



Department of Information Systems and Computing

BSc (Hons) Computer Science

Academic Year 2005-2006

**An Improved Image Recognition Software System for Implementation in
Bomb-Disposal Robotics**

Mr. Derek A. Colley, Student No. 0214867

A report submitted in partial fulfilment of the requirement for the degree of
Bachelor of Science

Brunel University
Department of Information Systems and Computing
Uxbridge, Middlesex UB8 3PH
United Kingdom
Tel: +44 (0) 1895 203397
Fax: +44 (0) 1895 251686

ACKNOWLEDGEMENTS

I would like to thank Dr. Stanislao Lauria and Dr. Syed Nasirin of the Department for Information Systems and Computing at Brunel University for their support and feedback throughout the development of this project; also, Dr. Howard Simmons of Palmer's College, Grays, for providing excellent reference information.

I am also grateful to the staff at Government Communication Headquarters, Cheltenham, and The Defence Academy, Salisbury Plain, for their assistance, and particularly to my interview subjects, who have requested not to be credited by name.

I would also like to thank Christine Connell for her patient proofreading and critical analysis of my work. Finally, I would like to express my gratitude to all my family for their unwavering support throughout my entire time at University.

I certify that the work presented in the dissertation is my own unless referenced

Signature.....

Date.....

TOTAL NUMBER OF WORDS: 9,973

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION	5
1.1 Background.....	5
1.2 Aims and Objectives	6
1.3 Research Methods.....	7
1.4 Dissertation Outline.....	7
 CHAPTER 2: LITERATURE REVIEW.....	 10
2.1 Introduction and History	10
2.2 Future Strategies	10
2.3 Technological Developments.....	11
2.4 The Complete Software Solution: The Three-Phase Model.....	12
2.5 Image Transmission and Image Processing	13
2.6 Image Rendering and Image Recognition	14
2.7 Current Methods of Edge Detection	15
2.8 Some Potential Problems	16
2.9 Safety Issues and Testing Standards	16
2.10 Considerations	17
2.11 Summary.....	17
 CHAPTER 3: RESEARCH METHODS	 19
3.1 Research.....	19
 CHAPTER 4: A REVIEW OF RELEVANT DEVELOPMENT METHODOLOGIES	 21
4.1 Introduction	21
4.2 Jackson's Systems Development (JSD) Methodology.....	22
4.3 Extreme Programming (EP).....	27
4.4 Conclusion	29
 CHAPTER 5: PRELIMINARY AND INFORMAL DESIGN SPECIFICATIONS	 30
5.1 Initial Designs.....	30
5.2 Functional Requirements and Problem Breakdown	31
 CHAPTER 6: FORMAL DESIGN SPECIFICATIONS	 33
6.1 JSD Sequence Diagram	33
6.2 Data Flow Diagrams	33
 CHAPTER 7: IMPLEMENTATION	 36
7.1 Introduction	36
7.2 Coding.....	36
7.3 Results of Implementation	40
 CHAPTER 8: TESTING.....	 41
8.1 Introduction	41
8.2 Risk Analysis.....	41
8.3 Test Plan.....	41
8.4 Positive Identification Testing.....	42
8.5 Negative Identification Testing.....	43
8.6 Test Results	45
8.7 Results Analysis.....	46
8.8 Threshold Analysis.....	46
8.9 Erroneous Input Testing.....	49
 CHAPTER 9: EVALUATION AND CONCLUSION	 50
9.1 Summary of the Dissertation.....	50
9.2 Research Contributions	51
9.3 Future Development	51
9.4 Comparison to Aims and Objectives.....	52
 BIBLIOGRAPHY AND REFERENCES	 53
 APPENDIX A: PRELIMINARY DESIGNS	 55
 APPENDIX B: ALTERNATIVE CODE IMPLEMENTATIONS	 56

CHAPTER 1: INTRODUCTION

1.1 Background

This project aims to develop an improved software system for image-recognition within the military application of bomb-disposal robotics. Specifically, the software that this project produces is designed to form part of a larger, theoretical expert system that could work alongside existing robotic technologies in order to make bomb-disposal less dangerous. This report documents the steps taken to produce an image-recognition function that uses colour-matching of pixels in an intelligent way to analyse the similarity of two given images and produce a percentage of similarity and make a discrete decision on the probability that the images portray the same object, regardless of factors such as distortion or different camera angles.

Development of new bomb-disposal technologies tends to concentrate on the robotic capabilities of the bomb disposal machine. The U.S. magazine "TECHBeat" proudly discusses advancements in the latest bomb-disposal robot: "It could lift 35 pounds vertically from a point 18 inches in front of its body, climb stairs at angles of up to 40 degrees, and operate at a 300-yard range ... it had the capability to fire a disrupter, deploy digital x-rays, and provide views from multiple color [sic] cameras."

Even now, current bomb disposal technologies rely almost solely on still-image and video streaming to allow the human operator to study the device. Hollingum (1999) stated that "Technology research sponsored by MoD covers known technologies or those believed to be important for meeting future teleoperation and Remote Land Vehicle (RLV) system requirements. Specific areas include telepresence and stereo vision; augmented reality; novel vision techniques; image compression; and image processing and optical flow."

If we believe this comment to be true, then it is evident that the role of the information supply, the object recognition and the human-computer interaction systems has been underdeveloped. Therefore, the object of this project is to develop an element of a decision support system for bomb disposal that is designed for implementation alongside robots used by the Police and the Armed Forces. It should be noted that this project shall not address the robotics element of the robot; rather, the object-recognition algorithms of such an improved system.

One of the key aims of this project is to make the bomb-disposal robot portray rudimentary intelligence. This is not to say that intelligence equals self-awareness; but intelligence must play an important part in the role of this robot. Pfeifer and Gómez (2005) note "True intelligence always requires the interaction with a real physical and social environment". Interaction with a physical environment will occur in the object recognition phase of the project, and interaction with the social environment in the theoretical GUI. Interestingly, the Compact Oxford English Dictionary defines intelligence as follows:

intelligence

• **noun** **1** the ability to acquire and apply knowledge and skills. **2** a person with this ability. **3** the gathering of information of military or political value. **4** information gathered in this way.

— ORIGIN Latin *intelligentia*, from *intelligere* 'understand'.

Surely definition 1 applies to most artificial computer systems capable of learning, this project being no exception. However, definition 2 punctures a hole in the idea of machines ever having 'intelligence'. Like many prominent academics and philosophers, I would disagree with the argument above that intelligence can only apply to a person. I plan to make this computer system intelligent by enabling it to fulfil the criterion expressed in definition 1, above; to make it able to **acquire** knowledge in the form of new information for the knowledge base, to **assess** the inputs using pre-defined procedures and to be able to **apply** that knowledge effectively in the form of outputs.

1.2 Aims and Objectives

AIMS

- 1) To discover common and uncommon problems faced by bomb disposal experts from the technological perspective, and from this research to design a solution that solves some or all of these problems.

- 2) To develop a software solution to be implemented in the field of bomb disposal that will address the problems as described above, that will out-perform current technology and that, specifically, will be more reliable, accurate and safe.

OBJECTIVES

- 1) To undertake research as defined in aim 1 by approaching experts in the field of bomb-disposal, requesting documents of procedures corresponding to bomb-disposal, and reading relevant studies and other literature pertaining to areas within the field of bomb-disposal, such as robotics, image recognition and audio analysis.
- 2) To develop software to deal with object recognition: this means to develop a system that will positively identify suspicious devices using a number of input criteria.

1.3 Research Methods

The research methods I have used are discussed fully in Chapter 3, however in brief, the methods I have used for research are semi-structured interviews with appropriate industry professionals and study of relevant academic literature. The insights I gained from the interviews and the information I gained from the literature both affected the way I developed my final software solution.

To develop a system to deal with object recognition: this means to develop a system that will positively identify suspicious devices using a number of input criteria.

1.4 Dissertation Outline

Following this chapter, the dissertation is organised as follows.

Chapter 2, "Literature Review", introduces some of the recent work surrounding the area of object recognition. In particular, a brief history of bomb-disposal is presented, and the future direction of the Armed Forces in this area is discussed. Recent technological achievements in robotics are examined. The chapter also examines factors affecting image quality and techniques of vector recognition. Some potential problems are identified, and this leads into a discussion of the stringent testing standards surrounding safety-critical systems, an area highly relevant to this project. Finally, a summary of the findings and relevance of the literature is given and some considerations for the development of my project are discussed.

Chapter 3 outlines the methods of research that were used to gather information prior to the design of my software. Methods were primarily semi-structured interviews with professionals in the military. The discoveries of my research and the effects upon the development of my project end this chapter.

Chapter 4 outlines two principle development methodologies that I shall be following during the design and development of my project; Jackson Systems Development Methodology and Extreme Programming. A detailed description of each methodology is given, the shortfalls and advantages of each methodology listed, and a summary of the direction of my design methodology discussed in conclusion.

Chapter 5 shows some of the preliminary design ideas and informal design specifications that marked the start of the design phase of my project. In keeping with the E.P. methodology and RAD methods, the design followed an iterative design-test sequence that gradually added greater functionality. The functional requirements are defined and, using the principle of critical path analysis (although without the chronological element), each task is broken down into sub-tasks, which are again broken down until an atomic list of tasks is created. This forms the basis of my formal design specification.

Chapter 6, "Formal Design Specifications", uses the principles from the JSD methodology alongside the notary principles of data flow diagrams to document the logical design of my system. This section mainly comprises of diagrams, and an overview is given at the end.

Chapter 7 shows how the design was implemented into a working software system. The code produced is demonstrated and specifics of implementation are discussed.

Chapter 8 deals with the testing of the implemented system. As the software is safety-critical, testing was thorough, following a designated test plan. All the test data is shown and discussed, and appropriate graphical and mathematical analysis of the results follows. As a result of the testing, a proposed improvement to the system is presented, which was introduced to the software in the final stages of implementation. Types of testing are discussed and the results summarised.

Chapter 9 is a critical evaluation of my project; how well it met the aims and objectives defined above, and the areas in which it failed. It looks at the problems faced during development and how they were overcome. The capabilities and limitations of the product are also discussed. This chapter concludes the report, and gives a summary of the products of the project and how the products contribute to the appropriate research area. The main findings are given in a chapter-by-chapter format, and some ideas are given for how the project could be expanded upon and improved to incorporate new functionality.

The following chapter is a critical review of the literature surrounding the subject area and comments upon its relevance to this project.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction and History

The safe disposal of mines, bombs and other explosive devices is a key component of the duties of both the Armed Forces and of civilian defence personnel in every nation of the world. Jefferson (1997) claims that, "In the seven countries in which the HALO Trust works, the official UN figure for the number of mines is 40 million." There are hundreds of de-mining organisations working on minesweeping throughout Europe, Southern America, the Russian Federation and the Middle East. Terrorist bombing has become a way of life for many not just in war-torn countries such as Israel and Palestine, but sadly for the Western world as well.

Since 1939, and more consistently since 1972, most major cities in the UK have been targets for the provisional and Real IRA groups, using mainly home-made explosives (BBC News, 2001). The Madrid train bombings, the July 7 London bombings and the multitude of bomb scares in recent months in Birmingham, Coventry and Manchester have all been evidence of the increasing terrorist bomb threat that we now face.

The history of the institution of bomb disposal experts can be traced back to the early World War I era. Both the British and the Americans realised that there was a need to locate and safely destroy unexploded shells, of which there were many, before they could become a danger to civilians. Thus the British allocated a section of the Royal Engineers to become experts in the disposal of explosives, while the Americans established the USN Mine Disposal School at the Naval Gun Factory, DC, in 1941 (Answers.com cites Wikipedia.org, updated Dec. 1999).

2.2 Future Strategies

The British Army in particular is keen to recognise the growing threat of bombs, both on the battlefield and in civil life. The UK Defence Secretary Geoff Hoon has recently outlined a restructuring of the British Army, creating a plan he has labelled the "Future Army Structure". In speaking of the development of new battalions, the report states "The FAS [Future Army Structure] has enabled plans to be put in place

to create new capabilities including ... a number of new sub-units including surveillance and target acquisition, bomb disposal and vehicle maintenance" (Military Technology Magazine, 2005).

Meanwhile, on the civilian side, since the July 7 attacks the Metropolitan Police has asked for funding for a further 1,500 anti-terrorist officers. Sir Ian Blair, the Metropolitan Police Commissioner, has been quoted as saying, "Intelligence exists to suggest that other groups will attempt to attack Britain in the coming months" (WebIndia123.com, 2005). It is clear that British military forces operating overseas, civilian defence troops and the Police force are all attempting to combat the growing threat of terrorist acts.

2.3 Technological Developments

So how has technology kept up with the growing demand for better protection from the terrorist threat? The principle techniques of bomb disposal, and the intricate details of the technology involved, are issues that are clouded in secrecy, for reasons of security, but some pioneering work, especially in the integration of robotics and information gathering, has been prevalent in current methods and details are reasonably accessible.

Recent developments in robotics have led to astounding advances in the quality of autonomous- and human-controlled- robots, and one such example is documented extensively in Neuhaus and Kazerooni (2001), with a human-assisted walking robot capable of carrying heavier loads than a human is capable of, that is capable of bipedal walking that is constantly adjusted to the speed and bearing of the human helper, and capable of navigating rough terrain. If this robot were to be equipped with sensors, it would be an excellent tool for use in the bomb disposal industry.

The Lawrence Livermore National Laboratory, a subsidiary of the University of California, CA and funded by the US Department of Energy, has developed the S.T.A.R. (Spiral Track Autonomous Robot), capable of autonomous movement using basic artificial intelligence to calculate a path from a given start point to a given end point, and is equipped with wireless video capability to relay a real-time stream of the surrounding environment to the operator.

These are but two examples of the current technology employed by the Armed Forces and Police for disposing of bombs. However, all the technological accomplishments of recent years have focussed on developing the robotics and automation components of such devices. There has been little, if any, development of the sensory capabilities and autonomous information gathering and processing that these machines have such potential for. Here, then, is the motivation and problem definition of this project; to develop such a software system to gather information, identify target devices, cross-reference that information and present that information to a human user in a remote location. This software solution to the problem will be able to be implemented onto any suitable autonomous robot, potentially making it a valuable tool to bomb disposal personnel.

2.4 The Complete Software Solution: The Three-Phase Model

The complete software solution could be abstracted into three distinct phases; the information gathering, the information processing and the information presentation. The scope of the project means that the information processing is of primary interest, as image gathering will be a hardware consideration, and information presentation is of little importance to the system being developed. The areas of information gathering and processing that are relevant are now discussed.

The information gathering stage will consist largely of gathering and rendering inputs from sensors attached to the robot. Whilst the gathering will be done by sensors, and is not part of the scope of this project, the rendering is of primary importance. As the sensory inputs to be rendered are going to be (largely) still images, the image rendering and recognition must be excellent. Graham and Barrett (1997) have a useful definition: "Image processing may be regarded as the application of a particular process to an image in order to obtain another image". Taking this as the target task of this phase, then, the key aim is to develop the process – the rendering – of the image. However, the target of this chapter of the project is not to design the rendering system, but to examine the history and development of the area since the birth of image processing, and also to study similar methods in the field such as error-reduction in stream transmission and methods of edge-recognition within images.

2.5 Image Transmission and Image Processing

According to Graham & Barrett, an early example of image transmission comes from the broadcast of digitised newspaper pictures sent by submarine cable in the 1920s. Since then, the development of broadcasting facilities, especially across wide geographic areas, has become more advanced; mainly driven by the necessity of scattered communities to have access to transmissions. An excellent example of this comes from Bridle (1988), who examined the development of satellite transmission facilities in Australia. Bridle explains how Australian population distribution is concentrated mainly around the coastal areas, and how facilities for those people spread throughout the Northern and Western Territories was minimal until approximately 1977. The development and recent availability of broadcast facilities to the outermost areas is now significantly advanced, with satellite transmissions being available to most inhabitants of the island.

Image transmission shares much in common with image processing, a principle capability of this project. Looking now at more recent examples, such as the rise of digital television in the West, Anastassiou (1993) lauds the advantages of digital transmission in television over traditional analogue transmission: "Under appropriate channel transmission conditions, combined with the help of error correction techniques, the zeros and ones representing the original signal remain intact, giving identical signal quality at the receiver". Reimers (1998) comments extensively on the replacement of analogue signals with digital, citing the low error rates. However, of particular interest are Reimer's comments on the accuracy of error control within image processing; "[digital television systems] ... transforms the traditional TV channel to a data transmission medium which may carry huge data rates at extremely low bit error rates (below $1 \cdot 10^{-11}$).". This is phenomenally low, meaning effectively that just 1 in 100,000,000,000 bits are inaccurate. This figure clearly represents the pinnacle of image processing, at least in the context of error handling, in the field.

referenced at the end of this chapter, lists some of the applications of the noteworthy Keyence CV-700 camera, which is capable of "checking the dimensions of manufactured parts, checking robotic handling and even measuring the pitch between connector leads".

2.7 Current Methods of Edge Detection

Siddique and Barner (1998) have some interesting observations on the nature of current edge detection methods, and the problems inherent in detection of edges within images. "Most edge detection methods operate on an image at a single resolution and output a binary edge map". It should be noted that this is the intention of the first phase of this project; to output a two-colour binary edge map of a picture in a single given resolution. Furthermore, Siddique and Barner go on to state that, "Edges within a image, however, generally occur at various resolutions, or scales, and represent transitions of different degrees, or gradient levels. Thus, single resolution edge-detection methods that output binary edge maps do not always yield satisfactory results." This is a major concern of the project here; that the results will not be of sufficient quality for reliable use.

The research of Siddique and Barner is highly relevant and interesting to this project. Their paper deals with the development of a different technique of edge-recognition, a wavelet-based algorithm designed to perform at multiple resolutions. The algorithm produces edge maps, stacked in a pyramid, with all edges on all scales and levels on an edge map at the bottom of the pyramid and large, coarser edges in a map at the top. This pyramid structure also reflects intensity changes and is non-binary, in order to hold more information.

This paper then continues on to discuss a classical edge-detection technique, using thresholds on stages throughout the image to determine whether an edge exists, then continuing to outline another method using intersects. The bulk of the paper, however, consists of an account of the mathematical theory, implementation and simulation of the developed algorithm, which is somewhat complex.

It is important to note that, while the field of edge-detection has been comprehensively explored, by no means is the use of edge-detection techniques

redundant in the context of the research question. Rather, it serves as an integral component of the aim; to develop an expert knowledge-based system for implementation in robots.

2.8 Some Potential Problems

It is important to identify the problems in the analysis stage of the project. One issue that will be a setback in the eventual implementation of the system is the sheer range of target devices that the sensory equipment will encounter. According to a paper published by the Center for Domestic Preparedness of the USA, explosives come in three main types: pyrotechnics, which tend to be visually captivating, low-yield with little damage; examples include fireworks and road flares. Propellants include black powder, solid and liquid rocket fuels. High explosives include solids such as TNT, C4 and dynamite.

Theoretically, if high-grade sensory equipment such as tactile samplers, x-ray facilities and chemical analysis tools were fitted to the robot, these distinctions would become important. However, due to the range of targets, it is clear that this device, armed with only a video camera and digital camera, would realistically be more suited to landmine detection, where devices will look very similar. This presents a major problem as it contradicts the very aim of the project. However, development of such a system could eventually be applied to bomb robots assuming the knowledge base is sufficient and the sensory equipment available.

2.9 Safety Issues and Testing Standards

When looking at software development in situations involving systems that lives can depend upon, it is expected that the test criteria for such a system are going to be rigorous. It is a requirement, for example, that aviation software is tested using a control-flow testing method called MCDC (Modified Condition / Decision Coverage), as its fault-detection effectiveness is high. The DO-178B standard requires this by law, according to research by Kapoor & Bowen (2005).

It is reasonable to question, then, whether the new system would, realistically and before implementation, have to undergo standardised formal testing procedures to

certify the safety and reliability. For the purpose of the project, this testing standard will not be adhered to as the system will not realistically be implemented; in addition, it is outside the scope and time restraints of this project. However, a formal test plan will be drawn up that aims to test all aspects of the system, from a usability point of view to unit, module and system testing, using established techniques. For the purposes of this project, reliability and accuracy will be anchors in the testing strategy, with development of both a priority.

2.10 Considerations

When considering this project, there are a number of issues not already discussed that need examination. The important question of whether extra information about suspected devices will realistically help, or affect methods and decisions surrounding disposal, is an important one.

To answer this question, one can look at rates of failure in information systems project implementation; a subject on which there is plenty of literature. Frese and Sauter (2003) cite Lewis (2003) in claiming that 70% of I.T. projects fail to meet their objectives. Yet it is not so much the fear of project failure, rather the fear that the human operators will simply use the existing video and audio technologies, leaving the new system to run redundantly alongside. Therefore, the wants, needs and considerations of the users must be of prime importance during development.

2.11 Summary

Bomb disposal technology has come a long way since the first training school for Royal Engineers in the 1920s. With the advent of computerisation, robotics and artificial intelligence, human operators are more often than not operating remote robots in order to achieve their goals. The principle aim of my project is to develop a software solution to the problem that there is no knowledge base other than the experience and training of the human team to guide decision-making in such a critical situation. I have outlined and reviewed some of the key methods and research surrounding the topics of image broadcast, image recognition, edge-detection,

rendering, interface design and concerns, safety and reliability, testing standards and potential problems.

This review has assisted me in developing a plan for development, and helped to decide which methods may work and which methods will be beyond the capacity of this project. Furthermore, it will help me to avoid some of the potential pitfalls of development and installation, to choose an original direction for my project, and assist in the cycle of re-iterative development by acting as supporting research with which to make key decisions.

The next chapter discusses the research methods used to gather information before the design of the software proceeds.

CHAPTER 3: RESEARCH METHODS

3.1 Research

To establish the user requirements and the functional requirements, I have undertaken research in two main directions.

1) Gather information from potential end-users and users of similar, implemented systems that are in use in the field already.

AIMS:

- To discover the shortfalls of the systems currently used
- To find out about end-user attitudes to more involvement of computer technology in such a safety-critical area, in order to establish the feasibility of any potential new software system.
- To find out about other relevant work that others have undertaken that may have some bearing on this project.

METHODS:

- Library and online research into journals covering object recognition and robotic developments.
- Semi-structured interviews with qualified professionals in the field.

RESULTS:

- I was able to obtain interviews with two professionals, one, a Lieutenant in the British Army based at GCHQ, Cheltenham, an Armed Forces research and development base. The second was with a technician investigating biometric identification methods at the Defence Academy, Salisbury Plain. The transcripts of the interviews are contained in the Appendices, however the conclusions drawn from these interviews are as follows:

CONCLUSIONS:

Interview 1: GCHQ, Cheltenham

Summary: The subject was interested in the research I was doing and the possible military applications it would have. He questioned the importance of the specifics of my work, stating that there were other more advanced solutions available in the commercial environment such as the research and development in biometric identification. However, he stated that improved technological solutions were always welcome, as the Armed Forces is increasingly reliant on technology. The subject expressed no concern over the greater use of technology in this area, but stated that public opinion may differ. The subject was not able to suggest any academic sources for this topic.

Interview 2: The Defence Academy, Salisbury

Summary: The subject was a specialist in object recognition and surrounding fields, and took a great deal of interest in the pixel-comparison method that I was suggesting. He suggested many other directions the project could head in, including background elimination, mathematical methods to sharpen images and lower the effect of distortion, and vertex recognition. He was not concerned over increased use of technology in military applications, stating that in his opinion, technology is often more reliable than human judgement. His opinion of the shortfalls of the current systems compared favourably to mine; he agreed that human judgement still takes precedent over software judgement, and that object recognition is still not used effectively in military applications, especially considering the boom in research and development in biometrics. The subject suggested several journal articles to look at, which are used and referenced in Chapter 2.

- 2) To research academic literature surrounding the areas of object recognition, edge-recognition, biometric developments and image manipulation.

RESULTS:

The results of my review of the academic literature are contained in Chapter 2: Literature Review. The following chapter discusses development methodologies.

CHAPTER 4: A REVIEW OF RELEVANT DEVELOPMENT METHODOLOGIES

4.1 Introduction

As is well known, there are several main components in the system life-cycle, the fundamental development strategy that is associated with systems proposal, analysis, development and implementation. It follows, therefore, that different categories of development methodology will favour different aspects of the SLC, and the type of methodology chosen will reflect the budgetary boundaries, timeframe, user requirements, and a multitude of other factors. When choosing a methodology to best suit the environment surrounding the systems development of my project, I must bear in mind the following factors:

- * The qualitative nature and imprecision of my user requirements.
- * A limited timescale to complete development.
- * The software to be developed is not a model; rather a function, or a solution, to a problem within a proposed software solution.
- * Testing is imperative, as the software to be developed is safety-critical.
- * User feedback is relatively unimportant; functionality will take precedence over usability.

Hence, these criteria limit the type of systems development methodology that will be suitable. Many methodologies are requirements- and analysis- based, concentrating on the first half of the SLC; that is, feasibility, analysis of user requirements and other pre-implementation stages. In order to develop a system that concentrates mostly on the design, testing, implementation and evaluation of the system, I shall need to select a methodology that concentrates on the processes within the system, rather than other methodological types (such as Ethics-based, organisational-based and file-system-based). After reviewing a number of methodologies within the process-based methodologies available, I have limited my selection to the following two.

4.2 Jackson's Systems Development (JSD) Methodology

Michael Jackson proposed a process-oriented methodology that considers system design and program design to be of the same stock. The programming-oriented approach of this methodology is ideal for my purposes, as my user requirements are quite unambiguous and uncomplicated, whilst my priority must be the development of my pre-defined function. JSD tends to be more suited to software development than to organisational factors. Also, JSD places more emphasis on the relevance of real-time within systems, dealing with it in a dynamic as opposed to a static fashion.

JSD has three main phases. These are:

Modelling Phase

- * Entity Action
- * Entity Structure

Network Phase

- * Initial Model
- * Function
- * System Timing

Implementation Phase

- * Physical System Specification

During the modelling phase, a modelling language such as UML (Unified Modelling Language) may be employed in order for the developers to ascertain the real-world influences upon the system, modelling requirements of the real world as applied to the development, and the entities and associated actions concerned with the system. This phase "sets up" the environmental requirements, the user requirements, the factors that influence the system or that the system is to model, and charts them in order to form a draft model of the intended implementation. The actions associated with each entity and the boundaries each entity has will be defined. Furthermore, the actions associated with each entity are chronologically ordered and can be dependent upon each other.

There are three requirements in JSD for the successful establishment of an entity.

- * An object, to be defined as an entity, must perform or be affected by actions in a chronological order.
- * The object to be modelled as an entity must exist in the real world, independently from the system that is modelling it.
- * It must be possible to create an instance of the object, and that instance must be able to be uniquely identified.

As with most other methodologies, entities that exist in the real world that do not have a relevant or existent influence upon the system to be modelled can be safely ignored. JSD deals only with the relevant entities that have influence upon or are influenced by entities within the system.

The modelling phase deals with the interconnectivity of the entities, and their relationships with each other. In this aspect, it is not so different from the principles underlying object-oriented systems development. However, JSD can easily be applied to a range of different projects utilising different styles of development, whereas OOSD deals strictly with hierarchal, abstracted and modularised structures.

As noted, a requirement of an entity is that it must perform or be affected by actions in a chronological order. Therefore it is necessary to define an action. JSD has requirements for what constitutes an action:

- * An action must take place at an atomic point in time, as opposed to taking place over a variable length of time.
- * It is not possible to modularise, or abstract, an action; it performs a purpose that cannot be sub-divided into other actions.
 - An action must take place in the real world and not simply be an action within the system itself.

The result of the first step of the modelling phase should be a list of entities, their

attributes and the actions that can be performed by, or upon, them. This stage of JSD eliminates functional entities and actions, and simply emulates real-world information flow; an important step in systems analysis.

The second step of the modelling phase is the entity structure step. Entity relationships and structures can be shown as tree diagrams, known formally as JSD structure diagrams. Each SD is designed to cover the complete lifetime of an entity, and will therefore contain the action that causes the entity to exist, and the action that causes the entity to cease to exist. The SD must show the chronological structure of the actions; and must show that concurrency between actions cannot occur. An example of a JSD structure diagram is given below, in Figure 1:

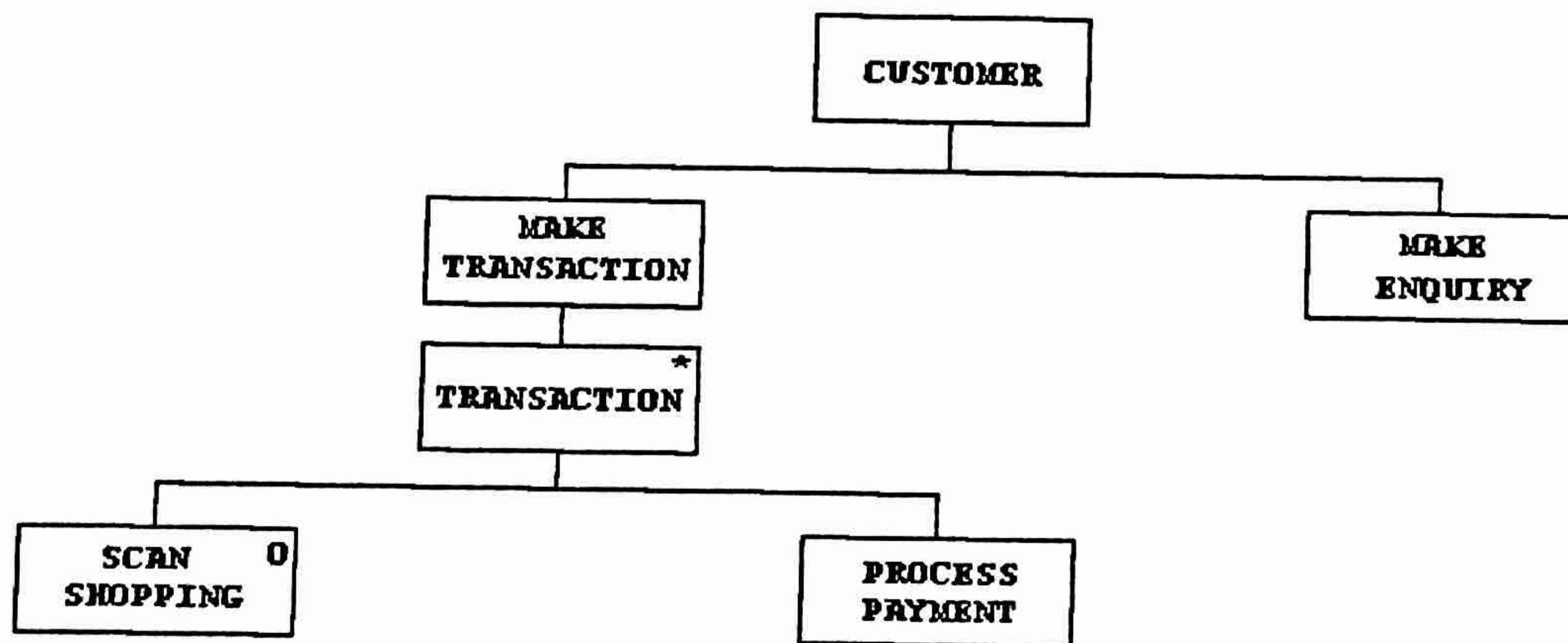


Figure 4.1: JSD structure diagram. "*" denotes an iteration, while "O" shows the choices under the iteration.

JSD also requires that if an entity is to have more than one role, each role must be documented separately. For example, if the entity "TRANSACTION" fulfils two roles, one containing actions such as "SCAN SHOPPING" and "PROCESS PAYMENT" and another role containing the actions "COUNT MONEY" and "PRODUCE REPORT", the entity TRANSACTION would have two S.D.s; one showing the actions and sub-entities associated with performing a transaction and one showing the actions and sub-entities associated with cashing the till at the end of the day. The result of the second step of the modelling phase should be a set of structure diagrams.

The second phase is the network phase. The first step of the network phase is the initial model step. In this step, a model is made of the real world. The entities are connected by sequences, or processes, that simulate activities that would happen in a use-case. It should be noted that in this stage, time becomes an influence and if an instance of a sequence takes 10 years to complete, the entities and actions within the model associated with the sequence will also be modelled as taking 10 years to complete.

The definitions of the sequences and the entities and actions they contain are documented both as JSD structure text, which is a hierarchal pseudocode description, and as accompanying diagrams.

There are two ways of connecting processes in JSD; these are data stream connection (DSC) and state vector connection (SVC). In a similar fashion to process interaction in data flow diagrams, DSC concerns one process outputting a sequence of messages to another process, which reads this stream. The synchronisation of communication using DSC in JSD is closely monitored. If, for example, there are multiple inputs to a process, set rules must determine the order in which the inputs are to be taken.

This merging of inputs is subject to three types of rule; fixed merge, where the rules are unchanging, data merge, where the data stream determines the precedence of communication (much like, to take an example from the real world, using first-class rather than second-class stamps on envelopes), and rough merge, determined by the availability of messages (a FIFO-style arrangement). In comparison, SVC is dependent on the speeds of return of the processes involved. An interesting aspect of JSD is that time-grain markers are allowed in chronological sequences or sets of sequences; they indicate the arrival of a real-world chronological point. This function is missing in most other methodologies, where time is not generally considered to be an influencing factor at all.

A state vector is an internal variable, which is defined and is particular to a specific process. In state vector connection, one process inspects the state vector of the second process. The first process can therefore inspect the second process at either specific chronological points, or when prompted by another process or internal

command to do so. It varies, therefore, from data stream connection, which can be an unprovoked communication at irregular times.

A system specification diagram (SSD) in JSD models the system as a network of interconnected processes showing the communication between these processes. Processes are shown in boxes, with the data streams that connect processes shown in circles with unique identification. Arrows show the direction of the data stream. State vectors are shown as diamonds. The result of the initial model step of the network phase is an SSD showing a set of communication processes, each specified by structure text and associated diagrams. This can be merged into the structure diagrams to give a complete model. My chosen methodology will incorporate this form of diagram to try and clearly demonstrate the data flow, or communication, between the program components.

The function step in the networking phase concerns the creation of functions to produce an output when certain combinations of sequences occur. These are added to existing diagrams and structure text to more accurately emulate the real world. A function will receive an input, and in response to the contents of that input, output a result. For example, a function named PRINT may activate when a message is sent to it. PRINT will read the data stream message, which will contain the information to be printed and the command to print, and the output will be to print the information sent to it. The end result of the function step is an updated SSD and associated structure text containing the functions associated with the system.

The system timing step, the final step in the network phase, deals with the real-life timing of the system. Consideration is given to the delay between the receipt of inputs and the production of outputs of a process. Time-grain markers, sent as data streams, contain timing information that helps to synchronise processes and actions within processes by triggering or delaying these actions. Time constraints are often specified by user requirements, such as the need for an annual report or the need for an immediate response to an input. The end result of this phase is a list of timing constraints and the associated time-grain markers paired with the appropriate processes. Processes designed specifically to ensure certain actions have been completed before further processes can be initiated, known as synchronisation processes, can be added to the SSD at this stage.

Finally, the implementation step of JSD deals with sharing processors amongst processes. The reason for this concentration is clear; a single processor deals with a single command at one time. It is therefore impossible to run processes truly concurrently. Jackson's methodology concerns the distribution of processors to processes, in order to maintain the time ordering inherent throughout. This phase is not relevant to my project, therefore I will not be discussing it; it is enough to know that the end result of this phase is a system implementation diagram showing the heirarchal structure of the scheduling processes and their relationships with external processes and data storage.

To conclude and summarise; JSD does not deal with all aspects of the development of the system; it does not concern itself with determining user requirements, testing or evaluation but only with the logical structuring of the processes. It places much emphasis on timing and synchronisation of message passing and activation of processes. As the function I shall be developing is not particularly time-critical and will be, in the main, a linear sequence of activities with minimal user interference, these reasons may determine that JSD is unsuitable for my purposes. I shall now examine Extreme Programming, which supports rapid application development (RAD) and is more of a set of principles of system development than JSD, which is a step-by step methodology.

4.3 Extreme Programming (EP)

As stated, EP is a set of principles rather than a step-by-step methodology, however there are still some ordered steps involved. The main output of the methodology is to produce software, and EP does not concern itself with the delivery of documentation to support the system. This is unacceptable in relation to my project, which is why I will be merging JSD for design documentation yet using principles of EP to ensure rapid development without getting overly concerned in design considerations.

The ordered steps in EP are:

Planning

Planning takes a role in the project, with the priority of the functions, human factors

(such as the team involved and their constraints), cost estimation and planning a schedule. Already, we can see that EP is much more "real-life" than JSD, and takes an interest in the external factors that influence the working of the methodology.

Design

Design in EP is an incremental, iterative process. Rather than attempting to tackle a problem as a whole, the problem is abstracted into many sub-problems, which are in turn abstracted, then distributed to the programming team according to the schedule produced in the planning stage. The emphasis in this stage is to ensure quick delivery of small segments of code. This enables the programming team to review each deliverable on a regular, routine basis, and discard any deliverables that are unsuitable.

EP could, therefore, be compared to a bottom-up development approach, where modules are produced, tested and combined to form larger modules, which are in turn tested and combined to produce the final system. It follows, therefore, that EP carries with it an element of risk; the larger a module becomes, the more difficult and costly it is to change the elements within it if the module proves unsuitable for purpose. Designing often overlaps development in EP due to it being a RAD-based methodology.

Development

EP describes the development phase well. A typical development cycle will involve a programmer, or more often a small team of programmers, developing a piece of code to perform a function. This function will be tested according to set test guidelines, specified in the planning phase. Should the code fail the test, extra code is added or the code is changed in order to make the function pass testing. Only then are the functions combined to create larger functions. This highly iterative approach means small problems that could potentially cripple a system designed using top-down development are solved in the very early stages of development. It follows that testing is merged with this phase and users are normally consulted constantly throughout, providing valuable feedback. Unfortunately, it can be difficult to set a schedule using EP development, as one can never tell what problems will occur. Whereas a top-down development methodology can use stubs to simulate the

function of unwritten modules, a bottom-up methodology has greater difficulty simulating the output of individual functions to the requirements demanded by the testing methodology.

Production

The production phase concerns itself mainly with system testing and rollout of software to users - beta testing. This phase tests the fitness for purpose of the entire system, and normally involves iterative maintenance and debugging, user-specified changes and evaluation of the system.

4.4 Conclusion

In reviewing both of the above methodologies, I can identify features from each that I can use to create my own custom-designed development methodology. The "fast-track" principles and bottom-up approach favoured by EP is attractive, as it complements some of the factors I specified at the beginning of this section. It favours a limited timescale; it is also good for developing the system as there will not be a great deal of interconnectivity between specific modules; rather, I plan to implement the function as one callable method, taking user-set parameters, that incorporates a number of private methods, each of which perform not more than one function. This is good design and using EP will be appropriate, as a bottom-up approach can be taken. EP also looks at more elements of the SLC than JSD, which concentrates on the analysis and documentation stages of the design.

However, JSD has its advantages, as the strict protocols specified for documentation will make the final design of my system very clear. Using this methodology will enable logical problems to become more apparent; EP only allows for discovery of problems upon coding. JSD also concentrates on the functionality of the system and does not concern itself with usability requirements (which are important in EP).

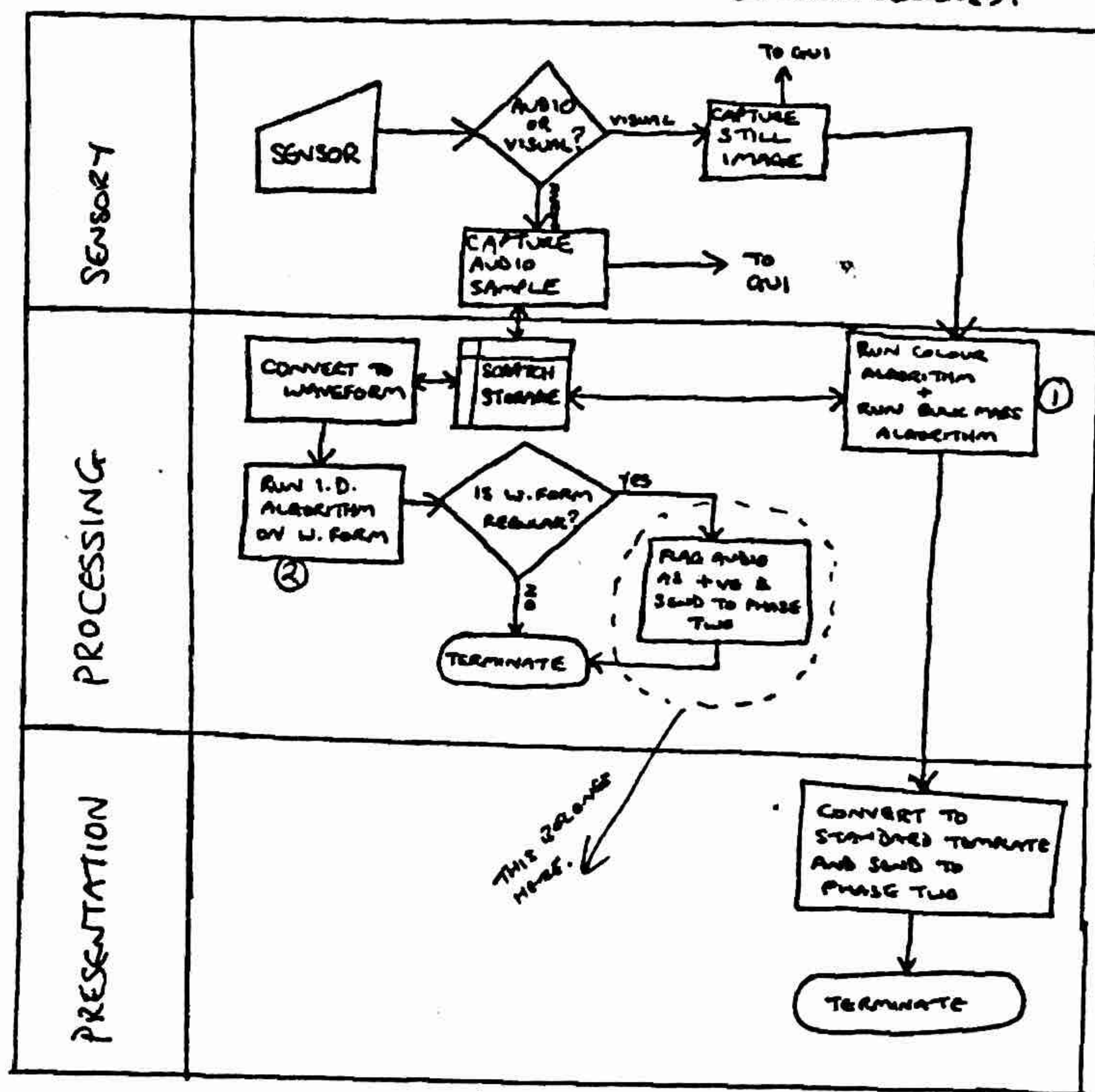
I have therefore decided to combine the Jackson Systems Development methodology and the Extreme Programming methodology to take advantage of the elements of each methodology that are appropriate to my planned software. The following chapter discusses the preliminary design specifications, prior to formal documentation.

CHAPTER 5: PRELIMINARY AND INFORMAL DESIGN SPECIFICATIONS

5.1 Initial Designs

My initial designs were paper-based and produced on a loose basis; ideas produced here were discarded or changed significantly to produce the final design. My initial diagrams also, to an extent, showed the design for the data comparison stage and the presentation stage of my project.

PHASE ONE: OBJECT RECOGNITION (PART ONE: BULK MASS RECOGNITION)
+ AUDIO RECOGNITION
(DOES NOT INCLUDE VIDEO/X-RAY/ANY OTHER NON-AUDIO OR VISUAL SENSORS).



NOTES:

- THIS COMPLETE PROCESS TO BE RUN FOR EACH SENSOR.
- ONLY TWO GIVEN ABOVE: CAN BE EXPANDED TO INCLUDE E.G. X-RAY.
- THERE ARE TWO MAIN ALGORITHMS TO DEVELOP, MARKED '1' AND '2'.
- ONCE INFORMATION HAS BEEN SENT TO STORAGE FOR ACCESS TO PHASE 2 (INFORMATION GATHERING + D.BASE), THE SCRATCH STORAGE CAN BE WIPED.

INPUTS: SENSOR ARRAY
 OUTPUTS: STILL IMAGE → GUI (PHASE 3)
 AUDIO SAMPLE → GUI (PHASE 3)
 PICTURE INFORMATION → PHASE 2
 AUDIO INFORMATION → PHASE 2

Figure 5.1: Preliminary Object Recognition Flowchart

As this project focuses only on object recognition rather than database design and GUI development, the two diagrams depicting flowcharts for these stages are contained in the Appendices for reference.

My initial design catered for the facility to recognise audio also; I decided later in the project that this is beyond the scope and therefore abandoned this idea.

From my diagrams, and by examining the user and functional requirements, I developed a list of tasks that my function should be capable of performing by abstracting the basic requirements into sub-tasks. I have noted the obvious problems I will have with each task and proposed a solution:

5.2 Functional Requirements and Problem Breakdown

BASIC FUNCTIONAL REQUIREMENT:

- (1) The function should be able to take two images, compare them and output a measure of similarity.

LAYER 1 ABSTRACTION:

- (1.1) The function should be able to take two images as its parameters.
- (1.2) The function must compare the pixels of the two images and decide whether they are similar.
- (1.3) The function must calculate how many pixels in the images qualified as similar and output a suitable response.

LAYER 2 ABSTRACTION:

Problem:

- (1.1) Images are difficult to manipulate using traditional programming languages as they are presented in a format that cannot easily be numerically manipulated.

Solution:

- (1.1.1) The function will use the Python Imaging Library (PIL) within Python to load the images into the buffers, and provide methods that can render images.

(1.1.2) The function will use the List.GetData method within Python to render each pixel into a numerical value and store in a list, or array. Two arrays will then be produced, each containing the information corresponding to an image.

Problem:

(1.2) Assuming a colour image will be in RGB format with 256 possible shades of each colour band, each pixel will be represented as a triple with the value (x, x, x) where each x can be in the range 0 to 255 inclusive. With this type of image, there are over 16 million different colour combinations. Colour matching is therefore unlikely to be successful, even allowing for an error margin, due to the large range.

Solution:

(1.2.1) Each image passed to the function as a parameter should first be converted into a 256-colour bitmap, meaning there are only 256 possible colours each pixel could be. This will mean each image is represented as a single numerical value from 0 to 255 inclusive, in one band only.

(1.2.2) To do this, I will use the Image.Convert method in ImageFilter, part of the PIL. This can convert most types of image into a number of set formats. It can also work with 256-shade greyscale images using this method.

Problem:

(1.3) No two images are going to be absolutely identical. This is due to a range of factors including picture angle, distortion, interference, different light sources, human error, size differences, image quality and others. How to introduce a solution where these factors are largely ignored?

Solution:

(1.3.1) Ensure the sizes of the images match. To do this, call the Image.Size method in the PIL, and render the larger image to the size of the smaller one.

(1.3.2) Ensure the formats of the images match. To do this, as mentioned, make sure the formats of the images match using the Image.Convert method.

(1.3.3) Introduce an error margin on the values of the pixels, set either by the developer or as a user-set variable. Test to see which value of error-margin is most effective.

The next chapter displays the formal design documentation.

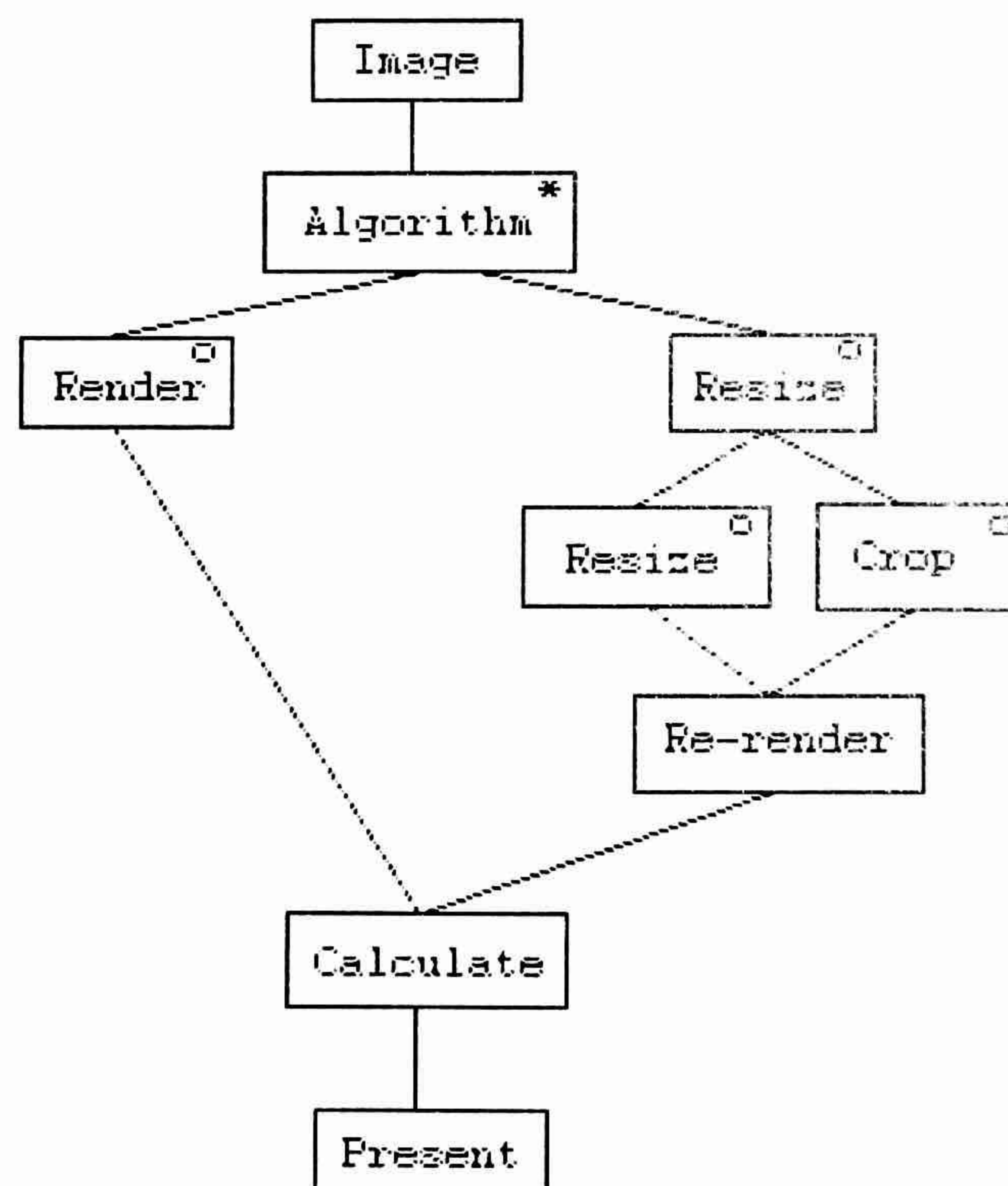
CHAPTER 6: FORMAL DESIGN SPECIFICATIONS

This chapter will document the preliminary design in a more formal context, using a JSD sequence diagram and a series of layered data flow diagrams (DFDs).

6.1 JSD Sequence Diagram

A JSD sequence diagram follows to show the entities and actions within the system.

Figure 6.1: JSD SD with entities and actions



6.2 Data Flow Diagrams

Finally, a series of data flow diagrams between components completes the formal design phase.

Figure 6.2: System Context Diagram

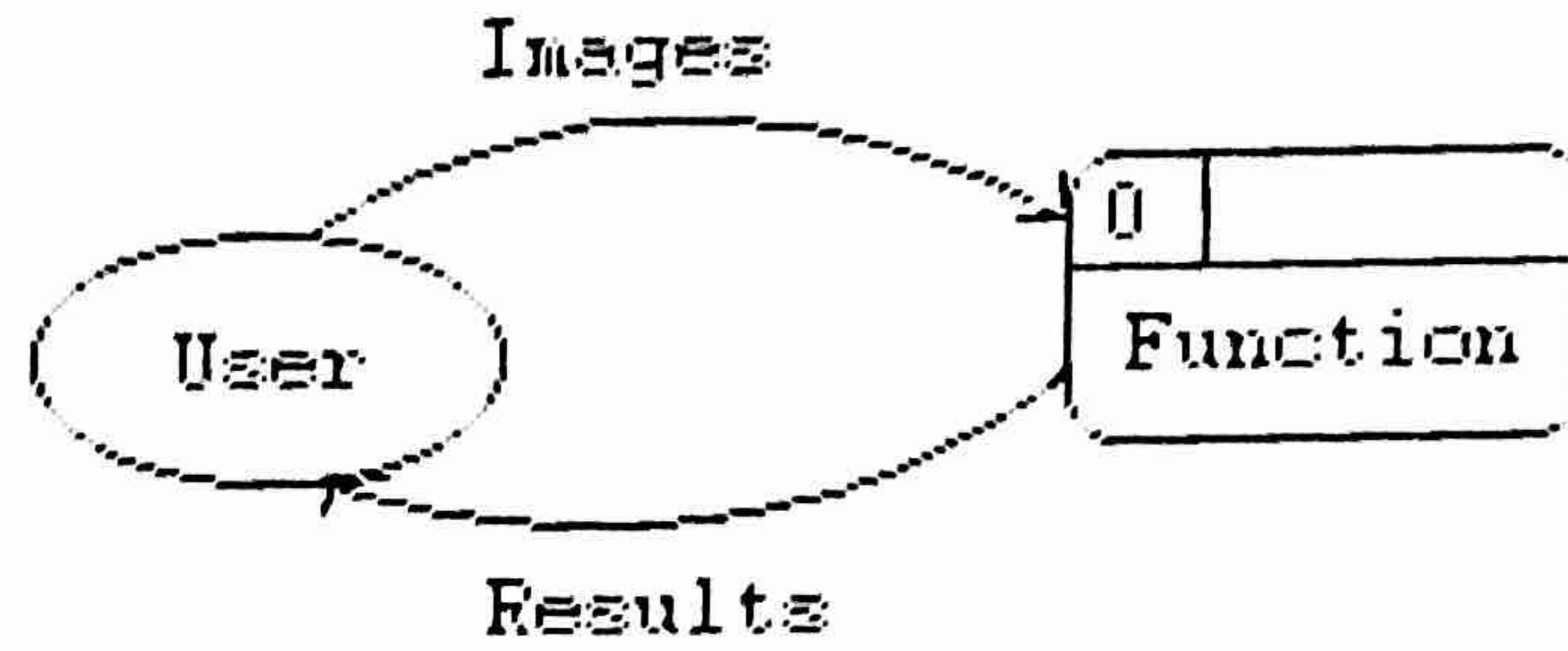


Figure 6.3: Top Level DFD (Layer 1)

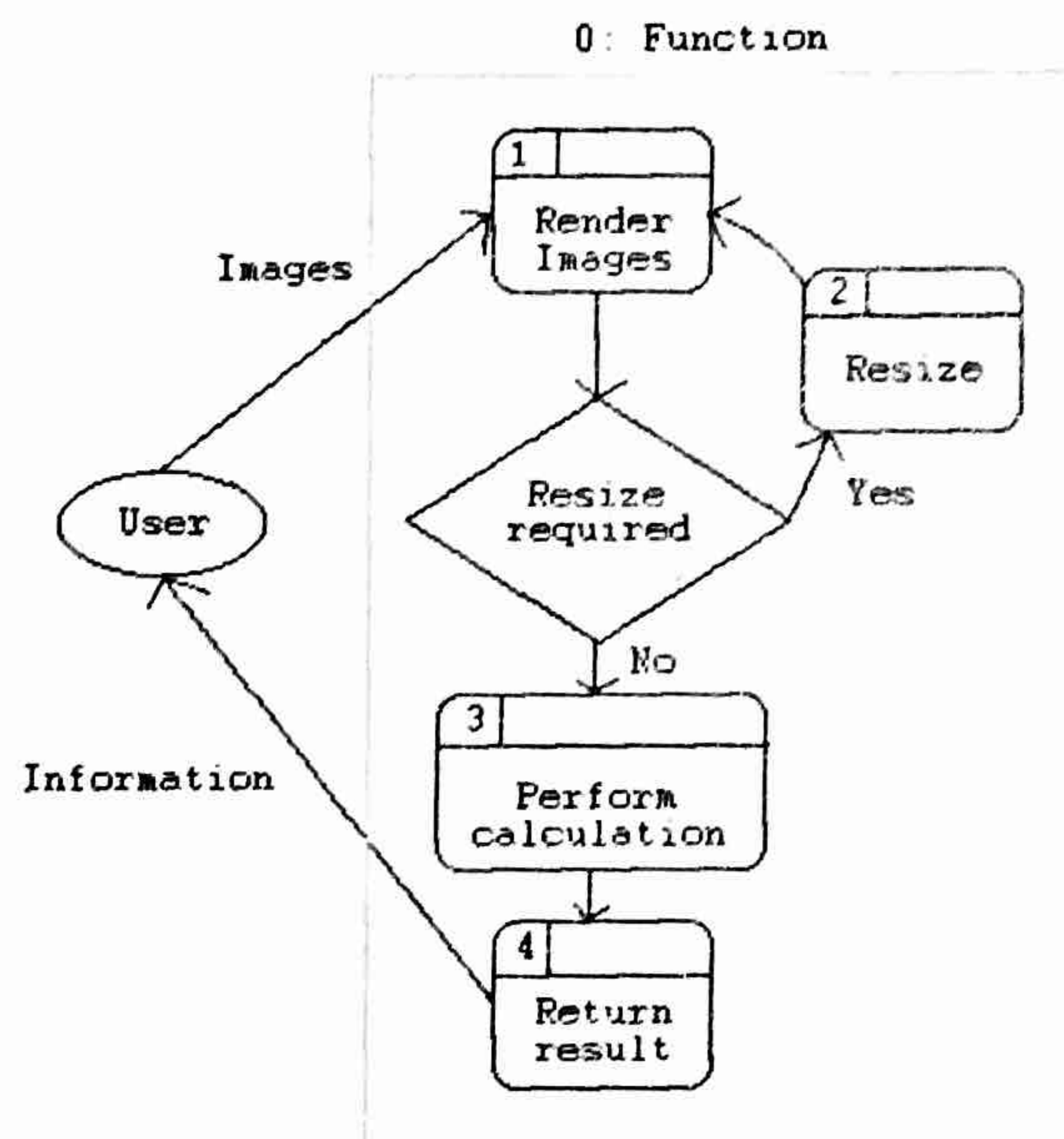


Figure 6.4: 2nd Level DFD (Layer 2) Proc. 1

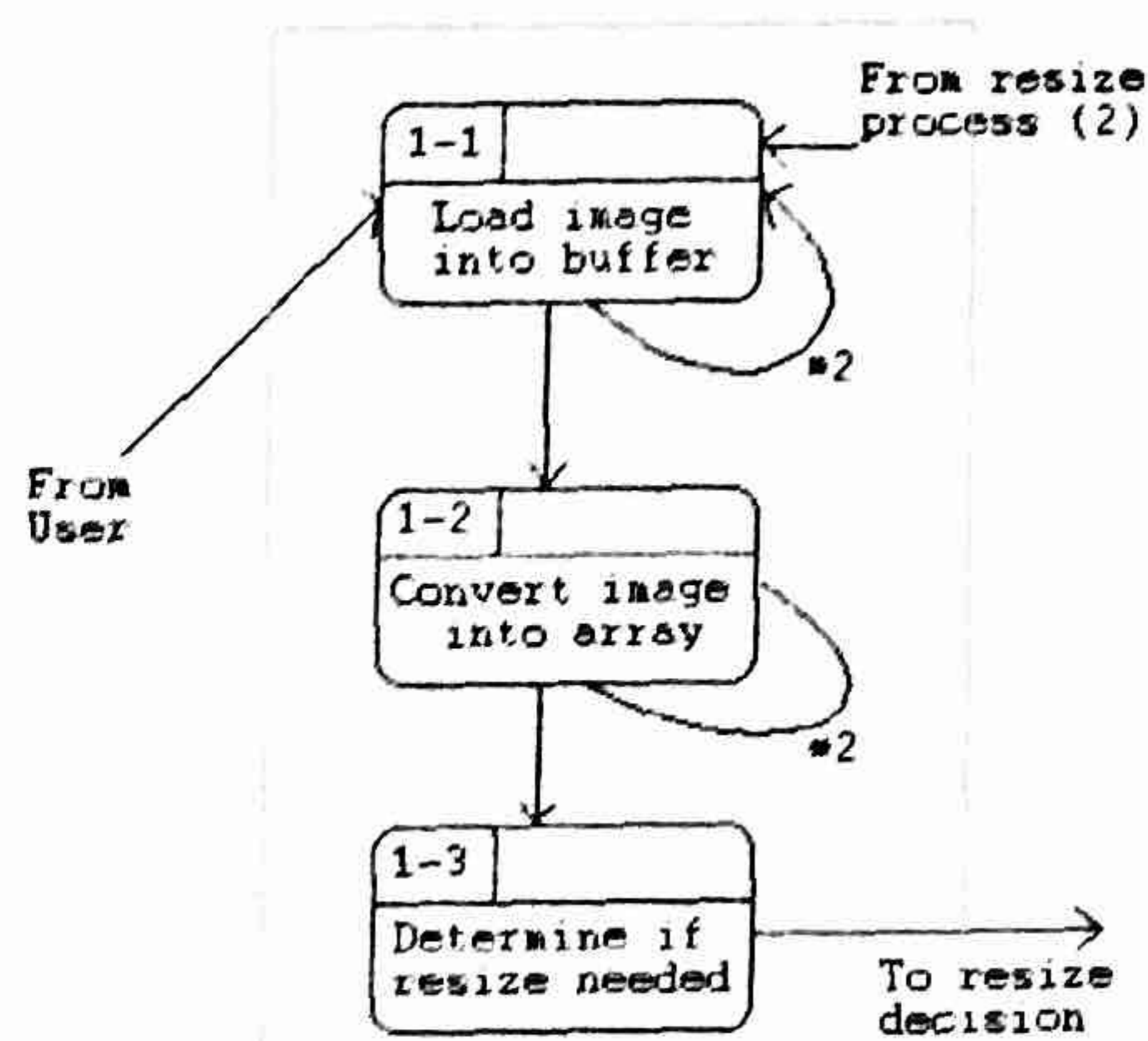


Figure 6.5: 2nd Level DFD (Layer 2) Proc. 2

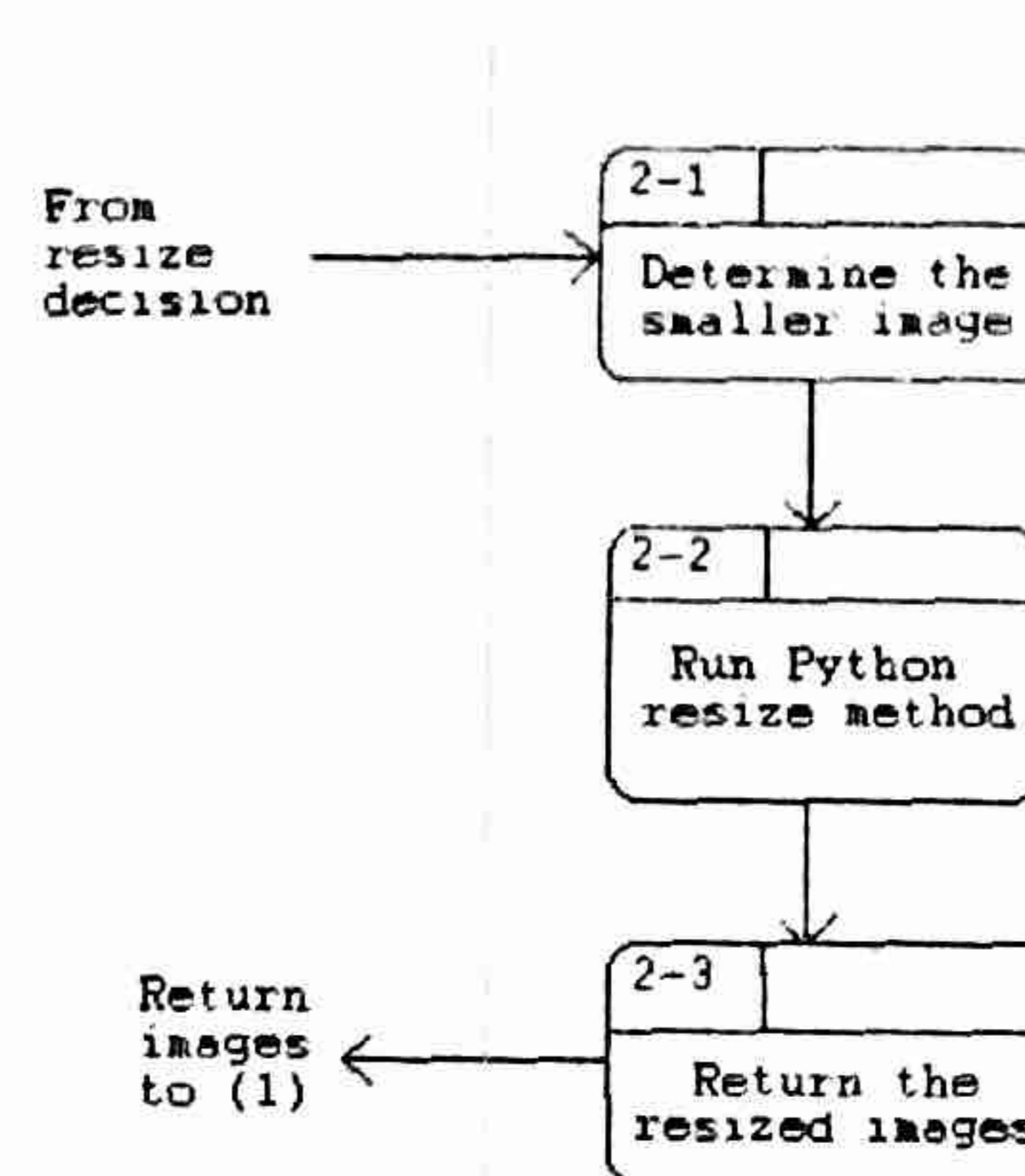
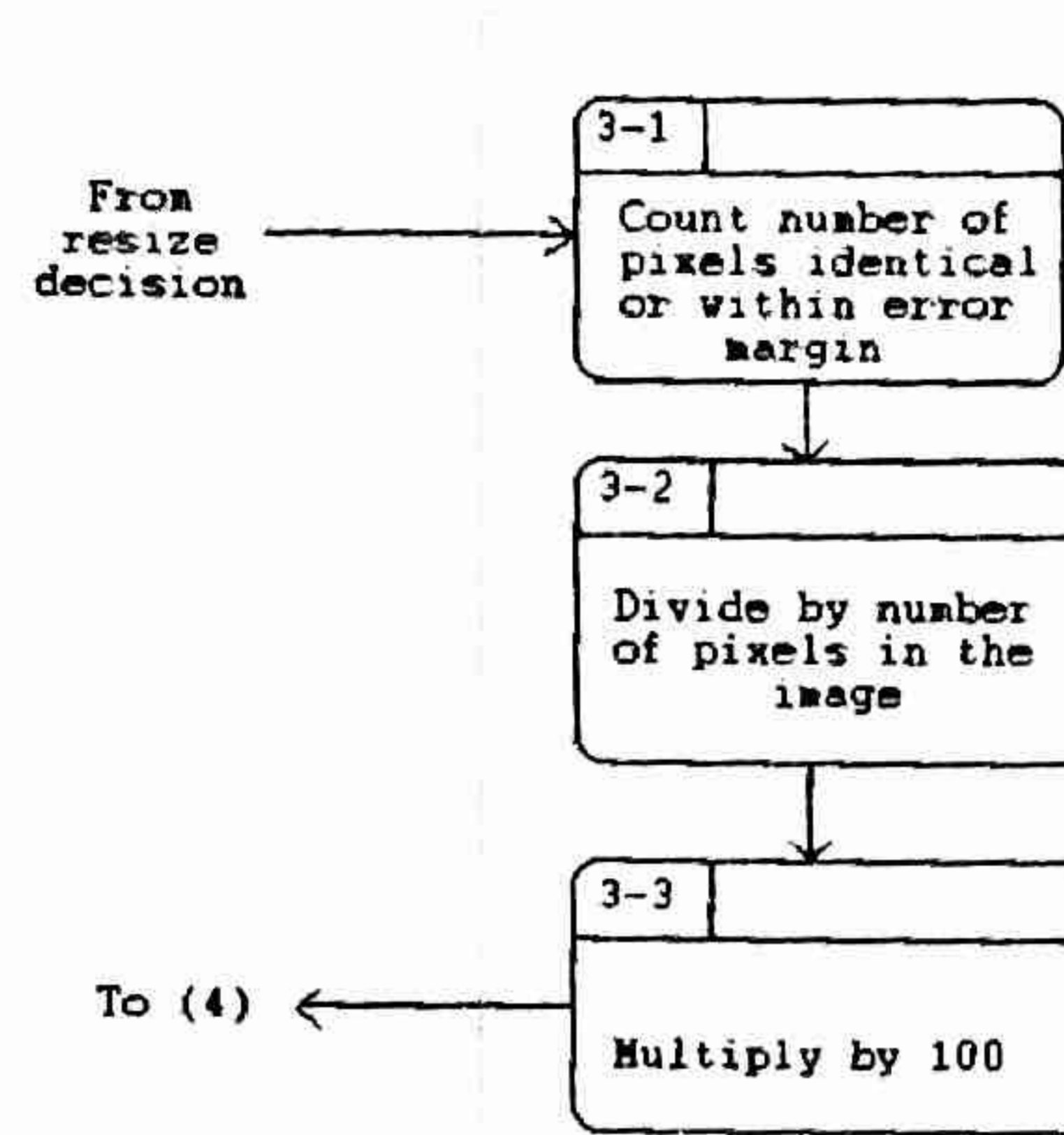


Figure 6.6: 2nd Level DFD (Layer 2) Proc. 3



This concludes the formal documentation and design stage. The next chapter addresses the implementation of the code using Python, an open-source free programming language that incorporates image manipulation facilities.

CHAPTER 7: IMPLEMENTATION

7.1 Introduction

The design was implemented using Python 2.3, available as open-source freeware from <http://www.python.org> and the Python Imaging Library, available as freeware from <http://www.pythonware.com/products/pil/>.

The design of the function is such that it is self-contained within one method, but structured logically throughout using sensible and relevant variable names. Two parameters are passed to the method when it is called, and these must be images within the same directory as the method, or on the path as defined in SYS.PATH.

7.2 Coding

The first step in my code is to import the relevant modules.

```
# PACKAGE IMPORT AND MODULE VARIABLE SETUP

import Image, ImageFilter, sys, time
count = float(0)    # var counts identical pixels
```

Figure 7.1: Importing external modules into the method

The '#' in Python denotes the following text as a comment. Indentation is extremely important in the language, as indentation denotes code structure. This is demonstrated in other code, below. Figure 7.1, above, shows that the Image, ImageFilter, sys and time modules are imported. Image and ImageFilter are part of the PIL package, and will enable image manipulation to take place. Sys allows the parameters to be on a different path than the local directory, and time allows the method to halt when instructed. This facility is important to allow the user time to digest information on the screen.

Next we convert each image into a List, an internal data type of the Python language. As a List is not easy to manipulate, the code also converts the list into an array of strings. In Python, if a string is a numerical value, it is recognised as such and can be evaluated numerically.

```
# IMAGE-ARRAY CONVERSION FOR IMAGE 1

pic1 = Image.open(image1) # img var pic1 = param1(image1)
imageinfo = list(pic1.getdata()) # pic1 converted to array of values
img1str = str(imageinfo) # img1str = str equivalent of array for comparison

# print "Image 1: " + img1str # display array contents (optional)
time.sleep(1) # system wait command to allow user to read screen
print ""

# IMAGE-ARRAY CONVERSION FOR IMAGE 2

pic2 = Image.open(image2)
imageinfo2 = list(pic2.getdata())
img2str = str(imageinfo2)
```

Figure 7.2: Image conversion into an array of strings

The next stage is to resize the images. Python's image resize method is used to do this, but first we must know which image is the smaller. Also, throughout the code, output to the user is required, as the following shows:

```

# CODE TO RESIZE IMAGE

if size1 < size2:
    smaller = size1
elif size1 > size2:      # These 4 lines set var "smaller" to smallest array
    smaller = size2
if size1 != size2:      # If images differ in size, execute the following...
    print ""
    print "WARNING!"
    print "-----"
    print ""
    print "These two images are not the same size."
    yesno = raw_input("Would you like to resize your images? (y/n)")
    if yesno == "y":
        if smaller == size1:
            print ""
            print "Your first image is smaller. Downsizing second image..."
            pic2 = pic2.resize(picsize1) # resizes 2nd image to match 1st
            time.sleep(2)
            print "Image 2 downsized. Image sizes now match."
            print ""
        if smaller == size2:
            print ""
            print "Your second image is smaller. Downsizing first image..."
            pic1 = pic1.resize(picsize2) # resizes 1st image to match 2nd
            time.sleep(2)
            print "Image 1 downsized. Image sizes now match."
            print ""

    if yesno == "n":
        print ""
        print "Note: This setting will compare your smaller image"
        print "      with an equally-sized portion of your larger"
        print "      image."
        print ""

    time.sleep(4)

# The above nested IF code compares the size of the arrays (the logical
# representation of the images). If they are not identical, the larger
# image is resized to match the smallest. The user has the option of
# whether or not to resize; if the user selects no, then a portion of
# the larger image equal to the size of the smaller image (from the top
# left hand pixel extending diagonally bottom-right) is compared. The
# accuracy and success of the RESIZE method is documented elsewhere.

```

Figure 7.3: Code to resize the images

The next step is to convert the formatted image(s) into new arrays for comparison.

This is achieved using similar code to that shown in Figure 7.2:

```
# ARRAY COMPARISON CODE

print "Comparing now..."
print ""
time.sleep(2)
imageinfo = list(pic1.getdata()) # Re-renders the altered image(s) into
imageinfo2 = list(pic2.getdata()) # a pair of arrays.

for a,b in zip(imageinfo, imageinfo2):

    if ((a >= (b * 0.9)) & (a <= (b * 1.1))) | ((b >= (a * 0.9)) & (b <= (a *
1.1))):

        count = count + 1

    else:

        count = count + 0
```

Figure 7.4: Generating new arrays for the formatted images

The code shown above incorporates an error margin of, in this example, 10%. This means that the code tests if the value of each pixel is within the numerical value of +10% or -10% of the equivalent pixel in the next image. This error value can be set to any value. Chapter 8 shows the test results when different error margins are tried.

The final stage is to generate the percentage of similarity between the two images. Figure 7.5 below shows how this is achieved.

```
# ACCURACY GENERATION CODE

print "Arrays compared. Generating percentage of similarity..."
time.sleep(1)
print ""

# The following 3 lines set var SIZE to equal the length of the arrays
# (both are now identical). The % accuracy is worked out by taking
# variable COUNT, dividing its contents by the actual number of pixels
# (or array elements) and multiplying by 100. Finally, the result is
# rendered into a string format as PYTHON cannot concatenate strings
# and integers using the PRINT command.

size = float(len(imageinfo))
percent = float((count / size)*100)
percentStr = str(percent)
print "These two images are " + percentStr + "% identical."
print ""
# print count # Optional
# pic1.show() # Shows the 1st image (optional)
# pic2.show() # Shows the 2nd image (optional)
```

Figure 7.5: Code to generate the similarity as a percentage of the two images

The final two lines in Figure 7.5 are optional, and will instruct Python to open the default image editor or browser on the operating system and display the images to the user.

7.3 Results of Implementation

I have produced four versions of this code. One version processes greyscale images without an error margin (checking for identity only), the next version processes 256-colour images without an error margin, the third version processes greyscale images with an error margin and the final version processes colour images with an error margin. The fourth version is the version shown above. I have omitted some code such as the method and variable setup, but the full version is contained in the Appendices and the differences between each version highlighted.

The next chapter specifies the test plan, and shows how testing was carried out and analysed. The discrete decision-making algorithm with test-result values is also introduced.

CHAPTER 8: TESTING

8.1 Introduction

As mentioned in Chapter 1, testing is of utmost importance to this system, as it will be implemented in a safety-critical environment. Therefore, a number of types of test have been carried out upon the software to find faults in the code. As I am using images, exhaustive testing is impossible as there can be an infinite number of different input patterns. It is important that the reliability of the software is extremely high; therefore throughout my iterative testing I have strived to eliminate every possible fault.

8.2 Risk Analysis

Before developing a test plan, I must specify the risks that will be taken should the system fail in a real-life implementation.

- The software will fail to identify a package as suspicious and this will influence the human team's decision on whether to destroy the package, endangering life.
- The software will misidentify a package as suspicious too often, leading the human team to lose confidence in the reliability of the system.
- The software will fail when presented with some inputs, leaving the human team without a decision support system.

8.3 Test Plan

The test plan that is to be developed must plan tests to address the risks documented above. My tests must therefore be organised into three categories:

- Tests to check if the system positively identifies a given package.
- Tests to check if the system does not identify a package that has no relation to the image to which it is being compared.
- Tests to check if the system will fail when presented with erroneous inputs.

Specifically, then, I have developed a number of tests for the system involving pairs of images. All tests shall be run on the greyscale version of the design with the error margin set to a variable value, specified below.

8.4 Positive Identification Testing

Hypothesis:

The software will present a high similarity percentage.

Figure 8.1: Test Set 1 and results


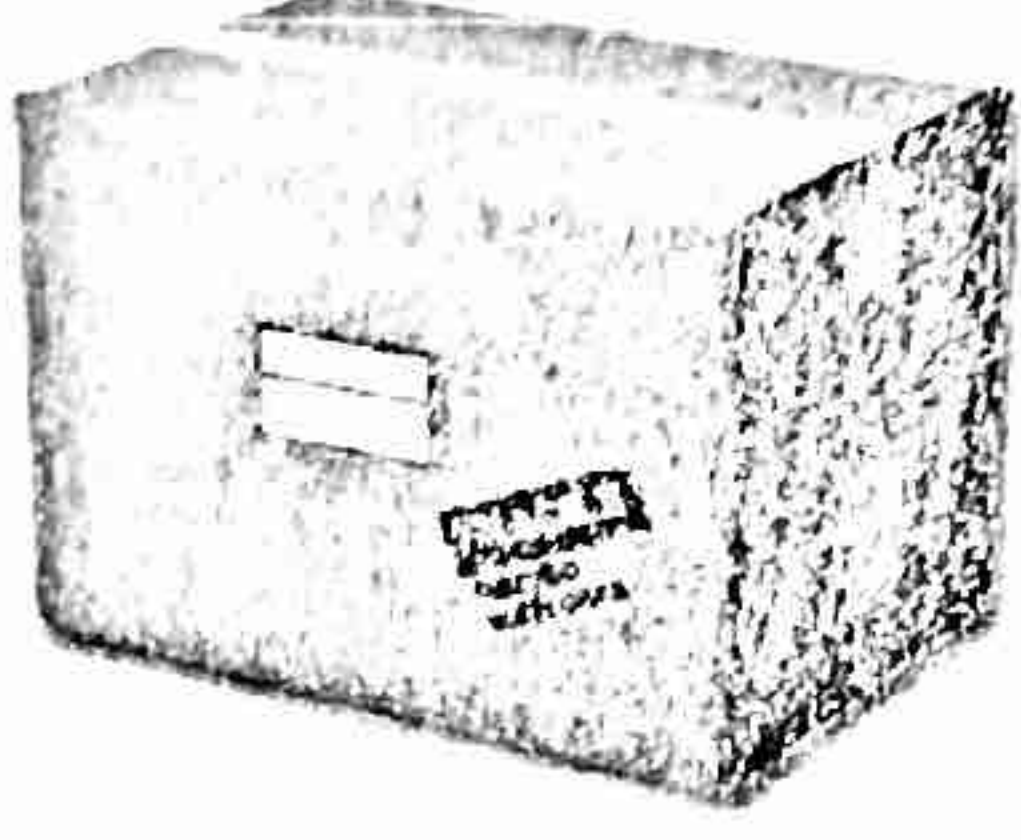
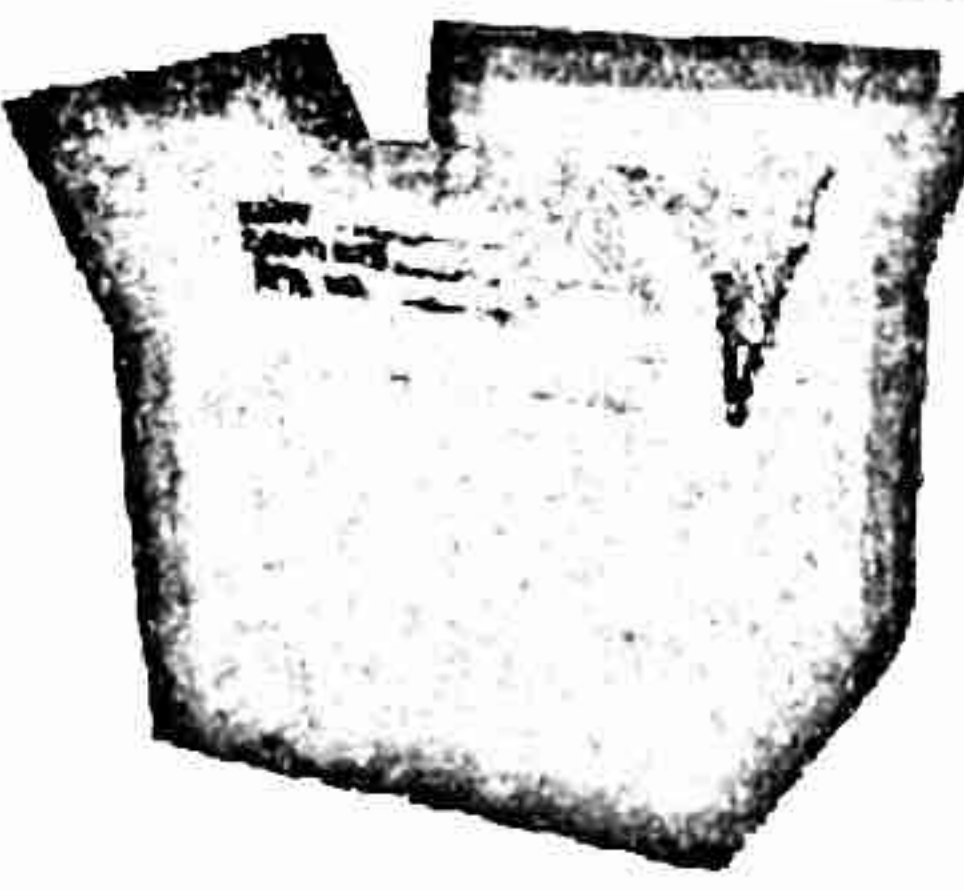
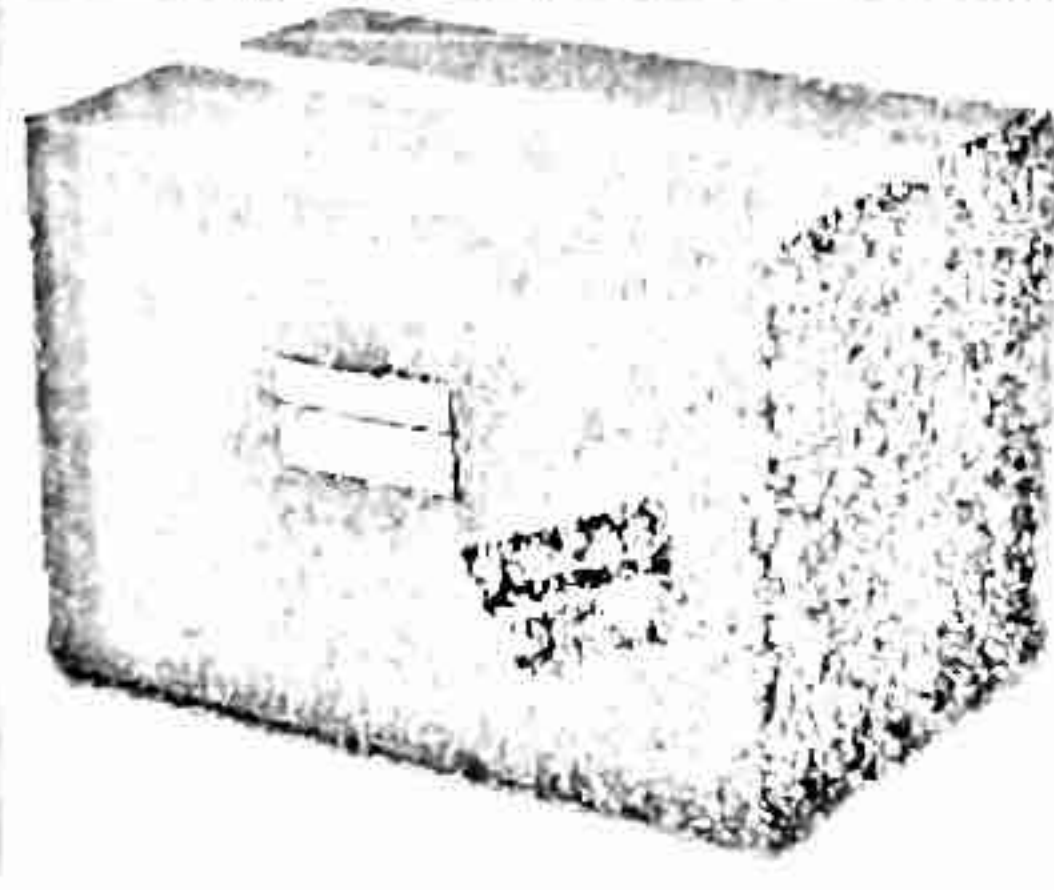




	Database Image	Test Image
Original Picture		
Picture after greyscale conversion and resize		

Figure 8.2: Test Set 2 and results

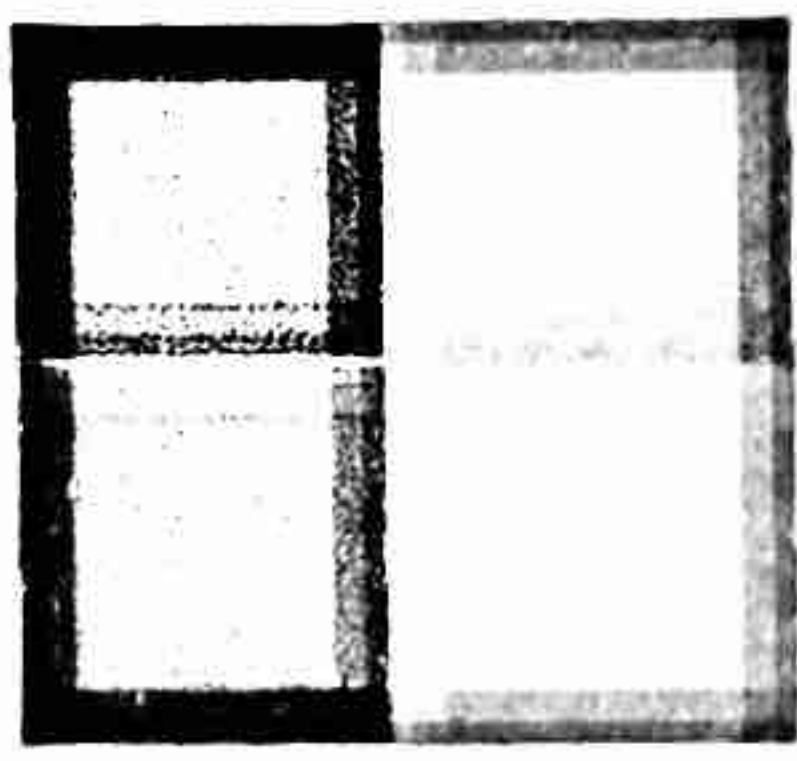
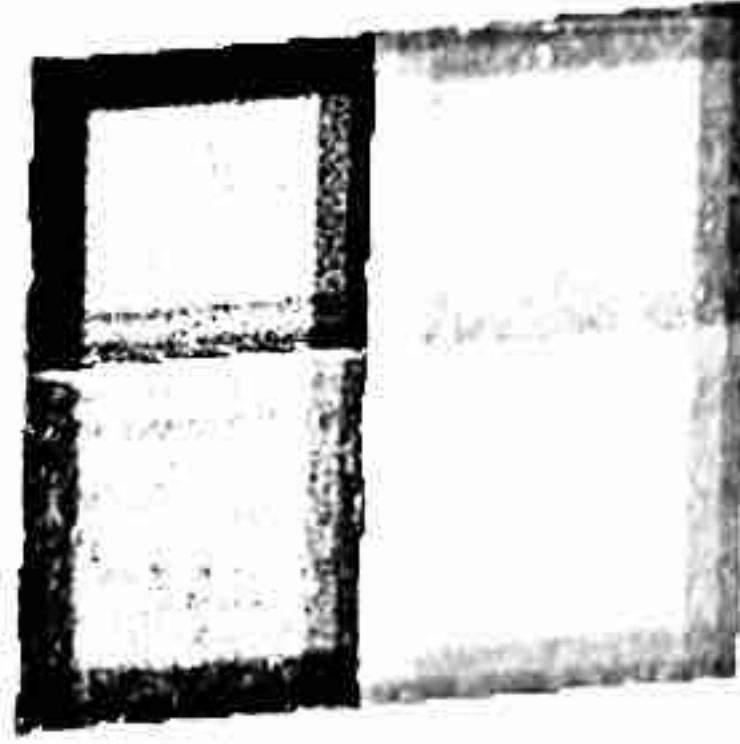
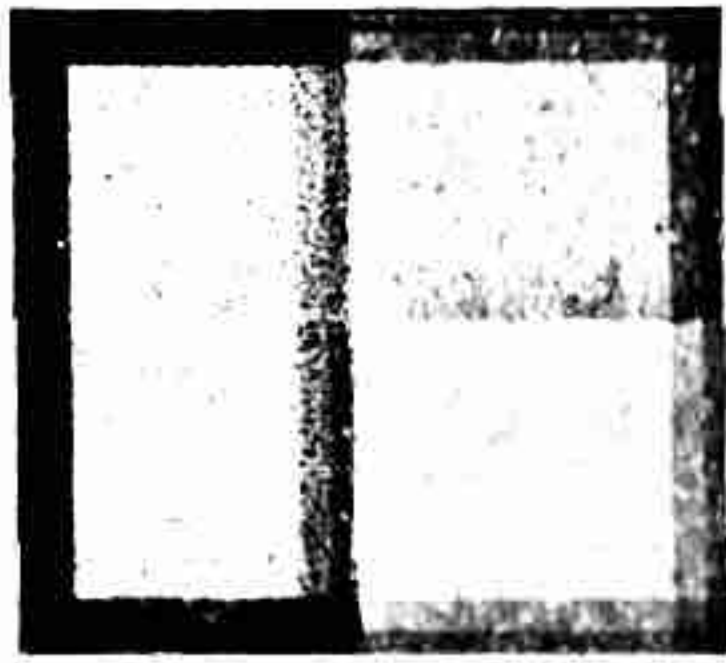
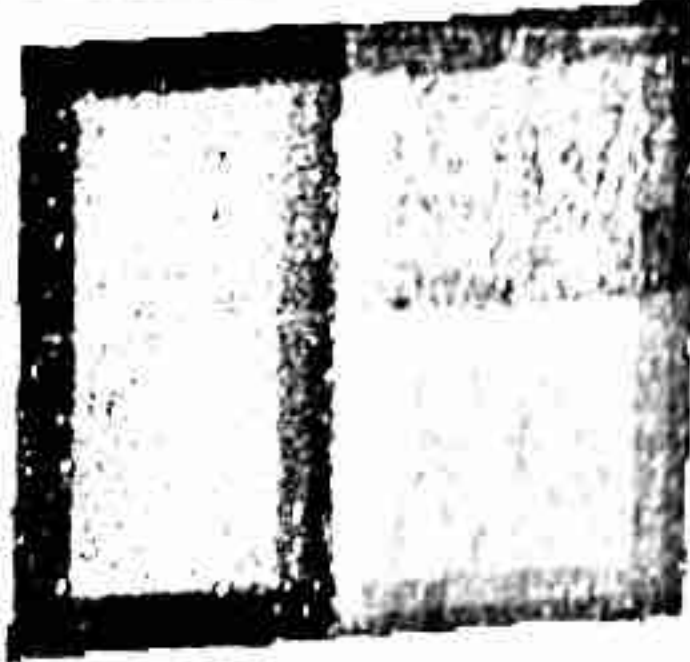
	Database Image	Test Image
Original Picture		
Picture after greyscale conversion and resize		

* Source: www.boxdepot.us/images/book_box_1.jpg

** Source: www.selectmovers.com/images/cardboard_box.gif

☐ Source: www.obkb.com/dcljr/IMAGES/beauty/beauty1.jpg

Figure 8.3: Test Set 3 and results

	Database Image	Test Image
Original Picture		
Picture after greyscale conversion and resize		

8.5 Negative Identification Testing

Hypothesis:

The software will present a low similarity percentage.

Figure 8.4: Test Set 4 and results




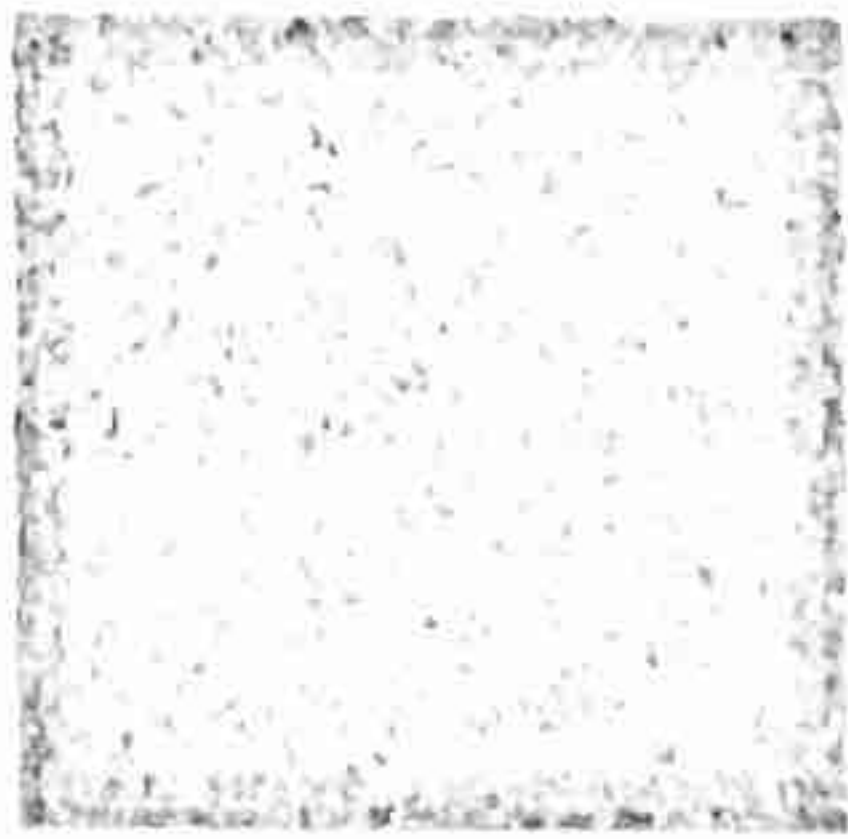
	Database Image	Test Image
Original Picture	 *	 **
Picture after greyscale conversion and resize		

Figure 8.5: Test Set 5 and results

* Source: <http://coveringkidsandfamilies.org/communications/partners/business/example-carton.jpg>

** Source: <http://fxhome.com/alamdv2/plugins/rawfiles/1440/large>

AN IMPROVED IMAGE RECOGNITION SOFTWARE SYSTEM
FOR IMPLEMENTATION IN BOMB-DISPOSAL ROBOTICS






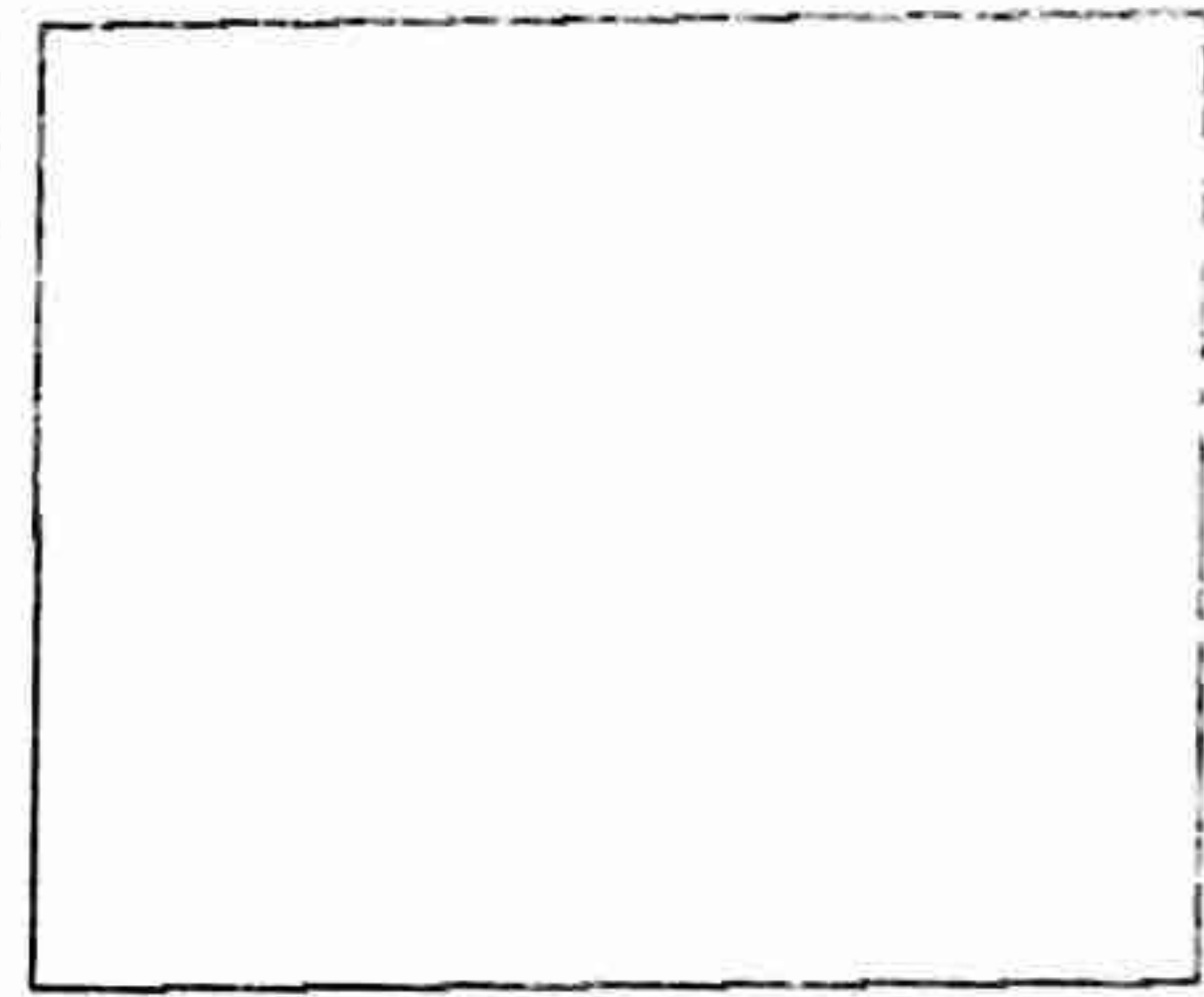

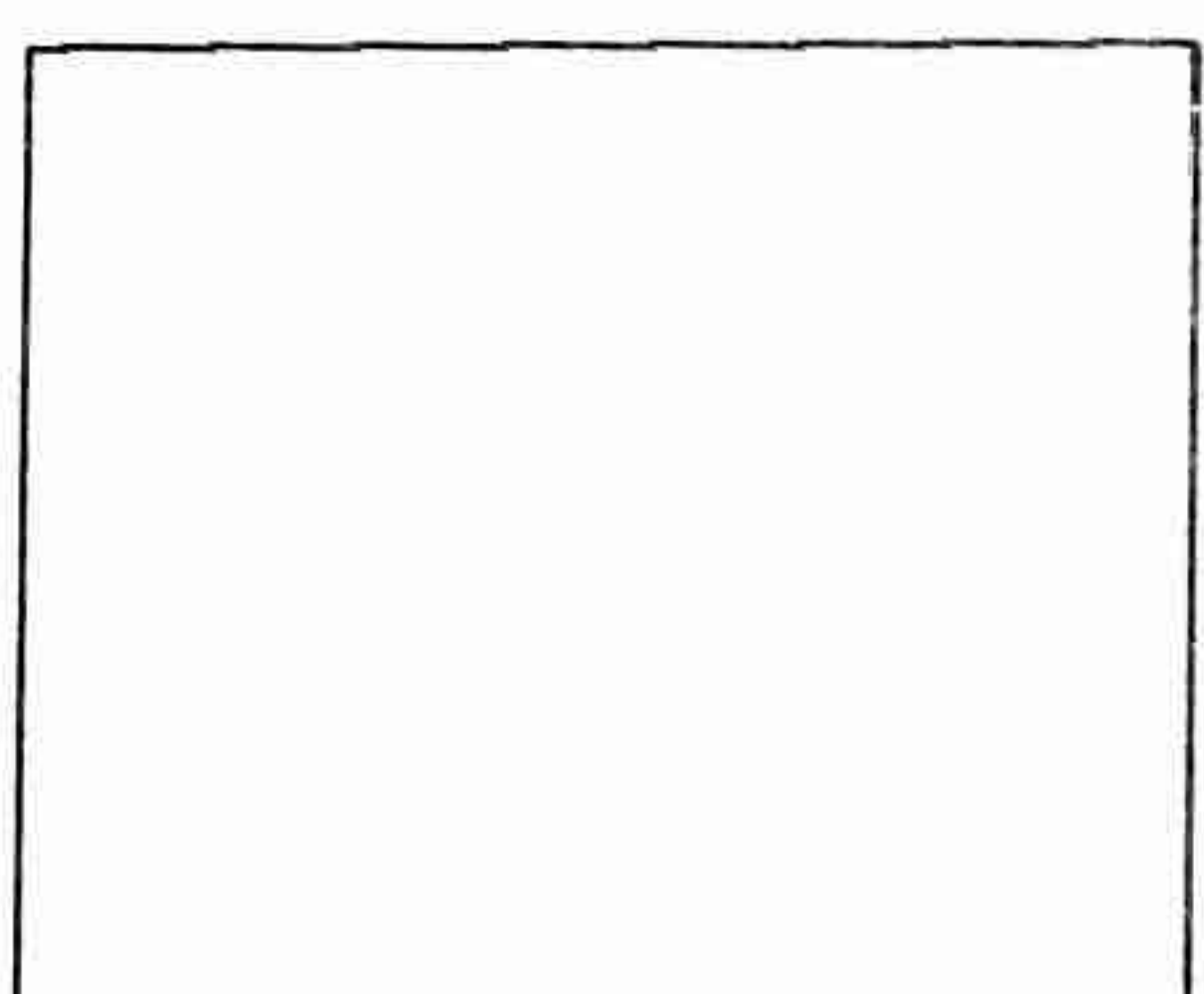
	Database Image	Test Image
Original Picture		
Picture after greyscale conversion and resize		

Figure 8.6: Test Set 6 and results

	Database Image	Test Image
Original Picture		
Picture after greyscale conversion and resize		

* Source: www.fotosearch.com/comp/BDX/BDX327/bxp60100.jpg

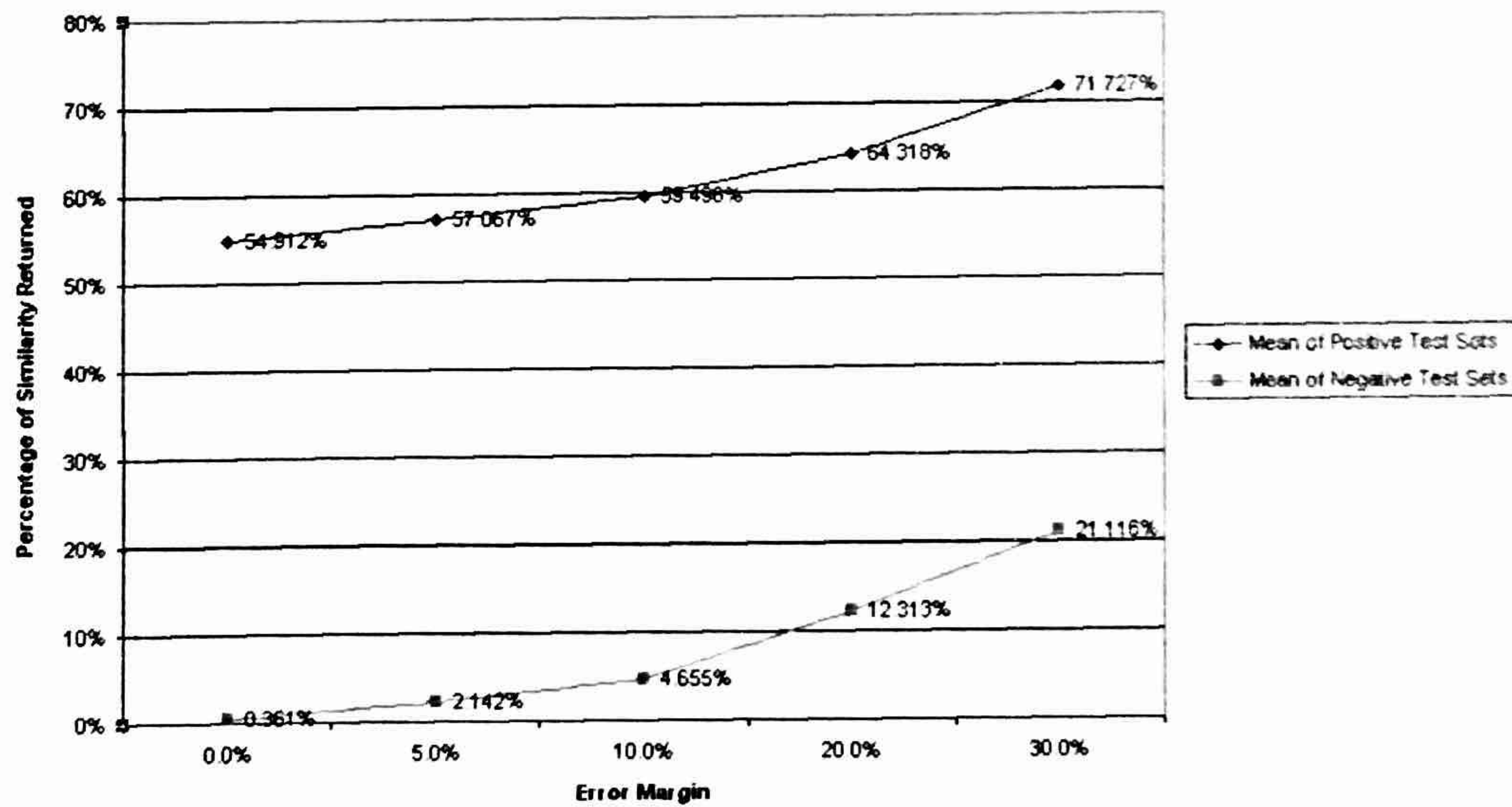
** Source: <http://www.cs.duke.edu/~goodson/vision/Phase1/User%20Hand.jpg>

8.6 Test Results

Figure 8.7: Table of Results

	PERCENTAGE OF ERROR MARGIN				
	0 0%	5 0%	10 0%	20 0%	30 0%
POSITIVE SET					
Test Set 1	25.342	31.807	39.094	52.274	64.841
Test Set 2	54.623	54.623	54.623	55.907	64.502
Test Set 3	84.772	84.772	84.772	84.772	85.837
Mean	54.912	57.067	59.496	64.318	71.727
Range	59.430	52.965	45.678	32.498	21.335
NEGATIVE SET					
Test Set 4	0.924	6.038	13.260	35.303	57.658
Test Set 5	0.158	0.389	0.706	1.637	5.690
Test Set 6	0.000	0.000	0.000	0.000	0.000
Mean	0.361	2.142	4.655	12.313	21.116
Range	0.924	6.038	13.260	35.303	57.658

Figure 8.8: Graph to show the mean of the test results in two categories



8.7 Results Analysis

The tests were in two categories of set; one, to test positively whether two similar images produced results to prove that the images were similar. The second was a negative test set, to test whether images that were completely different were recognised as completely different by the software.

Each test series incorporated three test sets. For each series, there was a simple image pair, a semi-detailed image pair and a complex image pair. Furthermore, in the negative test series, the last test set with the black and white images was designed to be a boundary test to check the algorithm was performing its function accurately.

It is interesting to note that the values between 0% E.M. and 30% E.M. for the simple image, test set 3, in the positive series, remained nearly identical throughout testing. I surmise the reason for this being that the image contained four very different colours; as the E.M. rose, the pixels that were not identical were not within the 30% margin. This effect is also evident in the complex picture, the face (test set 2) between 0% and 10% - the reason for this could be that the picture has a high level of inter-pixel contrast and the pixels did not fall within the E.M. until an E.M. of 15% was reached. The percentage of similarity of the semi-detailed picture, test set 1, rose normally throughout.

For the negative testing, it is interesting to note that the percentage of similarity rose spectacularly in proportion to the increase in the error margin. With an E.M. of 30%, one result rose as high as 57.658%.

8.8 Threshold Analysis

In order to find a threshold where the software can make a discrete decision on whether an image is similar or not, a value must be found at the halfway point between the highest returned percentage of similarity in the negative test series and the lowest returned percentage of similarity in the positive test series for any given error margin value. Mathematically:

$$T = ((\min[\text{pos}] - \max[\text{neg}]) / 2) + \max[\text{neg}]$$

Where T = threshold value, $\min[\text{pos}]$ = the minimum value of the positive test series for a given E.M. value and $\max[\text{neg}]$ = the maximum value of the negative test series for the same given E.M. value.

These boundaries are shown in the following table:

Figure 8.9: Table to show ideal thresholds for discrete decision-making

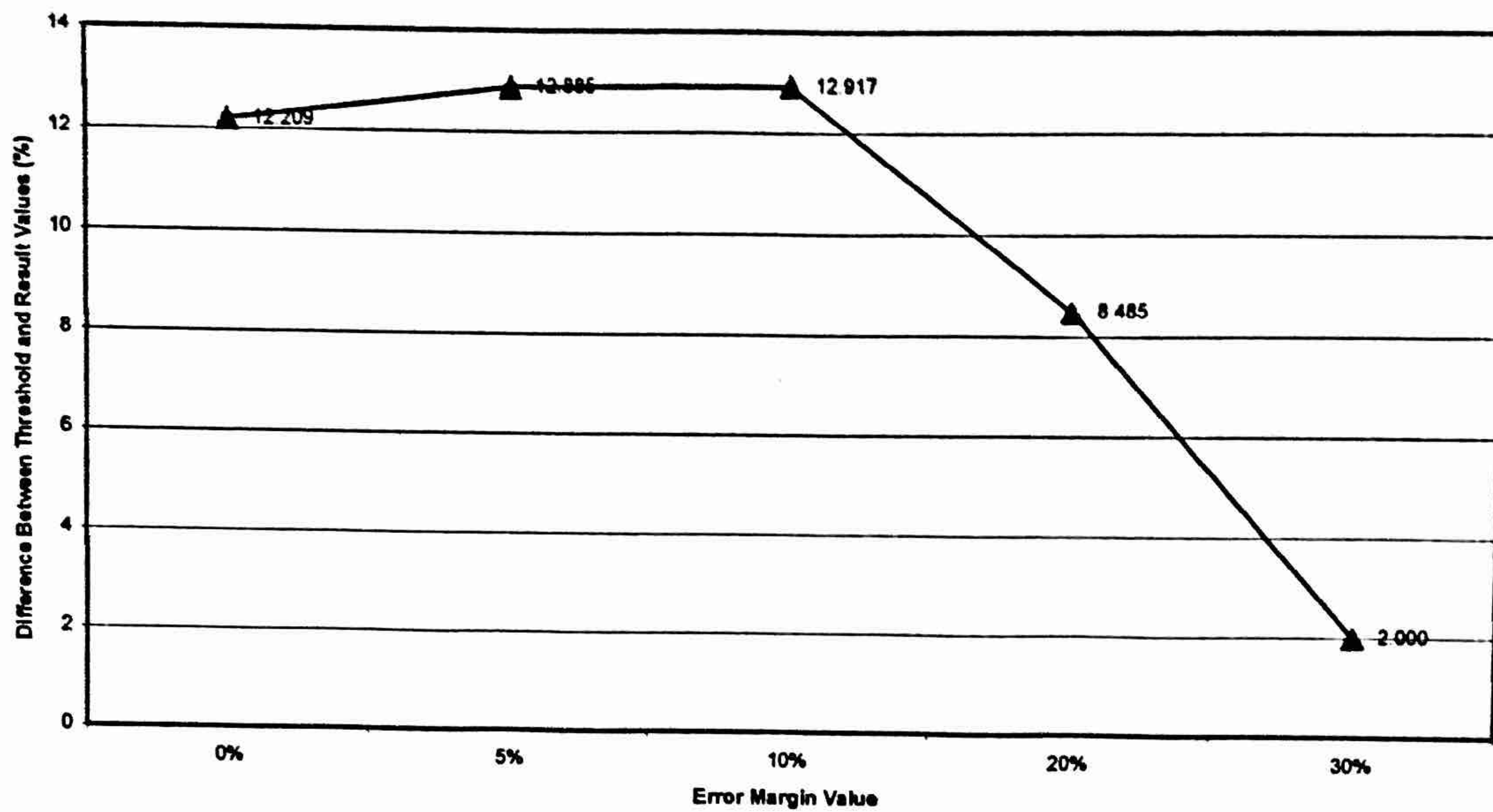
	Error Margin Value				
	0%	5%	10%	20%	30%
Threshold	13.133	18.922	26.177	43.789	62.502

Now, logically following this idea, to calculate the best E.M. value to use, we must compare the range between either the smallest result from the positive test series and the threshold value or the largest result from the negative test series and the threshold value (these two values will be identical due to the method used to calculate the threshold value). The E.M. value with the largest range will statistically be the most reliable E.M. value to take with the least chance of causing system failure, as it has the largest "safety buffer". The results are shown in the following table and line graph:

Figure 8.10: Table showing error margin values with safety zones

	Error Margin Value				
	0%	5%	10%	20%	30%
Threshold	13.133	18.922	26.177	43.789	62.502
Range	12.209	12.885	12.917	8.485	2.000

Figure 8.11: Graph showing the relative reliability of the algorithm against different error margin values



Looking at this data, we can conclude therefore that an error margin of 10% with a discrete threshold value of 26.177% is the most reliable to use. This means I can make an addition to the source code and enable the function to give a discrete "similar" or "not similar" answer in addition to the percentage, which will assist the human team further. The extra Python code, to be added at the end, is:

```
if percent < 26.177:
    print "Statistically, the images are dissimilar."

elif percent >= 26.177:
    print "Statistically, the images are similar."
```

The results shown above, and as depicted by the graph, show clearly that there were no failures in the software for these test inputs. Figure 8.8 above plots the means of the two test sets, and the results are widely spaced. This shows the software fulfils the hypotheses above.

While these tests have been unsuccessful, meaning that they have not caused the software to fail, the reliability of the algorithm has been firmly established. There is

one final set of tests to run, as specified in the Risk Analysis section above, and that is to test whether the system fails when presented with erroneous inputs.

8.9 Erroneous Input Testing

For these tests I shall use the following methods:

- Stress testing – the algorithm will be presented with an image far in excess of the expected inputs – in this case, a 32-bit colour digital camera image with a resolution of approximately 2,000,000 pixels (2 million pixels).
- False files – the algorithm will be presented with a file that is misnamed and is not an image file.

TEST 1: STRESS TESTING

Result: Inconclusive. The test was run on a system with the following specifications: AMD Athlon 800MHz processor, 256MB SDRAM, 133MHz Motherboard bus. The algorithm caused the system to halt – I surmise this was due to memory overflow.

Conclusion: The algorithm should theoretically be able to run well on large image sizes, however I do not have the hardware facilities to test this theory. In implementation therefore, I would recommend a maximum colour range of 256 and a maximum resolution of 640x480 pixels.

TEST 2: FALSE FILE TEST

Result: The files passed to the algorithm were a .WAV file and a .BMP file. The test was successful and the system failed. This is due to a fault – there is no error-catching facility built into the algorithm. Rather, Python caught the error, displayed as “IOError: cannot identify image file”.

Conclusion: A possible improvement to this algorithm would be to build in error-catching facilities that do not cause the function to halt unexpectedly.

The next chapter evaluates the project and concludes the report, and highlights possible improvements to be made in the future.

CHAPTER 9: EVALUATION AND CONCLUSION

9.1 Summary of the Dissertation

This report began by introducing the background area of the proposed software system, and identifying strengths and weaknesses in current research. The initial chapter provided a justification for developing the software, and outlined the environment in which it would be used.

The literature review in Chapter 2 critically assessed some of the literature relevant to the software. The findings in each case are discussed and the weaknesses and the relevance of the findings to my proposed system are examined. A broad overview of the area of image manipulation was given, and some directions for my research and development were established.

Chapter 3 outlined the research methods used to gather information to assist me in defining my objectives, my methods and my design. Key information was gained from the interviews while notable alternative work was discovered.

The fourth chapter compared two key development methodologies suitable for application to this project. Justification was made for the choice of methodology and the methods contained within them were applied to the development plan of my project.

Chapter 5 showed the preliminary designs that the final design was based upon.

The sixth chapter shows the formal design specifications, notarised in data flow diagrams and JSD structure diagrams. This documentation provided an invaluable resource for the coding and implementation of the final product.

Chapter 7 shows the methods used to implement the final product, and the coding that was generated. My methods are examined in detail and justification is given to the choice of methods used.

Chapter 8 defines a formal test plan and shows how the tests were carried out. The results of the test are collected, broken down, analysed and relevant conclusions are

drawn from the results. An improvement to the product is then designed based on the test results, that enables the final product to adhere to the original specifications.

Finally, this chapter will evaluate and conclude the report, looking at the contribution this project has made to research and development in the appropriate field, possible future developments based on this work and the boundaries of the software.

9.2 Research Contributions

The introductory material for this project stated how commercial research and development tends not to focus on the software element of robotics, but rather the capabilities of the robot. By producing a software system that can work alongside the most advanced robot to enhance the overall performance, I have made a significant advance in the functionality of bomb-disposal robots. The methods I have used within the software have been proved to be reliable and safe, and present a solution to the image-recognition problem that has not been used before.

9.3 Future Development

Of course, there is always room for improvement in any developed system. To identify the areas that could be improved, I must first identify the areas in which the software is weak; where the functionality could be improved or is non-existent:

- The software can only process images below a certain size
- The software cannot handle non-bitmap image formats
- The software does not have a facility for the user to set the error-margin
- There is no help documentation for the user
- The database system and the graphical user interface have not been implemented
- There is no facility for vector recognition to work alongside the existing methods
- The code could be modularised in order to create a better structure

These are the deficiencies in the software, and future research and development could focus on any one of these criteria.

9.4 Comparison to Aims and Objectives

The aims and objectives of the project were quite broad, however the software has met the demands made. It is a functional image-recognition module capable of examining two images, resizing if necessary, and comparing them intelligently in order to differentiate between images that are similar or identical and images that are not. The software is able to make a discrete decision based on proved algorithms that can be used to support a human decision. The software has therefore met the aims and objectives and can be considered a success.

END OF REPORT

BIBLIOGRAPHY, APPENDICES AND FIGURES LIST FOLLOW

BIBLIOGRAPHY AND REFERENCES

Newspapers, Magazines and Periodicals:

Jefferson, P.A.S., "Mines, Damn Lies and Statistics", *The Manchester Guardian*, 3 September 1997 pg. 17, accessed from microfiche 17/12/05.

Military Technology Magazine, "The British Army's Future Structure", March 2005, Vol. 29 Issue 3, pp. 21-65

Journal Articles

Anastassiou, D. (1993), "Digital Television", *Proceedings of the IEEE*, published by the IEEE, Volume 82, Issue 4, pp. 510-519.

Benyon, D. (1993), "Adaptive Systems: A Solution to Usability Problems", *User Modelling and User-Adapted Interaction*, published by Springer Science & Business Media Ltd., Volume 3, Issue 1, pp. 65-87.

Bridle, M. & Australian Broadcasting Corporation (ABC Aus.), "Satellite Broadcasting in Australia", *IEEE Transactions on Broadcasting*, published by the IEEE, Volume 34, Issue 4, pp. 425-429.

Hanson, J. (1994) Morphological Constrained Feature Enhancement with Adapted Cepstral Compensation (MCE-ACC) for Speech Recognition in Noise and Lombard Effect *IEEE Transactions on Speech and Audio Processing* 2(4) pg. 598

Hatton, J. (2004), "Digital Video Imaging with Small Telescopes - Possible Applications for Research and Education using the SOFIA Upper Deck Research Facility", *Proceedings SOFIA Upper Deck Science Opportunities Workshop*, published by NASA Ames Research Facility, CA, USA, pp. 48-50.

Hollingum, J. (1999) Robots for the dangerous tasks, *Industrial Robot: An International Journal* 26(3), 178-183 Publisher: MCB UP Ltd.

Kapoor & Bowen (2005), "A Formal Analysis of MCDC and RCDC Test Criteria", *Software Testing, Verification and Reliability*, published by J. Wiley & Sons Ltd., Volume 15 Issue 1, pp. 21-40.

McCann, T. and Clark, E. (2005) Using unstructured interviews with participants who have schizophrenia, *Nurse Researcher* 13(1), 7-18 Supplied By: EBSCO Host Research Databases

McIvor, A. (1989), "Edge Recognition in Dynamic Vision", *Computer Vision and Pattern Recognition, Proceedings CVPR '89, IEEE Computer Society Conference on 4-8 Jun 1989*, pp. 128-133

Neuhaus, P. & Kazerooni, H., "Industrial-Strength Human-Assisted Walking Robots", *Robotics and Automation Magazine*, published by the IEEE, Dec. 2004 Volume 8, Issue 4, pp. 18-25.

Pfeifer and Gómez (2005) Interacting with the real world: Design principles for intelligent systems *Artificial Life and Robotics* 9(1) pg. 1 Publisher: Springer-Verlag Tokyo Inc.

Reimers, U. (1998), "Digital Video Broadcasting", *Communications Magazine*, published by the IEEE, Volume 36 Issue 6, pp. 104-110.

Siddique, J. & Barner, K. (1998), "Wavelet-based multiresolution edge-detection utilising gray level edge maps", *Proceedings of ICIP '98: 1998 International Conference on Image Processing 4-7 Oct '98*, Volume 2, pp. 550-554.

Online Sources

Answers.com, "Bomb Disposal", available from <http://www.answers.com/topic/bomb-disposal>, accessed 19/12/05 citing an article of the same title from Wikipedia.com, available at http://en.wikipedia.org/wiki/Bomb_disposal, last updated 19/12/05.

BBC News UK, "The IRA Campaigns in England", published on-line on 4/3/01, accessed 17/12/05, available at <http://news.bbc.co.uk/1/hi/uk/1201738.stm>.

Center for Domestic Preparedness: Homeland Security, "Awareness Level WMD Training: Explosive Devices", published as an independent paper by the author, date unknown, available from http://cdp.dhs.gov/pdfs/agert/Explosive_Devices.pdf.

National Law Enforcement and Corrections Technology Center, "*Building a better bomb robot*", TECHBeat Magazine, Summer 2004 edition, <http://www.nlectc.org/techbeat/summer2004/04-Building%20A%20Robot.pdf>, accessed 20 October 2005.

Other Materials

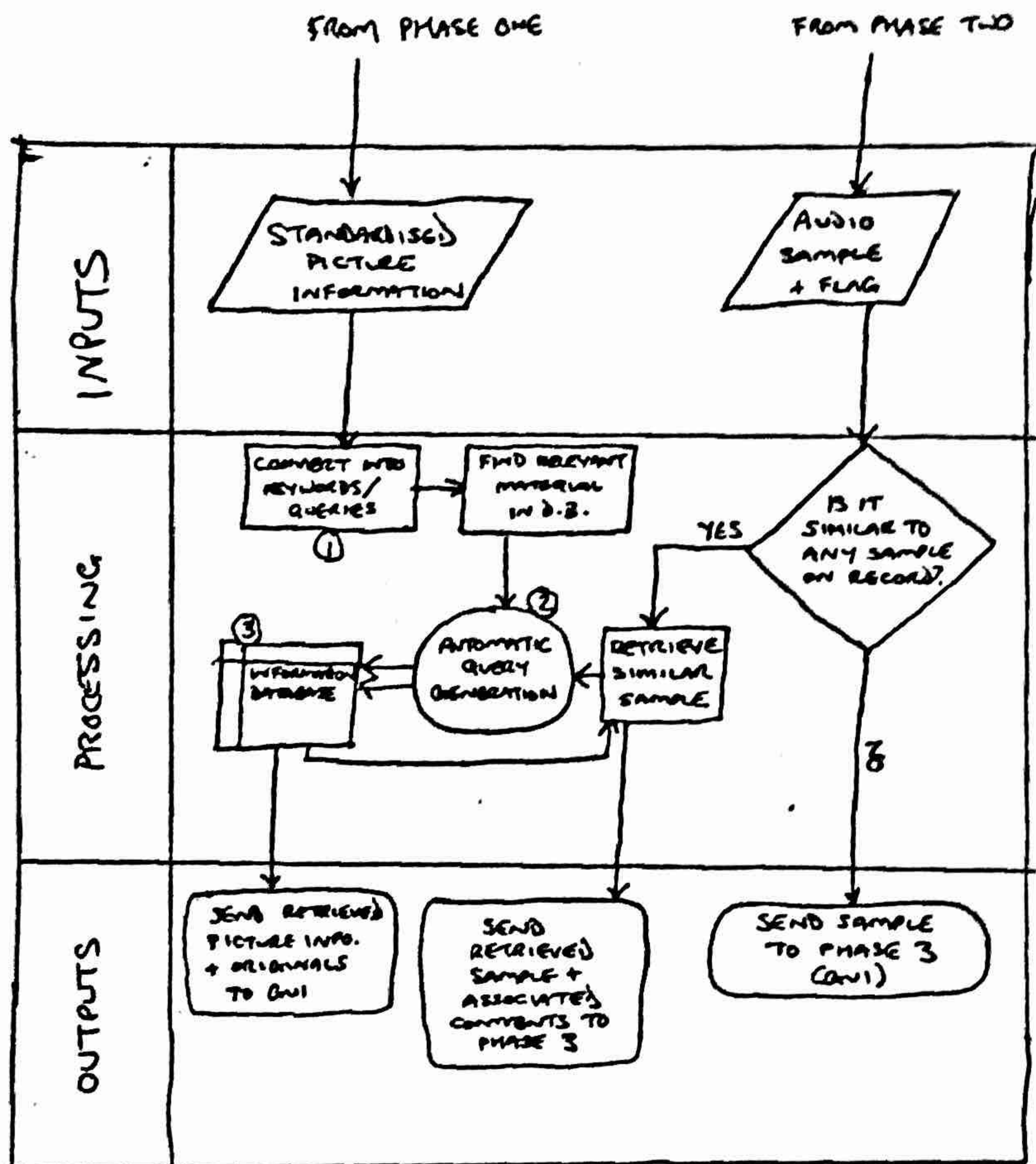
CITED: Dingwall R (1997) Accounts, interviews and observations. In Miller G, Dingwall R (Eds) *Context and Method in Qualitative Research*. London, Sage.

Hawker, S. (ed), Soanes, C. (ed), Spooner, A. (ed) *The Compact Oxford English Dictionary, Thesaurus and Wordpower Guide*, 2002 edn, Oxford University Press, Oxford, England

APPENDICES

APPENDIX A: Preliminary Designs

PHASE TWO: INFORMATION RETRIEVAL



NOTES:

- ① THIS PROCESS WILL INVOLVE FITTING THE PICTURE INFORMATION (SIZE, COLOUR DISTRIBUTION, PRESENCE OF WIRES ETC.) INTO A STANDARD TEMPLATE FOR USE BY THE AUTOMATIC QUERY GENERATION PROCESS.
- ② THIS PROCESS WILL TAKE IN STANDARDISED INPUTS AND QUERY THE D.B. ON THE RESULTS WILL BE SENT AS SHOWN ABOVE.
- ③ THIS IS A DATABASE CONTAINING INFORMATION ABOUT POTENTIAL EXPLOSIVES, SIZES, TYPES AND ALL OTHER PERTINENT INFORMATION. IT ALSO CONTAINS GENERAL INFO., ALL OF WHICH THAT IS SELECTED WILL BE RELATED TO THE GUI FOR USE BY THE HUMAN TEAM.

Figure A1.1: Preliminary Design of theoretical Phase Two (database interaction)

APPENDIX B: Alternative Code Implementations

Version 1: Greyscale without E.M. facility

```
def convert(image1, image2):

    # PACKAGE IMPORT AND MODULE VARIABLE SETUP

    import Image, ImageFilter, sys, time
    count = float(0)    # var counts identical pixels

    # IMAGE-ARRAY CONVERSION FOR IMAGE 1

    pic1 = Image.open(image1).convert("L") # img var pic1 = param1(image1)
    imageinfo = list(pic1.getdata()) # pic1 converted to array of values
    img1str = str(imageinfo) # img1str = str equivalent of array for comparison

    # print "Image 1: " + img1str # display array contents (optional)
    time.sleep(1) # system wait command to allow user to read screen
    print ""

    # IMAGE-ARRAY CONVERSION FOR IMAGE 2

    pic2 = Image.open(image2).convert("L")
    imageinfo2 = list(pic2.getdata())
    img2str = str(imageinfo2)

    # print "Image 2: " + img2str # by now image2 is in a second array
    time.sleep(1)
    print ""

    # DEPENDENT VARIABLE SETUP

    size1 = float(len(imageinfo)) # var size1 = no of elements in imageinfo1
    size2 = float(len(imageinfo2)) # var size2 = no of elements in imageinfo2

    picsize1 = pic1.size # var picsize1 = actual pic size (as tuple)
    picsize2 = pic2.size # var picsize2 = actual pic size (as tuple)

    # CODE TO RESIZE IMAGE

    if size1 < size2:
        smaller = size1
    elif size1 > size2:
        smaller = size2 # These 4 lines set var "smaller" to smallest array
    if size1 != size2:
        # If images differ in size, execute the following...
        print ""
        print "WARNING!"
        print "-----"
        print ""
        print "These two images are not the same size."
        yesno = raw_input("Would you like to resize your images? (y/n)")
        if yesno == "y":
            if smaller == size1:
                print ""
                print "Your first image is smaller. Downsizing second image..."
                pic2 = pic2.resize(picsize1) # resizes 2nd image to match 1st
                time.sleep(2)
                print "Image 2 downsized. Image sizes now match."
                print ""
            if smaller == size2:
                print ""
                print "Your second image is smaller. Downsizing first image..."
                pic1 = pic1.resize(picsize2) # resizes 1st image to match 2nd
```

AN IMPROVED IMAGE RECOGNITION SOFTWARE SYSTEM
FOR IMPLEMENTATION IN BOMB-DISPOSAL ROBOTICS

```
        time.sleep(2)
        print "Image 1 downsized.  Image sizes now match."
        print ""

    if yesno == "n":
        print ""
        print "Note:  This setting will compare your smaller image"
        print "      with an equally-sized portion of your larger"
        print "      image."
        print ""

    time.sleep(4)
    pic1.show()
    pic2.show()

# The above nested IF code compares the size of the arrays (the logical
# representation of the images).  If they are not identical, the larger
# image is resized to match the smallest.  The user has the option of
# whether or not to resize; if the user selects no, then a portion of
# the larger image equal to the size of the smaller image (from the top
# left hand pixel extending diagonally bottom-right) is compared.  The
# accuracy and success of the RESIZE method is documented elsewhere.

# ARRAY COMPARISON CODE

print "Comparing now..."
print ""
time.sleep(2)
imageinfo = list(pic1.getdata())    # Re-renders the altered image(s) into
imageinfo2 = list(pic2.getdata())   # a pair of arrays.

# This code compares each pixel value to its counterpart in the 2nd
# picture.  If the values are identical, then "count" is incremented
# by 1.

for a,b in zip(imageinfo, imageinfo2):

    if a == b:

        count = count + 1

    else:

        count = count + 0

# ACCURACY GENERATION CODE

print "Arrays compared.  Generating percentage of similarity..."
time.sleep(1)
print ""

# The following 3 lines set var SIZE to equal the length of the arrays
# (both are now identical).  The % accuracy is worked out by taking
# variable COUNT, dividing its contents by the actual number of pixels
# (or array elements) and multiplying by 100.  Finally, the result is
# rendered into a string format as PYTHON cannot concatenate strings
# and integers using the PRINT command.

size = float(len(imageinfo))
percent = float((count / size)*100)
percentStr = str(percent)
print "These two images are " + percentStr + "% identical."
print ""
# print count    # Optional
# pic1.show()    # Shows the 1st image (optional)
# pic2.show()    # Shows the 2nd image (optional)
print len(imageinfo)
```

Version 2: Greyscale version with E.M. facility

Note: The code is identical up to this point.

```
# ARRAY COMPARISON CODE

print "Comparing now..."
print ""
time.sleep(2)
imageinfo = list(pic1.getdata()) # Re-renders the altered image(s) into
imageinfo2 = list(pic2.getdata()) # a pair of arrays.

# The following code compares each pixel value in the 1st image to the
# corresponding value in the 2nd image. If the values are within 10% of
# each other, either positively or negatively, then the counter "count"
# is incremented by 1 and the percentage of accuracy (calculated below
# this section) is calculated accordingly.

for a,b in zip(imageinfo, imageinfo2):
    if ((a >= (b * 0.7)) & (a <= (b * 1.3))) | ((b >= (a * 0.7)) & (b <= (a *
1.3))):
        count = count + 1
    else:
        count = count + 0

pic1.show()
pic2.show()

# ACCURACY GENERATION CODE

print "Arrays compared. Generating percentage of similarity..."
time.sleep(1)
print ""

# The following 3 lines set var SIZE to equal the length of the arrays
# (both are now identical). The % accuracy is worked out by taking
# variable COUNT, dividing its contents by the actual number of pixels
# (or array elements) and multiplying by 100. Finally, the result is
# rendered into a string format as PYTHON cannot concatenate strings
# and integers using the PRINT command.

size = float(len(imageinfo))
percent = float((count / size)*100)
percentStr = str(percent)
print "These two images are " + percentStr + "% identical."
print ""
# print count # Optional
# pic1.show() # Shows the 1st image (optional)
# pic2.show() # Shows the 2nd image (optional)
print len(imageinfo)
```

Please note: The difference between greyscale and colour is that in line 10, the part-line ".convert("L")" is removed.

Version 4: Colour version with E.M. facility

```
def convert(image1, image2):

    # PACKAGE IMPORT AND MODULE VARIABLE SETUP

    import Image, ImageFilter, sys, time
    count = float(0) # var counts identical pixels

    # IMAGE-ARRAY CONVERSION FOR IMAGE 1

    pic1 = Image.open(image1) # img var pic1 = param1(image1)
    imageinfo = list(pic1.getdata()) # pic1 converted to array of values
    img1str = str(imageinfo) # img1str = str equivalent of array for comparison

    # print "Image 1: " + img1str # display array contents (optional)
    time.sleep(1) # system wait command to allow user to read screen
    print ""

    # IMAGE-ARRAY CONVERSION FOR IMAGE 2

    pic2 = Image.open(image2)
    imageinfo2 = list(pic2.getdata())
    img2str = str(imageinfo2)

    # print "Image 2: " + img2str # by now image2 is in a second array
    time.sleep(1)
    print ""

    # DEPENDENT VARIABLE SETUP

    size1 = float(len(imageinfo)) # var size1 = no of elements in imageinfo1
    size2 = float(len(imageinfo2)) # var size2 = no of elements in imageinfo2

    picsize1 = pic1.size # var picsize1 = actual pic size (as tuple)
    picsize2 = pic2.size # var picsize2 = actual pic size (as tuple)

    # CODE TO RESIZE IMAGE

    if size1 < size2:
        smaller = size1
    elif size1 > size2: # These 4 lines set var "smaller" to smallest array
        smaller = size2
    if size1 != size2: # If images differ in size, execute the following...
        print ""
        print "WARNING!"
        print "-----"
        print ""
        print "These two images are not the same size."
        yesno = raw_input("Would you like to resize your images? (y/n)")
        if yesno == "y":
            if smaller == size1:
                print ""
                print "Your first image is smaller. Downsizing second image..."
                pic2 = pic2.resize(picsize1) # resizes 2nd image to match 1st
                time.sleep(2)
                print "Image 2 downsized. Image sizes now match."
                print ""
            if smaller == size2:
                print ""
                print "Your second image is smaller. Downsizing first image..."
                pic1 = pic1.resize(picsize2) # resizes 1st image to match 2nd
                time.sleep(2)
                print "Image 1 downsized. Image sizes now match."
                print ""

        if yesno == "n":
            print ""
            print "Note: This setting will compare your smaller image"
            print " with an equally-sized portion of your larger"
            print " image."
            print ""
```

AN IMPROVED IMAGE RECOGNITION SOFTWARE SYSTEM
FOR IMPLEMENTATION IN BOMB-DISPOSAL ROBOTICS

```
time.sleep(4)

# The above nested IF code compares the size of the arrays (the logical
# representation of the images). If they are not identical, the larger
# image is resized to match the smallest. The user has the option of
# whether or not to resize; if the user selects no, then a portion of
# the larger image equal to the size of the smaller image (from the top
# left hand pixel extending diagonally bottom-right) is compared. The
# accuracy and success of the RESIZE method is documented elsewhere.

# ARRAY COMPARISON CODE

print "Comparing now..."
print ""
time.sleep(2)
imageinfo = list(pic1.getdata()) # Re-renders the altered image(s) into
imageinfo2 = list(pic2.getdata()) # a pair of arrays.

for a,b in zip(imageinfo, imageinfo2):

    if ((a >= (b * 0.9)) & (a <= (b * 1.1))) | ((b >= (a * 0.9)) & (b <= (a *
1.1))):

        count = count + 1

    else:

        count = count + 0

# ACCURACY GENERATION CODE

print "Arrays compared. Generating percentage of similarity..."
time.sleep(1)
print ""

# The following 3 lines set var SIZE to equal the length of the arrays
# (both are now identical). The % accuracy is worked out by taking
# variable COUNT, dividing its contents by the actual number of pixels
# (or array elements) and multiplying by 100. Finally, the result is
# rendered into a string format as PYTHON cannot concatenate strings
# and integers using the PRINT command.

size = float(len(imageinfo))
percent = float((count / size)*100)
percentStr = str(percent)
print "These two images are " + percentStr + "% identical."
print ""
# print count # Optional
# pic1.show() # Shows the 1st image (optional)
# pic2.show() # Shows the 2nd image (optional)
```

END OF DOCUMENT