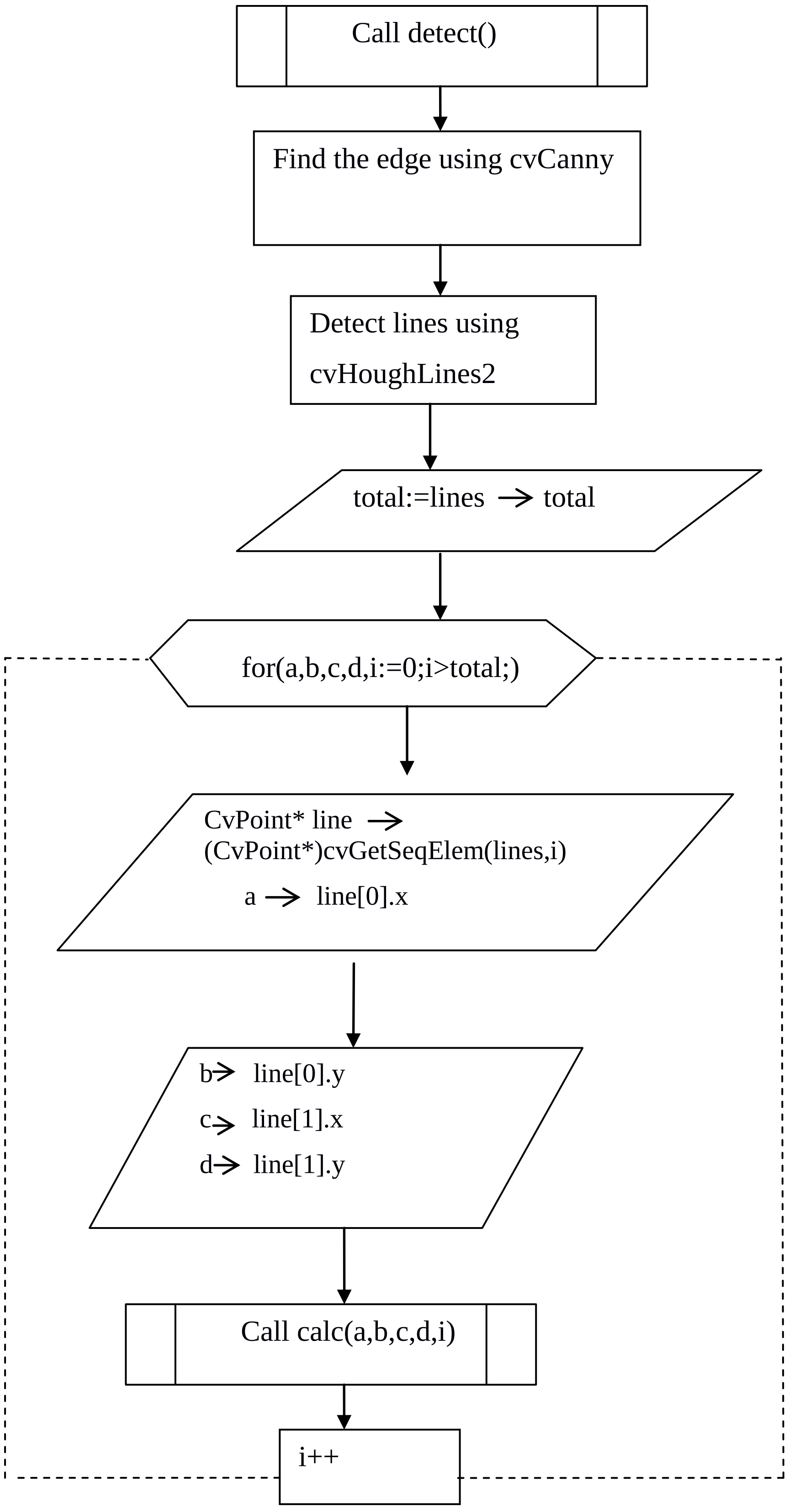
  Represented as rectangles (or oblongs).
- Input/Output

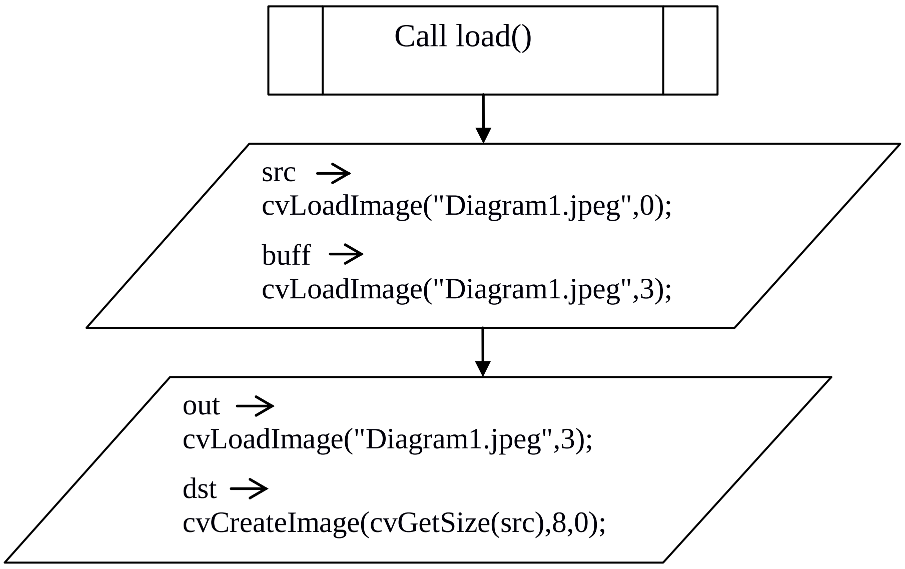
  Represented as a parallelogram.
- Conditional or decision

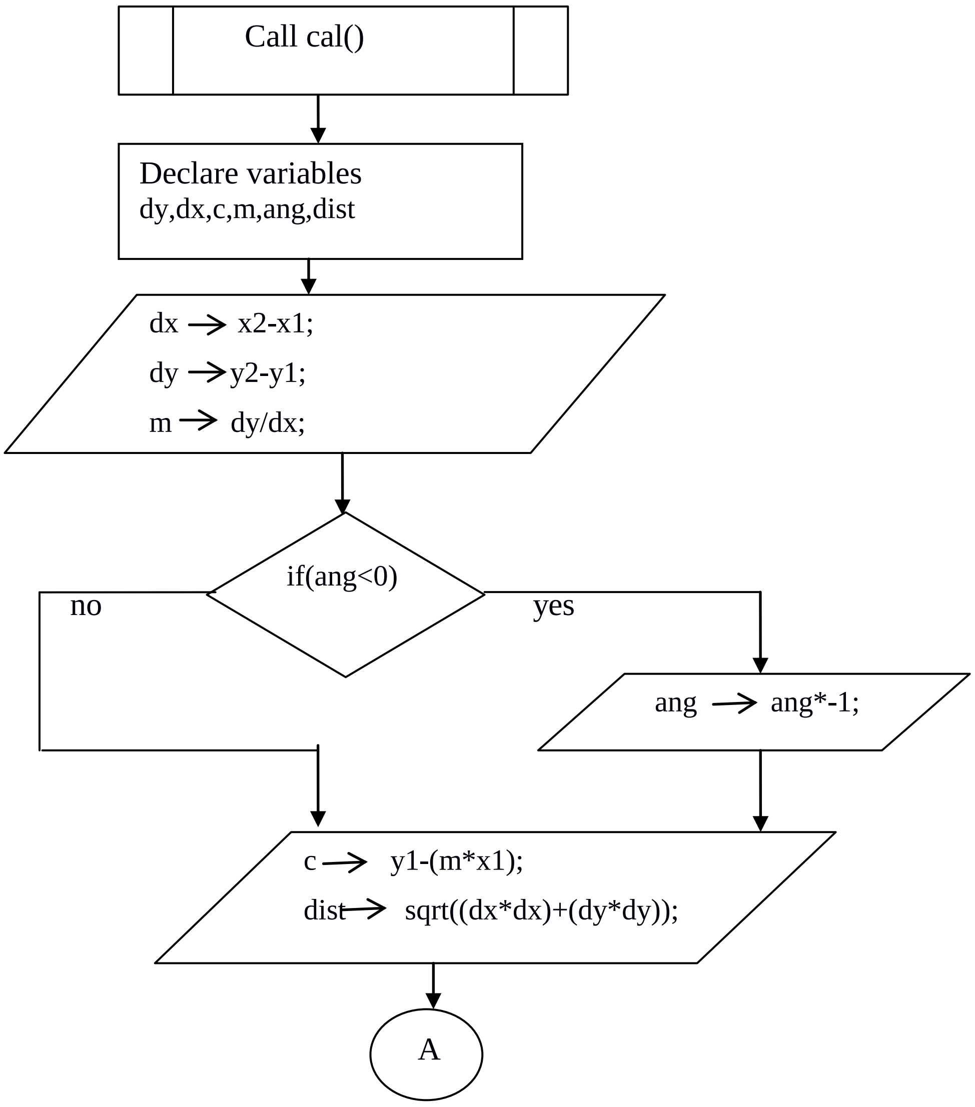
  Represented as a diamond (rhombus). These typically contain a Yes/No question or True/False test. This symbol is unique in that it has two arrows coming out of it, usually from the bottom point and right point, one corresponding to Yes or True, and one corresponding to No or False. The arrows should always be labelled. More than two arrows can be used, but this is normally a clear indicator that a complex decision is being taken, in which case it may need to be broken-down further, or replaced with the "pre-defined process" symbol.

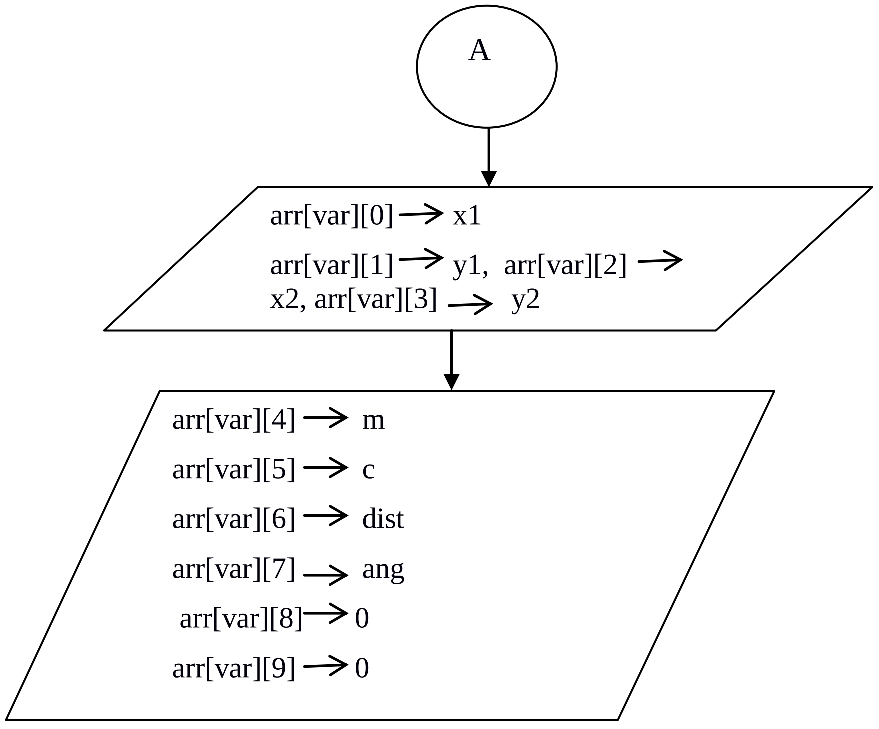
3.7.2 Project flowchart

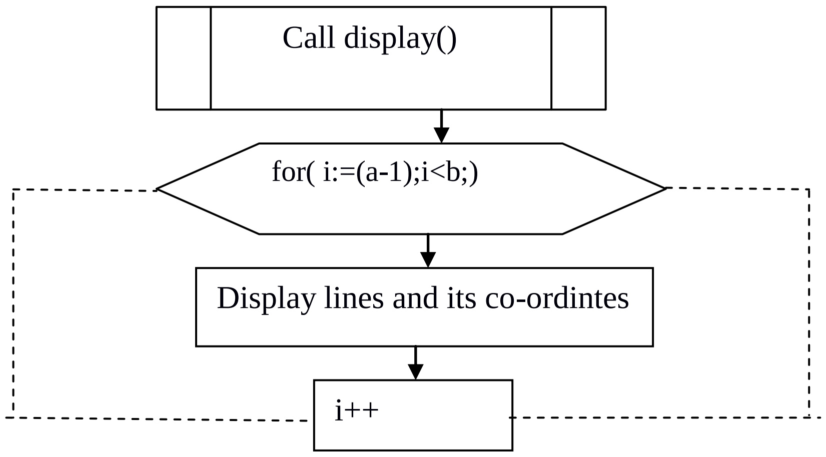


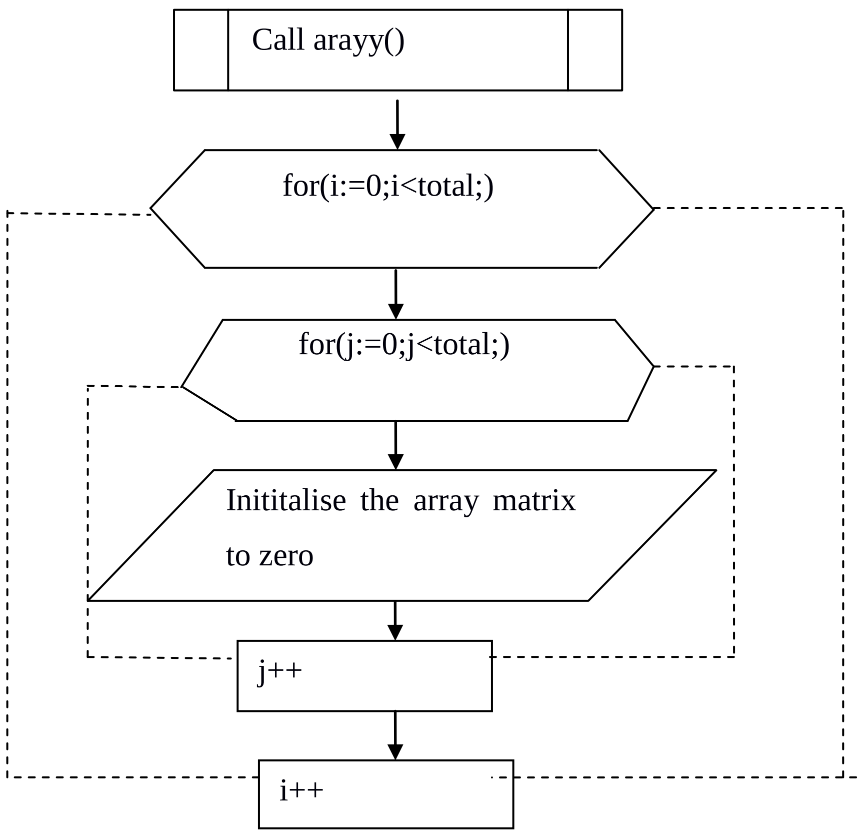

Call detect()
Find the edge using cvCanny
Detect lines using cvHoughLines2
total:=lines → total
for(a,b,c,d,i:=0;i>total;)
CvPoint* line → (CvPoint*)cvGetSeqElem(lines,i)
a → line[0].x
b→ line[0].y
c→ line[1].x
d→ line[1].y
Call calc(a,b,c,d,i)
i++


Call load()
src → cvLoadImage("Diagram1.jpeg",0);
buff → cvLoadImage("Diagram1.jpeg",3);
out → cvLoadImage("Diagram1.jpeg",3);
dst → cvCreateImage(cvGetSize(src),8,0);


Call cal()
Declare variables
dy,dx,c,m,ang,dist
dx → x2-x1;
dy → y2-y1;
m → dy/dx;
if(ang<0)
no
yes
ang → ang*-1;
c → y1-(m*x1);
dist → sqrt((dx*dx)+(dy*dy));
A


A
arr[var][0] → x1
arr[var][1] → y1, arr[var][2] → x2, arr[var][3] → y2
arr[var][4] → m
arr[var][5] → c
arr[var][6] → dist
arr[var][7] → ang
arr[var][8] → 0
arr[var][9] → 0


Call display()
for(i:=(a-1);i<b;)
Display lines and its co-ordintes
i++



3.8 Technology Explanation

3.8.1 Open CV

Open CV is a computer vision library originally developed by Intel. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these commercial optimized routines to accelerate itself. The library is mainly written in C, which makes it portable to some specific platforms such as Digital signal processor. But wrappers for languages such as C# and Python have been developed to encourage adoption by a wider audience. The main features of Open CV are Image data manipulation, matrix and vector manipulation and linear algebra routines, Structural analysis, basic GUI, image labelling, object recognition.

Example applications of the Open CV library are Human-Computer Interaction, segmentation and recognition, face recognition, gesture recognition, camera and motion tracking, ego motion, motion understanding, structure from motion (SFM), stereo and multi-camera calibration and depth computation, mobile robotics.

3.8.2 Dia

Dia is free and open source general-purpose diagramming software. Dia has a modular design with several shape packages available for different needs: flowchart, network diagrams, circuit diagrams, and more. It does not restrict symbols and connectors from various categories from being placed together.

Dia can be used to draw many different kinds of diagrams. It can load and save diagrams to a custom XML format, can export diagrams to a number of formats, including EPS, SVG, XFIG, WMF and PNG, and can print diagrams (including ones that span multiple pages).

4. TESTING

4.1 Introduction

Testing is one of the major hurdles in the development of the system. Testing is the process of finding errors in the system. Only error free software will be stable for a long time. There are different types of techniques for finding the bugs in software. Testing presents an interesting anomaly for the software. Testing is vital to the success of the system. Errors can be injected at any stage during development. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. During testing, the program to be tested is executed with set of data and the output of the program for the test data is evaluated to determine if the programs are performing as expected. A series of testing are performed for the proposed system before the system is ready for user acceptance testing.

4.1.1 System testing

A test plan has the following steps

- Prepare test plan
- Specify conditions for user acceptance testing
- Prepare test data for program testing
- Prepare test data for transaction path testing
- Plan user training
- Compile/assemble programs
- Prepare job performance aids
- Prepare operational documents

4.1.1.1 Syntax testing

We use syntax testing to eliminate errors in software. In the system we have input fields like text, numeric fields. We should allow only numeric fields and should avoid any alphabets. The program won't accept any alphabets in a numeric field. System testing involves unit testing, integration testing, acceptance testing.

Careful planning and scheduling are required to ensure that modules will be available for integration into the evolving software.

4.1.1.2 Unit testing

Unit testing comprises the set of tests performed by an individual programmer prior to integration of the unit into a larger system .these are four categories of tests that can be performed on a program unit.

- Functional unit
- Performance unit
- Stress unit
- Structure unit

4.1.1.3 Integration testing

Integration testing involves bottom-up integration, top-down integration and sandwich integration strategy.

Bottom-up integration the traditional strategy used to integrate the components of software system into a functioning whole. Top-down integration starts with main routine and one or two immediate subroutine in the system structure. Sandwich integration is predominantly top-down, but bottom-up techniques are used in some modules and sub systems.

4.1.1.4 Acceptance testing

Acceptance testing involves planning and execution of functional tests, performance tests and stress tests in order to demonstrate the implemented system satisfies its requirements. Functional tests causes involve exercising the code with nominal files for which the expected outputs are known. Thus the software system developed in the above manner is one that satisfies the user needs, confirms to its requirements and design satisfactions and exhibits an absence of errors.

4.1.2 Code efficiency

The software design has to be error free and extremely planned for the programs to execute efficiently. A design, which is not planed enough, will lead the programmers in various directions and the completion of the project will be delayed. More than that the programs will be giving error full reports and the programs can crash also.

This is something like a building, which doesn’t have a strong foundation and plan. Unless otherwise the plan of the building is fixed and the construction work goes according to the plan, there will be serious setbacks in the building construction, and once the plan is fixed and approved there should not be any change in the plan.

Similarly the system has to be studied with at most care and the designing of the system should be stable. This helps the programmer to finish the project in time and the coding will be error free and stable. So the software will not be crashed in the future and maintenance cost of the system will be very minimum.

The coding has to be done by expert programmers, and there has to be strict conventions should be followed while programming. The code should be the shortest possible and fast.

One major aspect is the documentation part. The code has to be well documented so that anybody else following the code for some modifications won’t find it difficult to do the task.

In this project very much care has been taken to ensure the efficiency and the speed of the code execution and documentation of the system is precise and clear.

4.1.3 Maintenance

Once the software is fully developed and implemented, the company starts to use the software. The company also grows and more divisions can be attached to the company, or the database of the company can grow in size. So after sometime the software, which has been installed, needs some modification. If the software needs modification all the steps needed to develop new software has to be executed.

The need has to be studied, the design has to be made and coding has to be done. Then the new module has to be connected to the existing software modules.

Once the software is working perfect also we have to do routine testing and any new bug is found out, immediately it has to be fixed. No software ever developed will be bug free forever. When a new situation arises, the software can create an error, but if it's found out and repaired the software will not be causing more problems. Always maintenance has to be done on the software, for to make the software work perfectly without any errors.

4.2 Test Cases

The different levels of testing, attempt to detect different types of faults. Different test cases are applied on the project to check its efficiency.

4.2.1 Positive Test cases

- Hand drawn circuit diagrams and flow charts can be digitized

4.2.2 Negative Test Cases

- It can be used only for closed figure

5. IMPLEMENTATION

5.1 Introduction

Implementation is an activity that is contained throughout the development phase. It is a process of bringing a developed system into operational use and turning it over to the user. The new system and its components are to be tested in a structured and planned manner. A successful system should be delivered and users should have confidence that the system would work efficiently and effectively. The more complex the system being implemented the more involved will be the system analysis and design effort required for implementation.

There are three types of implementation that exist:

- Implementation of a computer system to replace a manual system. The problems encountered are converting files, training users, creating accurate files, and verifying printouts for integrity.
- Implementation of a new computer system to replace an existing one. This usually a difficult conversion. If not properly planned, there can be many problems. Some large computer systems have taken as long as year to convert.
- Implementation of a modified application to replace an existing one using the same computer. This type of conversion is relatively easy to handle, provided there are no major changes in the files.

Normally this stage involves setting up a coordinating committee, which will act as a sounding board for ideas; complaints and problems. The first task is implementation planning; i.e. deciding on the methods and timescale to be adopted. Apart from planning, the two major tasks of preparing for implementation are education and training of users and testing of the system. Education of users should really have taken place much earlier in the project ; at the implementation stage the emphasis must be on training in new skills to give staff confidence they can use the system. Once staff has been trained, the system can be tested.

5.2 Implementation Plan

In today's world everybody needs diagrams. Most users need to create one more often than they think. Getting more technical, there are always circuits and blueprints and the like. Without wasting time with an office app, the GIMP, or a paint program Dia, can be used which is an easy yet powerful made-for-diagrams editor.

6. CONCLUSION

6.1 Advantages and Disadvantages of the System

OpenCV that is used is a library with variety of uses and advantages. OpenCV's objects are safely deleted when not used. There is no need to invoke cvRelease().

The project gives an additional input option. A simple paper and pen could be used to draw diagrams and input into the system.

6.2 Future Scope

The project could be helpful in the education field. Presently there exists no effective system to directly transform a diagram in a book to digital format. This could be easily done using our project. So digitalising book becomes easy.

7.0 APPENDIX

7.1 Program Code

```
#include<opencv/cv.h>
#include <opencv/highgui.h>
#include<math.h>
#include<iostream>
#include <stdio.h>
IplImage* out;
IplImage* src;
IplImage* buff;
IplImage* dst;
CvSeq* lines=0;
CvMemStorage* storage=cvCreateMemStorage(0);
float arr[1000][10],intr[1000][7];
void detect();
void load();
void window();
void calc(int,int,int,int,int);
void display(int,int);
void linedisplay(int,int);
void intersect();
void extension();
void araydisply();
int dist(int,int);
int check(int);
void crction();
void arayy();
int total=0,ptl=0,buf=0,aray[1000][1000];
int main() //main program
{
/*the required functions are called*/
```

```
  arayy();
  load();
  detect();
  crction();
  int i=1,j=total;
  linedisplay(i,j);
  window();
  cvWaitKey(0);
}//end of main

void detect()
{
/* the lines of the image are detected and their end points are found*/
  cvCanny(src,dst,50,150,3);//edges of the source image are detected
lines=cvHoughLines2(dst,storage,CV_HOUGH_PROBABILISTIC,1,CV_PI/180,80,30
,10);
  total=lines->total;
  for(int a,b,c,d,i=0;i<total;i++)
    {
      CvPoint* line=(CvPoint*)cvGetSeqElem(lines,i);
      a=line[0].x;
      b=line[0].y;
      c=line[1].x;
      d=line[1].y;
      calc(a,b,c,d,i);
    }
}

void crction()
{
/* Multiple lines are detected for a single line in the function detect. In this function it is
corrected to a single line */
  float cmp=0;
  int flag=0,d,ds;
int b,c,di,v,u,s11,s12,s21,s22,e11,e12,e21,e22;
for(int i=0;i<total;i++)
 {
   for(int j=i+1;j<total;j++)
       {
         flag=0;
         cmp=arr[i][7]-arr[j][7];
         if(cmp<0)
           {
             cmp=cmp*-1;
           }
         if((arr[i][4]>=0)&&(arr[j][4]>=0))
           {
             flag=1;
           }
         if((arr[i][4]<0)&&(arr[j][4]<0))
           {
             flag=1;
           }
         if((cmp<2)&&(flag==1)&&(arr[i][8]==0)&&(arr[j][8]==0))
           {
             di=dist(i,j);
             if((di<10))
                  {
                    if(arr[i][6]>arr[j][6])
                      {
                        b=j;
                      }
                    else
                      {
                        b=i;
                      }
                    arr[b][8]=1;
                    buf++;
                    }
              }
           }
    }
}

void load()
{
/* A three channel image is loaded and then converted to gray scale image*/
  src=cvLoadImage("68.jpeg",0);
  buff=cvLoadImage("68.jpeg",3);
  out=cvLoadImage("68.jpeg",3);
  dst=cvCreateImage(cvGetSize(src),8,0);
}

void calc(int x1,int y1,int x2,int y2,int var)
{
/* The slope, length, angle with respect to x-axis of the line are found and are stored in
the multidimensional array named arr*/
  float dy,dx;
  float c,m,ang,dist;
  dx=x2-x1;
  dy=y2-y1;
  m=dy/dx;
  ang=atan(m)*180/3.14;
  if(ang<0)
    {
      ang=ang*-1;
    }
  c=y1-(m*x1);
  dist=sqrt((dx*dx)+(dy*dy));
  arr[var][0]=x1;
  arr[var][1]=y1;
  arr[var][2]=x2;
  arr[var][3]=y2;
  arr[var][4]=m;
  arr[var][5]=c;
  arr[var][6]=dist;
  arr[var][7]=ang;
    arr[var][8]=0;
  arr[var][9]=0;
}

void display(int a,int b)
{
/* The intersections points found are displayed*/
  for(int i=(a-1);i<b;i++)
    {
      std::cout<<"\n";
      std::cout<<"points : " <<intr[i][2]<<" "<<intr[i][3]<<"\n ";
      int li1=intr[i][0];
      int li2=intr[i][1];
      if(arr[i][8]==0);
      {
            std::cout<<"line "<<li1<<": "<<arr[li1][0]<<" "<<arr[li1][1]<<"
"<<arr[li1][2]<<" "<<arr[li1][3]<<" "<<arr[li1][4]<<" "<<arr[li1][7]<<"\n";
            std::cout<<"line "<<li2<<": "<<arr[li2][0]<<" "<<arr[li2][1]<<"
"<<arr[li2][2]<<" "<<arr[li2][3]<<" "<<arr[li2][4]<<" "<<arr[li2][7]<<"\n";
      }
    }
}

void linedisplay(int a ,int b)
{
/* The detected the line is converted into a neat diagram and is displayed in the dia
diagram editor*/
 char c,h[4000],s="",str[40]="<dia:point val=";
  int i=0,r;
  FILE *f1;
  FILE *f2;
FILE *f3,*f4,*f5;
```

```
  f1=fopen("line1.txt","r");
  while(!(feof(f1)))
    {
      c=fgetc(f1);
      h[i++]=c;
    }
  printf("%d \n",i);
  h[i]='\0';
  //printf("%s",a);
  fclose(f1);
 f2=fopen("lines1.dia","w");
  fwrite(&h,(strlen(h)-1),1,f2);
for(int i=(a-1);i<b;i++)
 {
      CvPoint p1,p2;
      p1.x=arr[i][0];
      p1.y=arr[i][1];
      p2.x=arr[i][2];
      p2.y=arr[i][3];
      if(arr[i][8]==0)
            {
              cvLine(out,p1,p2,CV_RGB(255,0,0),1,8);
              f4=fopen("line2.txt","r");
            r=0;
while((!feof(f4)))
{
c=fgetc(f4);
h[r++]=c;
}
h[r]='\0';
fclose(f4);
fprintf(f2,"\n");
fwrite(&h,(strlen(h)-1),1,f2);
fprintf(f2,"%s",str);
fprintf(f2,"%c",s);
```

```
  fprintf(f2,"%f,%f",(arr[i][0])/20,(arr[i][1])/20);
fprintf(f2,"%c/>",s);
fprintf(f2,"\n");
fprintf(f2,"%s",str);
fprintf(f2,"%c",s);
fprintf(f2,"%f,%f",(arr[i][2])/20,(arr[i][3])/20);
fprintf(f2,"%c/>",s);
f3=fopen("line3.txt","r");
fprintf(f2,"\n");
r=0;
while(!(feof(f3)))
{
c=fgetc(f3);
h[r++]=c;
}
h[r]='\0';
fclose(f3);
fwrite(&h,(strlen(h)-1),1,f2);
            }
    }
f5=fopen("line4.txt","r");
r=0;
while((!feof(f5)))
{
c=fgetc(f5);
h[r++]=c;
}
h[r]='\0';
fclose(f5);
fwrite(&h,(strlen(h)-1),1,f2);
  fclose(f2);
}

int check(int a)
{
```

```
/* The intersection points detected are checked whether they lie on the same line*/
  int flg=0,x,y;
  x=intr[ptl][2];
  y=intr[ptl][3];
  if(arr[a][4]>0)
    {
      if((x>=arr[a][0])&&(y>=arr[a][1])&&(x<=arr[a][2])&&(y<=arr[a][3]))
            {
              flg=1;
            }
    }
  else
    {
      if((x>=arr[a][0])&&(y<=arr[a][1])&&(x<=arr[a][2])&&(y>=arr[a][3]))
            {
              flg=1;
            }
    }

  return(flg);
}

void window()
{
/* the detected lines are displayed on the screen*/
  cvNamedWindow("source",1);
  cvShowImage("source",dst);
  cvNamedWindow("output",1);
  cvShowImage("output",out);
}

void arayy()
{
/* the array aray is initialized to zero*/
  for(int i=0;i<total;i++)
```

```
   {
     for(int j=0;j<total;j++)
          {
            aray[i][j]=0;
          }
   }
}

void extension()
{
/* to extend the lines to ensure intersection*/
   for(int i=0;i<total;i++)
   {
     if(arr[i][8]==0)
          {
            if(arr[i][0]!=arr[i][2])
            {
              int t=arr[i][0]-arr[i][2];
              int p=sqrt(t*t);
              if (p>=2)
                {
                     arr[i][0]=(arr[i][0])-15;
                     arr[i][1]=(((arr[i][4])*(arr[i][0]))+(arr[i][5]));
                     arr[i][2]=(arr[i][2])+15;
                     arr[i][3]=(((arr[i][4])*(arr[i][2]))+(arr[i][5]));

                }
            }
          else
            {
              arr[i][1]=arr[i][1]+15;
              arr[i][3]=arr[i][3]-15;
            }
          }
          }
```

```
}

int dist(int a,int b)
{
/* The distances between the parallel lines are found*/
 int d;
 if(arr[a][0]!=arr[a][2])
   {
    int m=arr[a][4];
    int c1=arr[a][5];
    int c2=arr[b][5];
    d=sqrt(((c2-c1)*(c2-c1))/(1+(m*m)));
   }
 else
   {
    int di=arr[a][0]-arr[b][0];
    if(di>0)
          d=di;
    else
          d=-di;
   }
   return (d);
}
```

7.2 Technology explanation

7.2.1 OpenCV

Open CV is a computer vision library originally developed by Intel. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these commercial optimized routines to accelerate itself. The library is mainly written in C, which makes it portable to some specific platforms such as Digital signal processor. But wrappers for languages such as C# and

Python have been developed to encourage adoption by a wider audience. The main features of Open CV are Image data manipulation, matrix and vector manipulation and linear algebra routines, Structural analysis, basic GUI, image labelling, object recognition.

Example applications of the Open CV library are Human-Computer Interaction, segmentation and recognition, face recognition, gesture recognition, camera and motion tracking, ego motion, motion understanding, structure from motion (SFM),stereo and multi-camera calibration and depth computation, mobile robotics.

7.2.2 Dia

Dia is free and open source general-purpose diagramming software. Dia has a modular design with several shape packages available for different needs: flowchart, network diagrams, circuit diagrams, and more. It does not restrict symbols and connectors from various categories from being placed together.

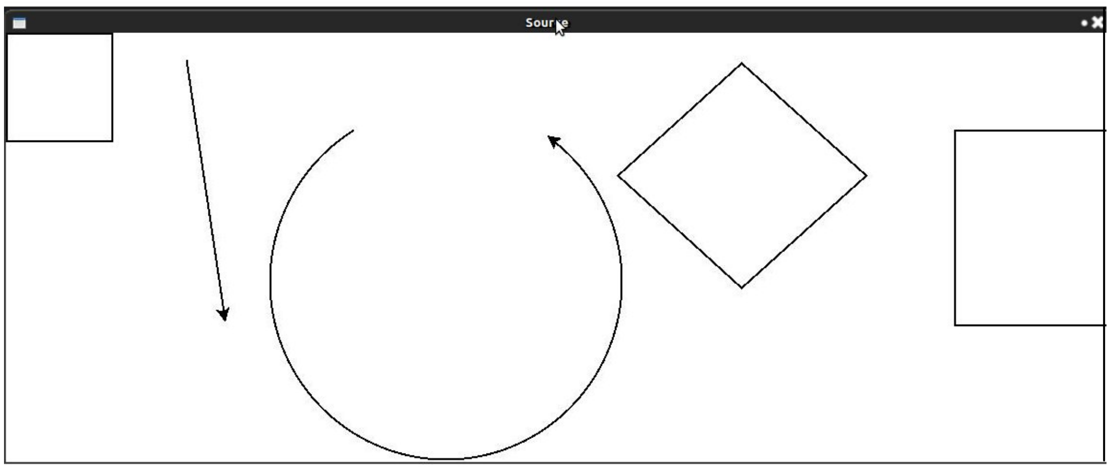
Dia can be used to draw many different kinds of diagrams. It can load and save diagrams to a custom XML format, can export diagrams to a number of formats, including EPS, SVG, XFIG, WMF and PNG, and can print diagrams (including ones that span multiple pages).

7.3 Glossary

7.3.1 Open CV	: Open Computer Vision
7.3.2 UNIX	: UNIX is a computer operating system originally developed in 1969, a group of AT&T employees at Bell Labs.
7.3.3 Linux kernel	: The Linux kernel is an operating system kernel used by Linux family of Unix-like operating systems.

7.3.4	GNU	: The GNU Project was launched in 1984 to develop a complete Unix-like operating system which is free software: the GNU operating system.
7.3.5	GNU GPL	: To make GNU a free software, it need to release under a free software license which is GNU General Public License(GNU GPL).
7.3.6	POSIX	: Portable Operating System Interface for Unix is a related standards specified by IEEE to define the application programming interface(API).
7.3.7	Rendering	: Process of generating an image from a model, by means of computer programs.
7.3.8	Graphics pipeline	: Giving the final appearance to the models.
7.3.9	BSD license	: Berkeley Software Distribution(BSD) represents a family of permissive free software licenses.
7.3.10	Cmake	: Cmake is a cross-platform to build automation.

7.4 Screen Shots



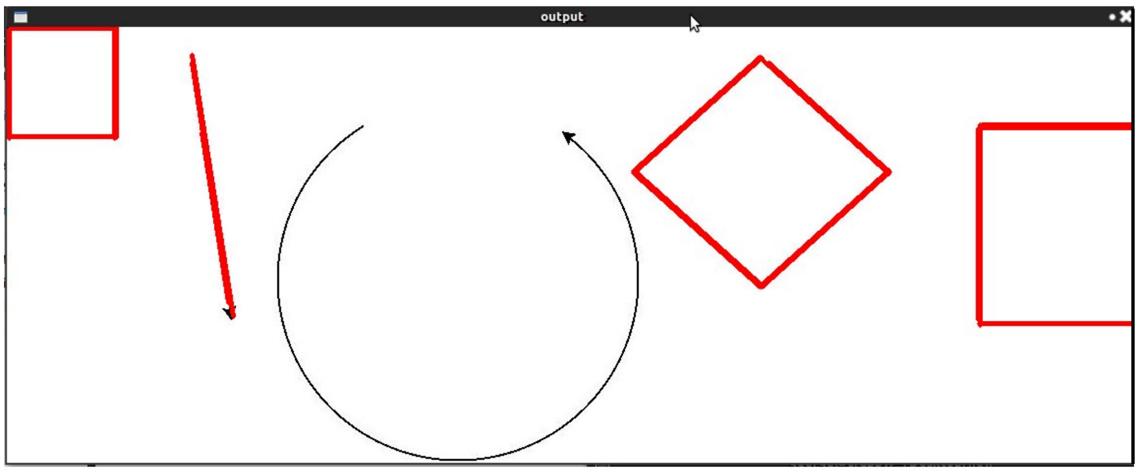
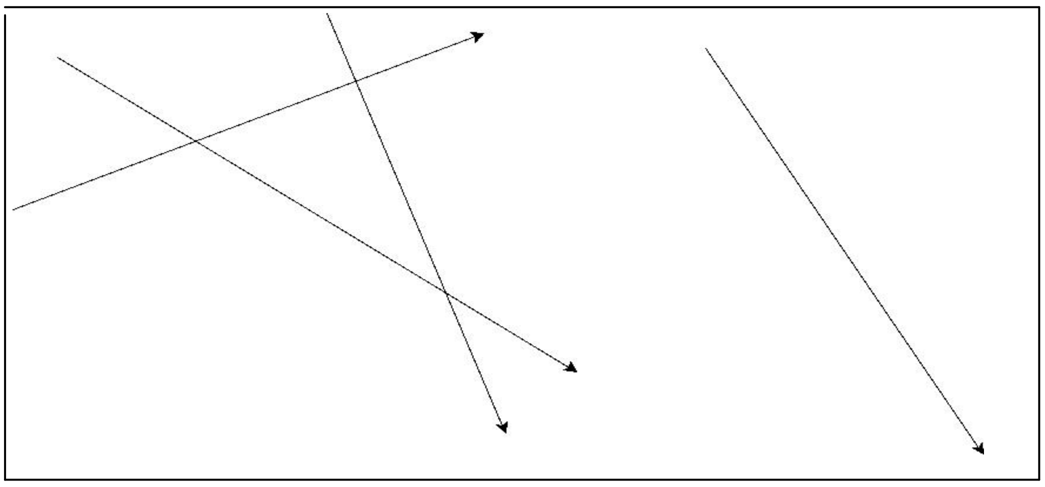


Fig 7.4.1: Proof of Concept Results



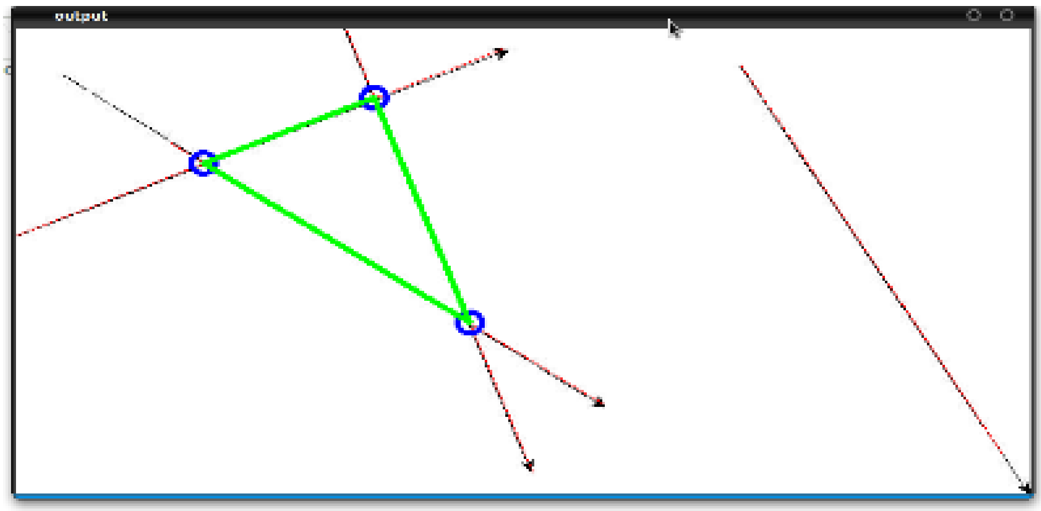
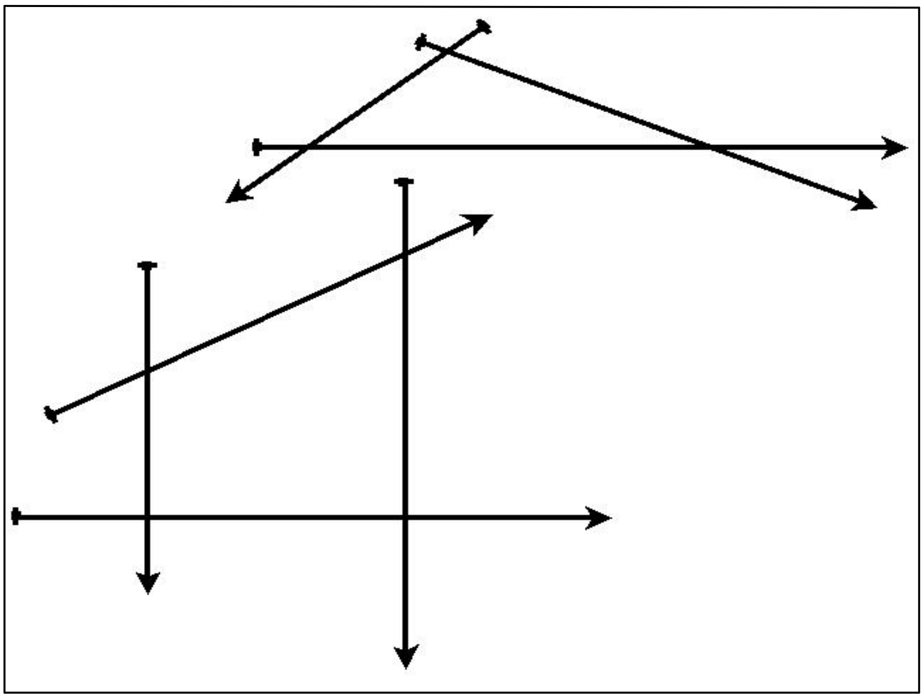
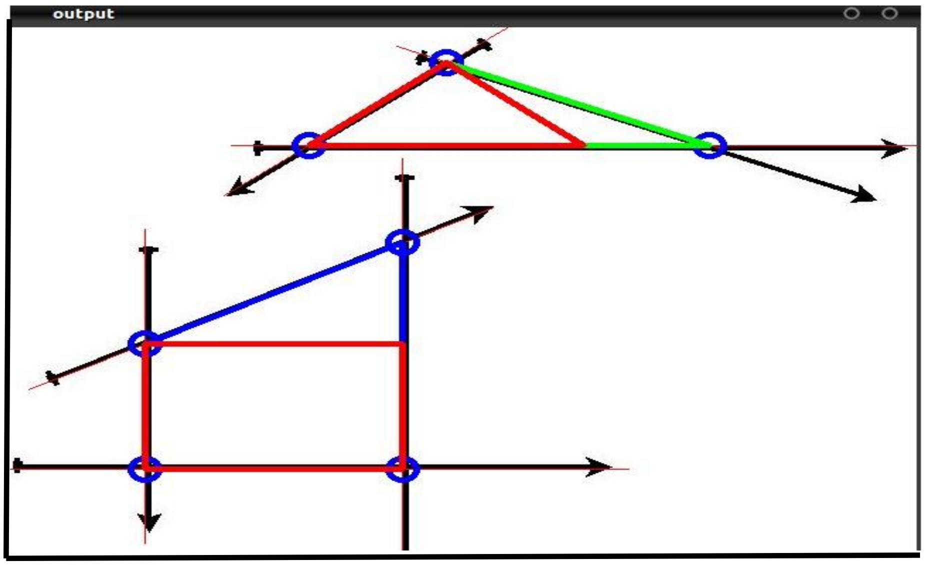


Fig 7.4.2: Detection of Different Figures of a Flowchart





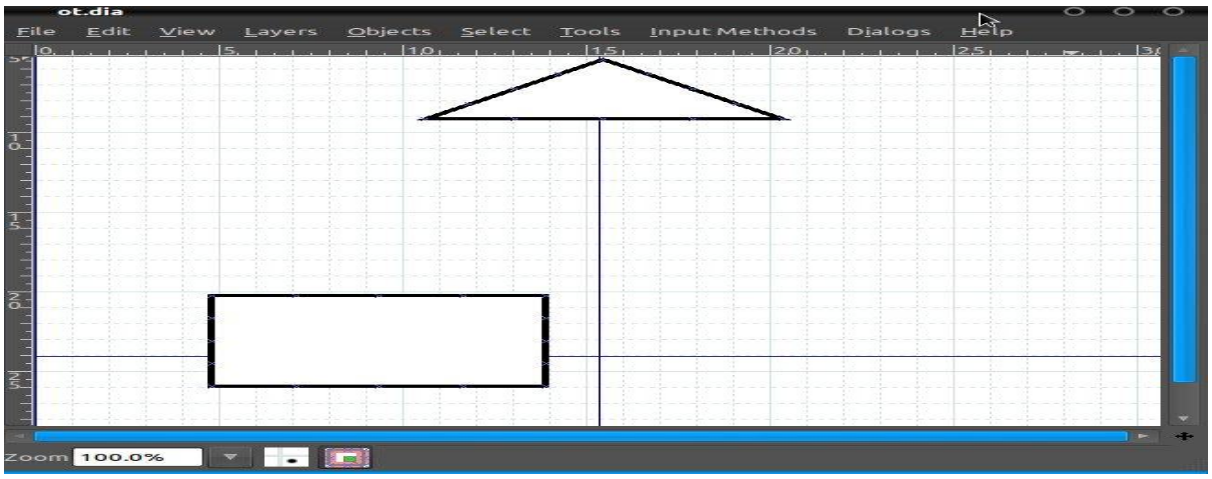

ot.dia
File Edit View Layers Objects Select Tools Input Methods Dialogs Help
Zoom 100.0%

Fig 7.4.4: Adjusting the Figures and Generating the Dia Diagram

REFERENCES

[1]. Learning OpenCV: Computer vision with the OpenCV library

Gary Bradski, Adrian Kaehler *(2008)*

[2]. opencv.org

[3]. [Borgefors86] Gunilla Borgefors. *Distance Transformations in Digital Images.* Computer Vision, Graphics and Image Processing 34, 344-371 (1986).

[4]. [Canny86] J.Canny.*A Computational Approach to Edge Detection*, IEEE Trans. on Pattern Analysis and Machine Intelligence, 8(6), pp.679-698 (1986).