A NEAR REAL TIME PHOTOGRAMMETRIC PC BASED SYSTEM TO STUDY REGIONAL BODY SURFACE MOTIONS OF HUMAN BEINGS DURING RESPIRATION

BY

BARBARA ANNE GUTSCHOW
B.Sc. (SURVEY) ENGINEERING
UNIVERSITY OF CAPE TOWN

Submitted to the University of Cape Town in full fulfillment of the requirements of the degree of Master of Science in Engineering

SEPTEMBER 1990

ABSTRACT

The purpose of this thesis was to develop a near real time photogrammetric PC based system to study the regional body surface motions of human beings during respiration.

By being able to measure the body surface motion of the human torso during breathing it is possible to evaluate the respiratory muscle functions, i.e. the functions of the diaphragm and intercostal muscles, the only skeletal muscles that are essential to life and which enable us to breathe.

Previous studies in this field over the past years have successfully employed stereophotogrammetric analysis. The traditional photogrammetric method however was very time consuming in that it involved the taking of stereo pictures with 35mm cameras, developing the negatives and measuring them in a stereocomparator. The time factor involved severely restricted the research to physiologic studies.

For the study to be expanded so that it could be used as a diagnostic tool in the hospital ward environment the time to obtain the necessary results had to be considerably reduced and simplified. A near real time photogrammetric PC based system had to be designed to replace the traditional photogrammetric method.

A near real time system was developed for capturing dynamic pictures of the breathing cycle of the patients using a pair of video cameras, a two channel split screen vision mixer and a video cassette recorder with digital storage, all mounted on a custom built hospital cot for easy use in the hospital ward.

The images to be measured are selected by viewing the split screen in dynamic stereoscopic mode and selected images are captured on an IBM personal computer equipped with an image processing card. A computing package called TAG has been designed for easy use by non-photogrammetric personnel to evaluate stored images. TAG guides the user through the different processing stages from connecting the video machine to the IBM computer, transferring the selected images, measuring control and object points on the images, and processing the data.

TAG allows for graphical and/or visual output of the results, with the movement of the torso between the extreme breath in and breath out being depicted as vectors superimposed on the breath out image of the patient. The vectors, when viewed in stereoscopic mode, clearly show the movement of the patient's torso during respiration. A more graphical output of the data on a HP plotter is also possible with the plotting of XY and Z displacement vectors and stereo pseudo vectors.

The video process has many advantages over the photographic process. CCD camera settings can be adjusted to obtain good video images as the captured images are immediately visible on the monitor. A whole series of breathing cycles is captured on video tape and processed at a later date, whereas a normal 35mm camera can only capture an instant in time. The video tape serves as a permanent record for that patient and can be compared to later video sessions.

The processing of the video session of a patient using TAG no longer takes up a great deal of time and only takes minutes to process. The system fulfills the requirements that were set for it, in that it is time efficient and easy to use.

The research has opened up a door to a much wider field - the power of three dimensional vision. This is not the type of "three dimensional view" that is constructed on a two dimensional computer monitor. Reconstructing truly three dimensional vision by viewing the images from the two CCD cameras in stereo, allows the mind to perceive the breathing movements of a patient as if they were actually occurring at that moment.

To be able to view movement as if it were occurring now in true three dimensions and at any video play back speed, opens up a whole new dimension to the study of motion. Simple observation of the motion may reveal more than any analysis possibly could.

ACKNOWLEDGMENTS

I would like to thank Professor L.P. Adams for his supervision and enthusiasm. Without him I would not have embarked into the study of photogrammetry. His interest and enthusiasm for photogrammetry especially in its role in the medical world made the subject come alive for me. I would also like to thank Miss Ann Tregidga without whose support and knowledge in the photogrammetric field I would not have progressed with the research as fast as I did. I would also like to thank Mr. Doug Kirby for constructing the control frames and other items needed in the research work and all the staff and students of the Department of Surveying at the University of Cape Town who created such a friendly and helpful atmosphere to work in. I would like to thank the Medical Research Council and Dr.S.Wynchank. Without their very generous bursary none of this research would have been possible for me. I would also like to thank Dr. Wynchank for his able and willing supervision of my research, especially since he has a hectic schedule. I would like to thank the Red Cross War Memorial Children's Hospital and Dr. Max Klein who started the original research with Prof. Adams and whose enthusiasm for the research and for his young patients was unbounding.

I am very grateful to have been able to do this research work, as it has a very human aspect which many other engineering projects don't have. It has been an opportunity for which I will always be grateful.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

"Biological form and its response to both external and internal forces has been and is one of the most engaging subjects in the history of human thought. " (Sheffer and Herron 1989)

1. INTRODUCTION

1.1 THE NEED FOR A NEAR REAL TIME PHOTOGRAMMETRIC PC BASED SYSTEM FOR THE STUDY OF REGIONAL BODY SURFACE MOTIONS DURING RESPIRATION

Respiratory muscles, the intercostals and the diaphragm, are the only skeletal muscles that are essential to life. As the bellows in the smithy will not function without the "muscle" of the apprentice pumping them, so the lungs will not function without the respiratory muscles.

The lungs themselves are "flaccid bags" with very little mobility of their own. For respiration to take place the pleural sac containing the lungs is attached to the rib cage and the diaphragm. The thoracic cavity is enlarged by the intercostal muscles pulling the ribcage upward and outward and by the diaphragm pulling down. As the thoracic cavity enlarges the space surrounding the lungs is also enlarged and the lungs'

internal pressures drop. The air moves from the high pressure volume outside the body to the low pressure volume inside the lungs. As the muscles relax, the thoracic cavity is reduced in size and the lungs "collapse" forcing out the gas within them. Thus it can be seen that weakness or disability in these muscles can result in respiratory failure.

By analysing regional body surface motions during breathing it is possible to study the functions of the principal respiratory muscles - the intercostals and the diaphragm.

To study the functions of respiratory muscles is a medical issue, the problem of measuring regional body surface motions can readily be answered by photogrammetry. Medical photogrammetry, or a preferred medical term - biostereometrics, can provide measurements of high precision which are needed in the study of the respiratory muscles.

Medical personnel are already fully engaged with their medical workload and with the acquisition of new knowledge and technology in their own field; it is important therefore how and in what manner measurement information for example is obtained. For respiratory motion studies, criteria that are important are firstly that medical staff must obtain the information on the regional body surface motions for analysis without having to acquire indepth knowledge of how this information is arrived at and secondly that the time involved

in obtaining this information is rapid. The information must be available for research work and also for diagnostic purposes.

Only if these criteria are met is it sensible for the photogrammetrist to design a system for use by the medical fraternity to study the regional body surface motions during respiration; a system which must be "turn key" and which is both user friendly and time efficient, and thus can be useful as a diagnostic tool.

1.2 ALTERNATE METHODS USED TO STUDY RESPIRATORY MOTION

In the medical field respiratory motion is conventionally studied using methods that measure torso circumference or diameter or linear motion and volume displacements. These methods do not easily distinguish between the two sides of the body and thus can only be used to study the movements of the diaphragm and the intercostal muscles in general.

In the past photogrammetric methods to study respiratory motion mainly used aspects of raster photogrammetry or Moiré topography. In these methods patterns of light are projected onto the body surface. By measuring these patterns recorded on photographs or video tape it is possible to reconstruct the surface of the torso.

These methods are more appropriate for the mapping of the chest's surface at any stage during a respiratory cycle. They cannot describe the movement of discrete points on the body surface during a respiratory cycle.

Normally movements of specific points and regions of interest are required for the elucidation of precise structure and function relationships - for the physiologic studies of the respiratory motion of human beings.

More recently in 1986, a stereophotogrammetric method developed at the University of Cape Town, using conventional cameras, has been used to measure the movement of discrete points on the body surface.

1.3 THE TRADITIONAL CAPE TOWN PHOTOGRAMMETRIC METHOD TO STUDY REGIONAL BODY SURFACE MOTIONS OF A HUMAN BEING

The method developed in Cape Town used traditional photogrammetry to study the regional body surface motion of human beings during respiration.

Discrete points of interest were targeted on the body surface. Using conventional cameras, stereo-photographic pairs were taken from the front of the subject reclining in a reference frame. Image pairs at the extremes of the breath-in and the

breath-out epochs were selected. The co-ordinates of image points were recorded by means of a stereocomparator linked to a computer.

Results were shown as computer generated contours and tridimensional plots. The motions of the discrete points on the body surface were also plotted as precise spatial motion vectors. The vectors were computed for each point, derived from the co-ordinates of the targets at the extremes of the breathing cycle. The vectors were presented numerically or as scaled arrows superimposed on the stereo photographs of the body surface which, by closely relating displacement to structure, facilitated data interpretation.

The Cape Town method, using traditional photogrammetry, was extremely labour intensive and time consuming and severely limited potential applications of the method.

1.4 THE NEAR REAL TIME PHOTOGRAMMETRIC SYSTEM TO STUDY REGIONAL BODY SURFACE MOTIONS DURING RESPIRATION

The maxim "Time is Money" affects the medical and photogrammetric worlds alike. Equipment and time cost money. Time in the medical world is not only critical in the monetary sense, but also critical for the health of the patients. With the advent of near real time photogrammetry a whole new

dimension has been added to photogrammetry, the dimension of speed.

The old conventional cameras are replaced by video cameras. Photographic film is replaced by video tape. Image display occurs through video monitors. The stereo comparator linked to a computer is replaced by image frame grabber hardware installed in a PC computer and digitising software.

With the new equipment the time consuming development of film has been done away with. Images received by the video cameras are immediately visible and bad imagery can be corrected. The whole series of events can be recorded onto tape whereas with a 35mm camera only a single moment in time can be captured.

Any images not required can be erased and the video tape reused. Instead of placing negatives in a stereocomparator for measurement, the images on the video tape are transferred to the PC computer via the frame grabber card, with no loss of resolution. All these changes constitute a saving in both time and money.

The remaining problem is the software to digitise the images of the discrete points on the body surface. As near real time photogrammetry is a fairly recent innovation, no software was available to undertake this photogrammetric task.

A easy to use turn key system had to be developed capable of operation by a non photogrammetrist. The system not only had to consist of the software for digitising but also of an equipment rig for easy set up of the video cameras and recording of stereo images.

The scope of this thesis is to present the development of a user friendly, time efficient, turn key system to be used by non-photogrammetrists to study the regional body surface motions of human beings in the hospital ward environment.

The feasibility in the development of the system not only lies in the system facilitating the measure of regional body surface motion but also in the visual impact of the procedure. The power of stereoscopic vision of recorded motion at varying speeds is as yet hardly realized, but may prove to be the most powerful tool of the whole system.

It must be borne in mind that this research is in the realm of the medical world. If the system is to be reproduced and used by the medical profession in general, it must be understandable to the medical user and thus some of the topics are approached and presented in a more simplistic fashion than is usual in photogrammetric research, directed specifically towards the photogrammetrist.

2. HISTORICAL DEVELOPMENTS IN THE STUDY OF RESPIRATORY MOTION BEFORE AND AROUND THE DEVELOPMENT OF THE CAPE TOWN METHOD

In the medical world respiratory motions are studied by:

1. the relationships between linear motion and volume displacement (Konno & Mead 1967)
2. methods sensitive to changes in torso circumference (Wade 1953).

2.1 LINEAR MOTION AND VOLUME DISPLACEMENT TO STUDY RESPIRATORY MOTION

Konno and Mead (1967) studied the respiratory movements of human beings by analysing volume changes in different parts of the torso wall (the rib cage and abdominal wall) and the anteriorposterior motion of the anterior wall of the rib cage and abdomen.

To study the relationship between motion and volume the subject had to perform different breathing exercises.

The transducers to measure motion were attached to the soft tissue of the chest and abdomen, see figure 2.1 below. As the tissue is distorted by the attachment of the transducer, this method could lead to erroneous results. As the study has been developed using standing subjects, it does not lend itself to the study of chronically ill patients with pulmonary diseases.

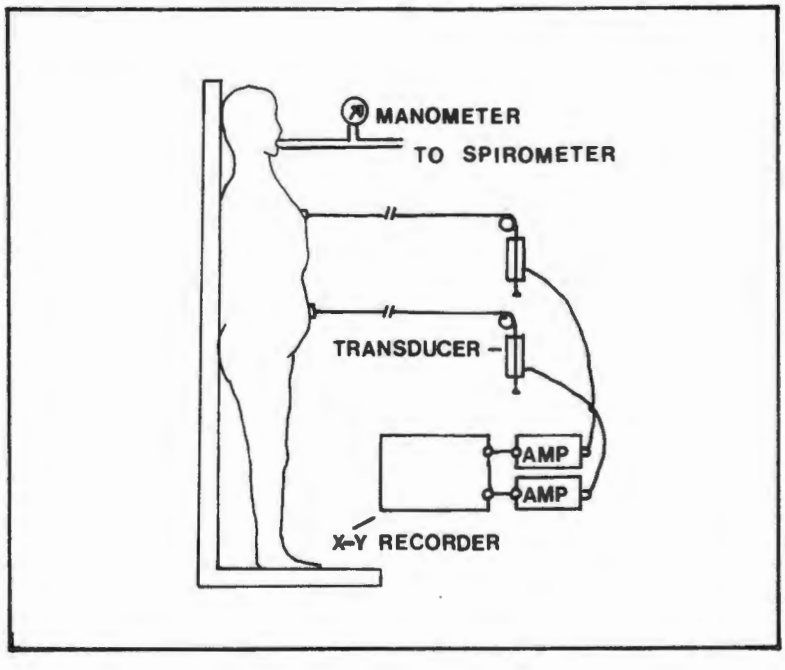

FIGURE 2.1 SEPARATE VOLUME CHANGES OF RIB CAGE AND ABDOMEN (from Konno et al 1967)

2.2 CHANGES IN TORSO CIRCUMFERENCE TO STUDY RESPIRATORY MOTION

Wade (1953) used methods sensitive to changes in torso circumference to study the relationship between movements of the diaphragm and the chest circumference. Transducers were used to measure the chest circumference, see figure 2.2 below.

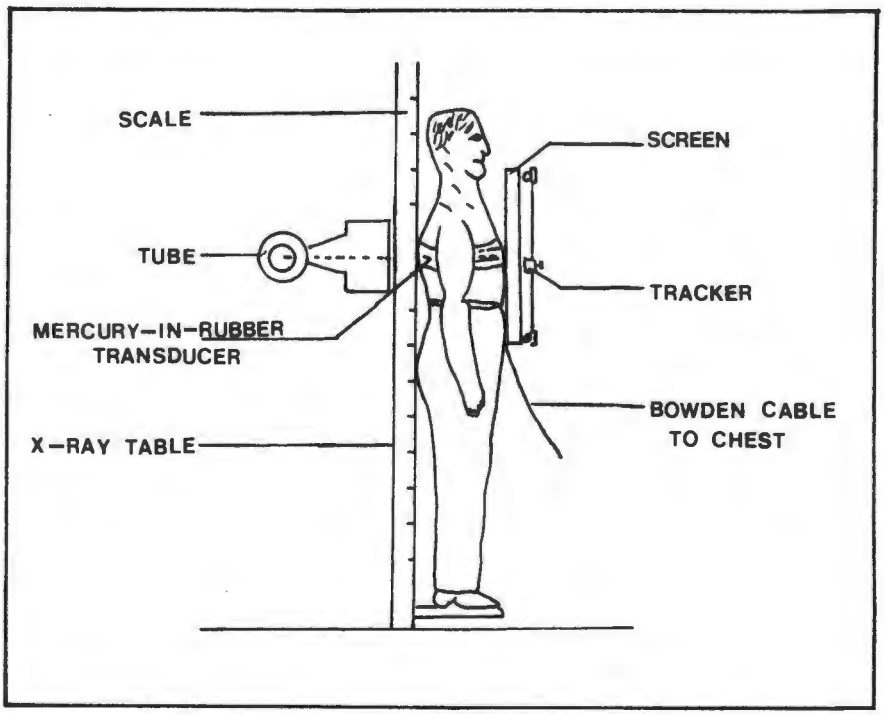

FIGURE 2.2 METHOD OF TRACKING DIAPHRAGMATIC MOVEMENT AND RECORDING THE RISE AND FALL OF THE CHEST AND CHANGES IN CHEST CIRCUMFERENCE
(from Wade 1954)

This method has the same disadvantages as the previous method.

As both these methods do not distinguish between the two sides of the body these methods can only essentially compare gross rib cage displacement (intercostal muscle) with gross abdominal (diaphragm) displacement. (Klein 1989).

2.3 THE THEORY OF RASTER PHOTOGRAMMETRY

In these methods light patterns/rasters made up of lines or grids of known geometry are projected onto the body surface, see figure 2.3 below. By measuring these patterns recorded on photographic film or video tape it is possible to reconstruct the surface of the human torso.

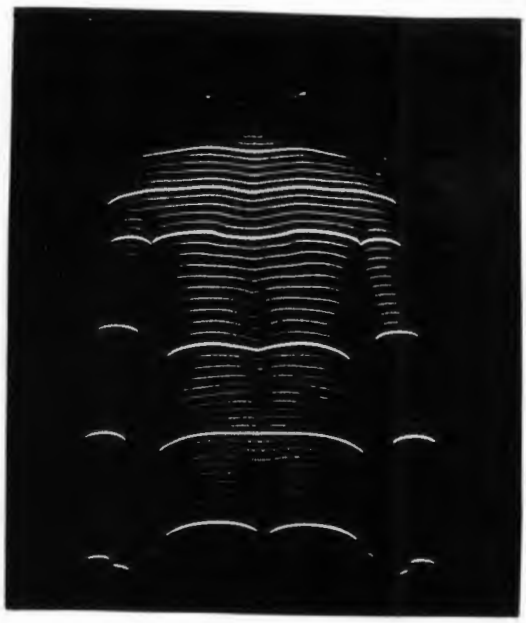

FIGURE 2.3 LINE RASTER STEREOGRAPH OF PATIENT'S BACK
(from Hierholzer et al 1989)

In raster photogrammetry a single photograph is taken of one aspect of an object. But since we can not perceive depth with only one eye we can not determine the third dimension with only one photograph. As the reversing of the light paths does not affect their geometry, it is possible to replace one of the two cameras with a projector with a raster diapositive, which projects the pattern of light onto the object, see figure 2.4 below.

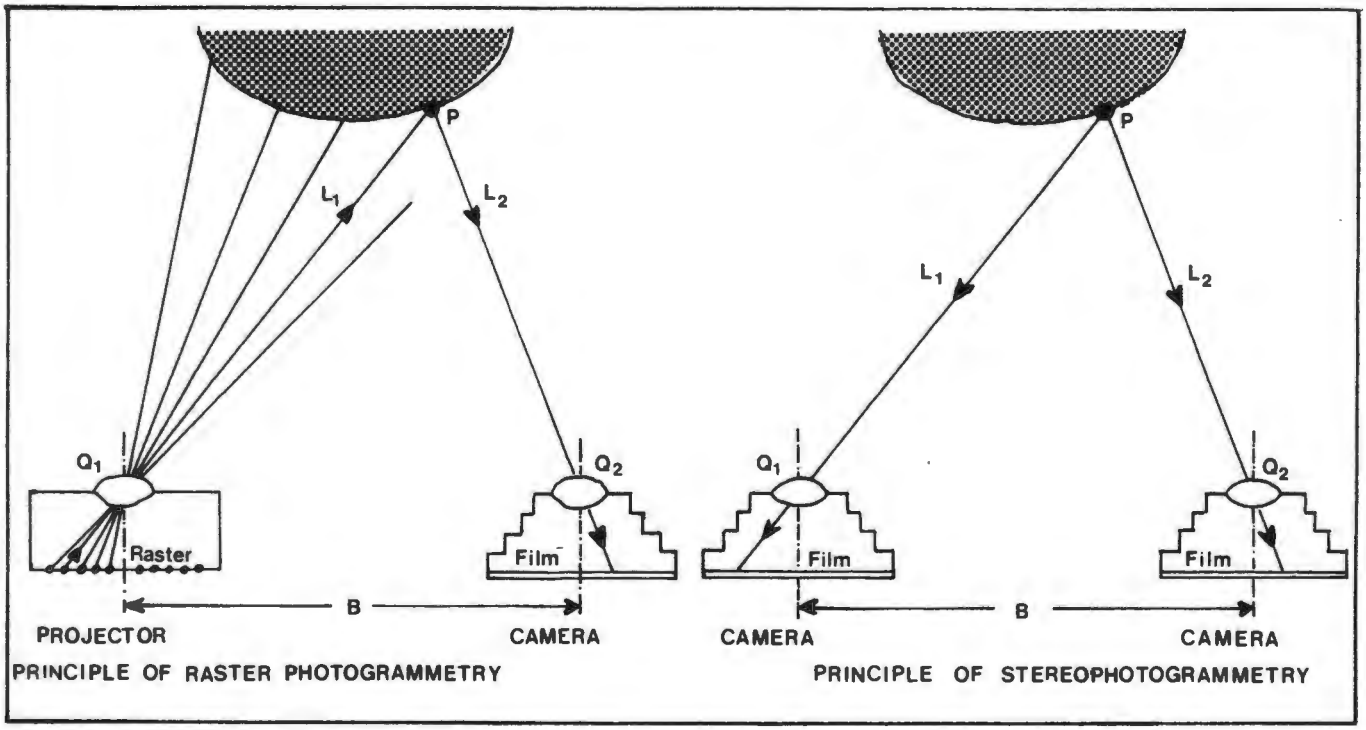

FIGURE 2.4 PRINCIPLES OF RASTER AND STEREOPHOTOGRAMMETRY (from Hierholzer et al 1989)

Thus instead of light from the object falling on the negative of a second camera, the light is reversed and the pattern projected onto the object is captured by only one camera. Thus both "camera" set ups (one camera and one projector) are captured on a single negative.

Referring back to figure 2.2, the human form that is photographed is clearly visible due to the light distortions of the pattern. These distortions hold the depth information of the subject - the human torso. As a "raster" photograph is the equivalent of a stereo pair of photographs, analytical techniques in photogrammetry, with modification, can be used to determine three dimensional information about the subject's surface, in this case a mapping of the human torso.

2.4 USE OF A CROSS RASTER IN RASTER PHOTOGRAMMETRY AS DEVELOPED BY KOVATS (1974)

Kovats (1974) was the first to use raster photogrammetry in the study of respiratory motion using a serial photo-geometric method. A cross raster, a regular square grid, was projected onto the body surface, see figure 2.5 below.

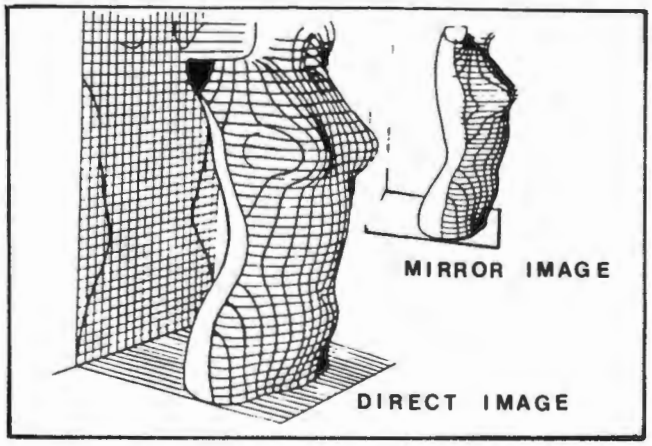

FIGURE 2.5 DIRECT IMAGE D AND MIRROR IMAGE M BOTH CAPTURED ON THE SAME EXPOSURE (from Kovats 1974)

The distortion of the square grids is captured on a series of photographs or video film. As is apparent, the left and right side of the torso are captured on one photograph as the far side of the torso with its raster pattern is projected onto the photographic film via a mirror.

A whole series of photographs or video frames is taken of the torso during respiratory motion. By measuring the whole series it is possible to measure the changes in the chest wall during breathing.

2.5 USE OF A LINE RASTER IN RASTER PHOTOGRAMMETRY AS DEVELOPED BY GOURLAY et al AND MORGAN et al (1984)

A similar method, using optical contouring, was developed to map the size and shape of the thoracoabdominal wall and the change in its shape with breathing (Gourlay et al 1984, Morgan et al 1984).

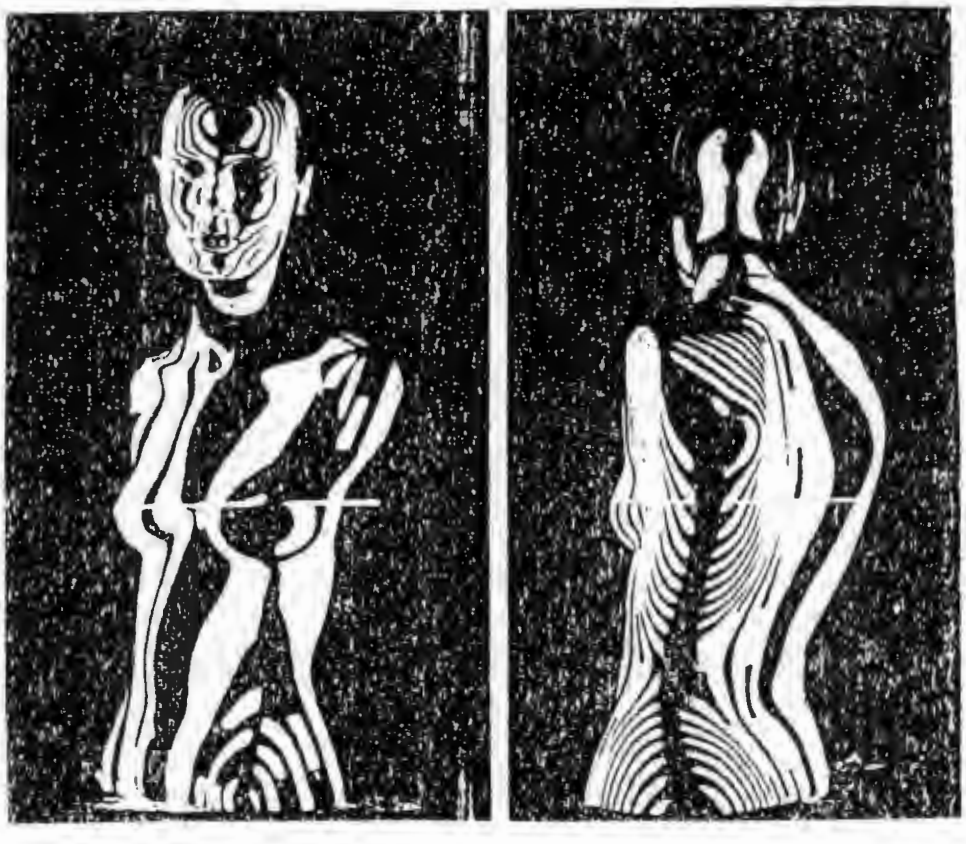

FIGURE 2.6 EXAMPLE OF OPTICAL CONTOURING; FRONT AND BACK VIEWS OF MANNIKIN (from Gourlay 1984)

A line raster, a fixed pattern composed of vertical lines, is projected onto the patient in a control framework, see figure 2.6 above.

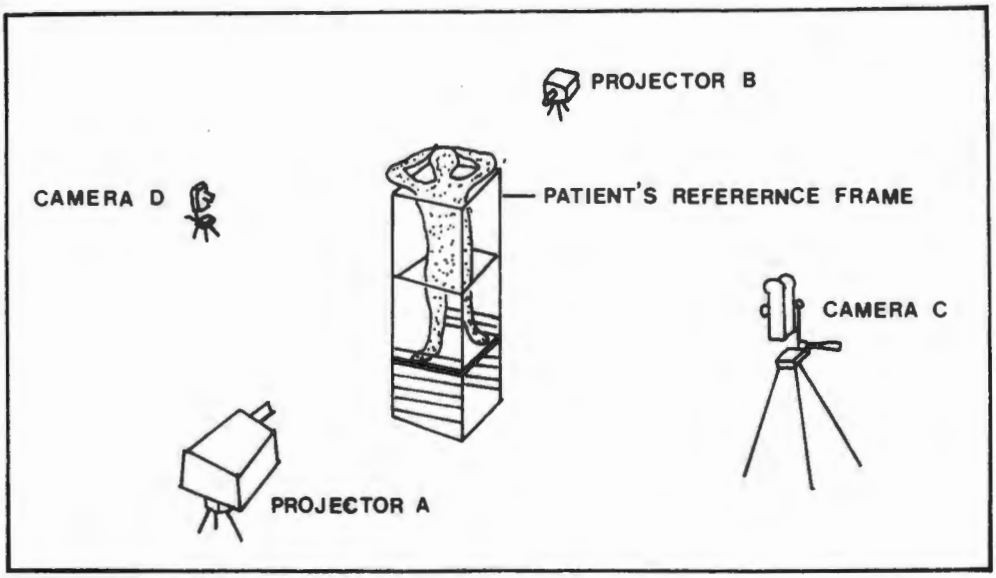

FIGURE 2.7 CAMERA AND LIGHT SET UP TO PHOTOGRAPH SUBJECT (from Denison 1982)

Two cameras are used to capture both sides of the torso, instead of the mirror method, using one camera, as used by Kovats; see figure 2.7 above.

By relating each contour of light on the torso to the vertical lines in the raster and by measurement on the recorded image, it is possible to determine three dimensional co-ordinates for the torso surface during breathing.

The studies so far discussed all relied on human subjects in a standing position, which is not feasible for chronically ill patients. It was Morgan,Gourlay and Denison (1984), who undertook studies on the movement on the chest wall of patients in a recumbent position. They showed that an overhead camera can capture 97% of the respired motion and that studying patients in a recumbent positions would give accurate results, as has been achieved using the traditional Cape Town method.

2.6 USE OF AN AUTOMATIC LIGHT SECTIONING SYSTEM IN RASTER PHOTOGRAMMETRY AS DEVELOPED BY SAUMAREZ (1986)

The problem encountered with raster photogrammetry and its possible automation is the large amount of information that has to be obtained from raster photography with up to 500 points to be digitised. Saumarez (1986), by using an automatic light sectioning system, overcomes this problem by continuously measuring the movements of the torso during breathing.

This is done by illuminating the body with a raster of vertical lines. A video camera operating at high speed scans the body for these planes of light and digitally records the angles of perception of the camera onto magnetic tape, see figure 2.8 below.

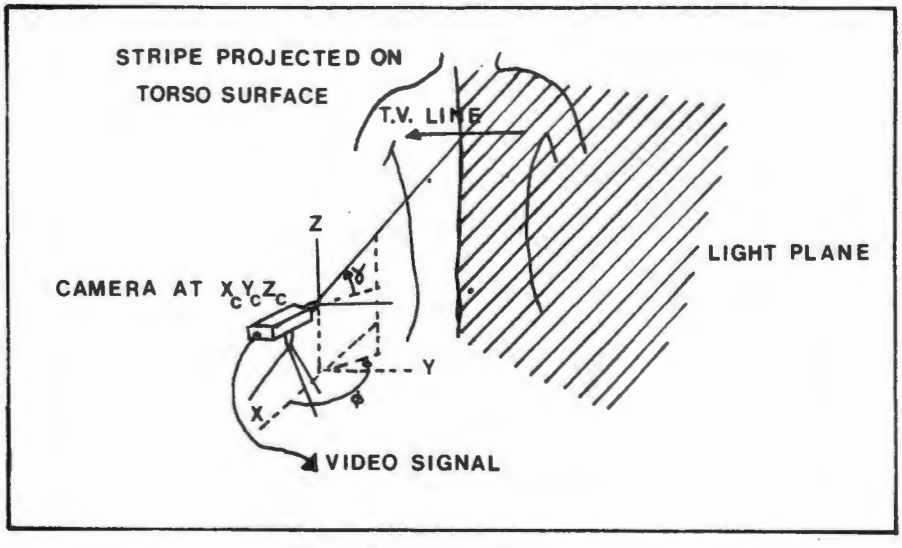

FIGURE 2.8 VIDEO SCAN CAMERA SCANNING SUBJECT
(from Saumarez 1986)

The geometry of the torso during respiration is then reconstructed using the geometry of the vertical lines and the recorded information

Although all the methods so far investigated determine the three dimensional co-ordinates for the chest's surface during respiratory motion, they are more suited to mapping the torso surface. They do not lend themselves readily to physiological studies. These studies require the movement of specific points or regions of interest on the surface to be determined and related to their associated structures. (Adams and Klein 1986).

The traditional Cape Town stereo photogrammetric method developed (Adams and Klein 1986) does fulfill the requirements for physiological studies and presents the movement of specific points in true three dimensional form by depicting the movement as spatial vectors superimposed on stereo photographs.

3. THE THEORY OF STEREO PHOTOGRAMMETRY

Stereo photogrammetry is defined as the technique of measurement of a pair of stereo photographs for the determination of primarily geometrical data such as size, position and shape of the object.

The lay person is understandably hesitant to approach an unfamiliar topic. The subject of stereo photogrammetry often causes this reaction in individuals, without them realising that stereo photogrammetry is based essentially on stereoscopic human vision, see fig 3.1 below.

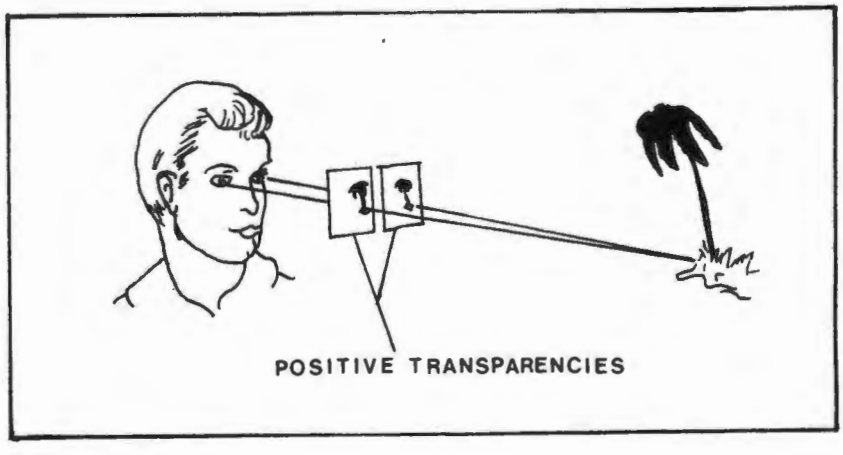

FIGURE 3.1 STEREOSCOPIC HUMAN VISION
(from Mikhail 1989)

The disparity in the human eyesight allows for depth perception, see figure 3.2 below. As angle øa is larger than angle øb, object a is to be perceived closer than object b.

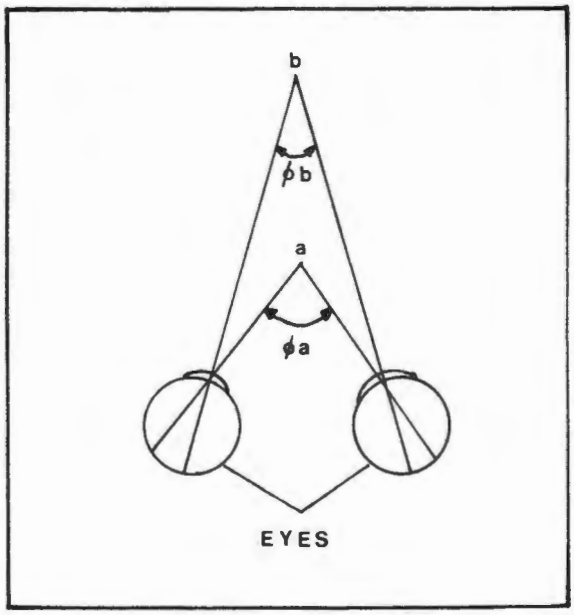

FIGURE 3.2 PARALLAX - ALLOWING FOR DEPTH PERCEPTION
(from Mikhail 1989)

This disparity is known as parallax and is either an angular or linear measure. It is used in photogrammetry to determine the third dimension.

Linear parallax consists of x parallax (Px) and y parallax (Py) where:

$$Px = x - x'$$

$$Py = y - y'$$

where:

x, y - are image co-ordinates on the left photograph

x',y' - are the co-ordinates of the corresponding image of the same object point on the right image

Px - is the x-parallax, which is in the general direction of the base

Py - is the y-parallax in the perpendicular direction

There are several mathematical models which can be used for the determination of three dimensional co-ordinates of an object. A suitable method is described by Adams (1981) which lends itself well to close range photogrammetry and gives good results.

4. THE PROJECTIVE TRANSFORMATION TO MEASURE IMAGE CO-ORDINATES

The calculation of the position of object points in space expressed as tridimensional co-ordinates can be derived by measuring the two dimensional rectangular co-ordinates of common image points appearing on a stereopair of pictures. The space positions can be conveniently derived through the use of the mathematics of the projective transformation.

If it is required to analytically determine space co-ordinates of image points using a pair of pictures, then provided a minimum of six suitably distributed free net control points (points with known 3D co-ordinates) exists within the object space and are imaged in the picture plane, we can by the use of projective transformations and using homogeneous co-ordinates and collinearity equations calculate the three dimensional co-ordinates of common image points in terms of the free net control system. The theory and mathematical techniques employed in this somewhat complex projective geometry problem are discussed in detail in appendix 1. For ease of reference the appropriate formulae are summarised below.

Calculation of transformation parameters

For every image point "P" on a single picture the following projective relationship holds:-

$$Xi\ b11 + Yi\ b12 + Zi\ b13 + b14 - xi\ Xi\ b31 - xi\ Yi\ b32 - xi\ Zi\ b33 = xi$$

$$Xi\ b21 + Yi\ b22 + Zi\ b23 + b24 - yi\ Xi\ b31 - yi\ Yi\ b32 - yi\ Zi\ b33 = yi$$

where:

Xi, Yi, Zi are free net space co-ordinates of point Pi,
xi, yi are plane image co-ordinates of point Pi referred to an arbitrary origin,
bij are transformation parameters pertaining to a single camera.

Calculation of space co-ordinates using a stereopair of pictures

After the bij parameters of the individual cameras have been calculated the position in space in terms of a free net space system of common image points appearing on a pair of pictures can be derived using the following equations:

$$(xi\ b31 - b11)\ Xi + (xi\ b32 - b12)\ Yi + (xi\ b33 - b13)\ Zi + xi - b14 = 0$$

$$(yi\ b31 - b21)\ Xi + (yi\ b32 - b22)\ Yi + (yi\ b33 - b23)\ Zi + yi - b24 = 0$$

$$(\overline{xi}\ \overline{b31} - \overline{b11})\ Xi + (\overline{xi}\ \overline{b32} - \overline{b12})\ Yi + (\overline{xi}\ \overline{b33} - \overline{b13})\ Zi + \overline{xi} - \overline{b14} = 0$$

$$(\overline{yi}\ \overline{b31} - \overline{b21})\ Xi + (\overline{yi}\ \overline{b32} - \overline{b22})\ Yi + (\overline{yi}\ \overline{b33} - \overline{b23})\ Zi + \overline{yi} - \overline{b24} = 0$$

where the unbarred elements refer to the left hand picture and the barred elements to the right hand picture.

5. TRADITIONAL STEREO PHOTOGRAMMETRY IN THE CAPE TOWN STUDY OF REGIONAL BODY SURFACE MOTION DURING RESPIRATION IN HUMANS

In the traditional method developed at the University of Cape Town to study regional body surface motion during respiration in human beings, precise spatial motion vectors of an almost unlimited number of discrete points on the body surface of children as small as 2.5kg are determined, see figure 5.1 below (Adams & Klein 1986).

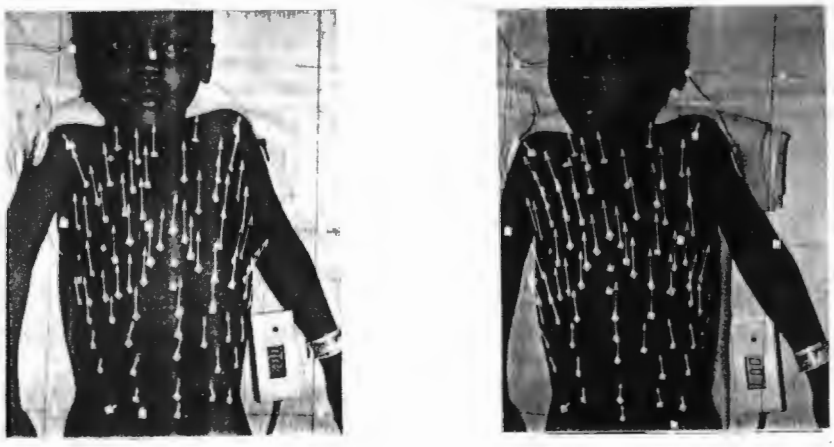

Figure 5.1 SPACE VECTORS SUPERIMPOSED PHOTOGRAPHICALLY ON A BREATH OUT STEREOSCOPIC PAIR

As the surface of the torso is nearly featureless, points of interest on the body were marked with white adhesive targets annotated with black crosses. The subject was then placed in one of the control frameworks (baby or child/adult frame), see figure 5.2 below.

FIGURE 5.2 BABY PHOTOGRAPHED IN CONTROL FRAME

Attached to the control framework were rigid supports to hold two motor driven 35mm cameras and a flash with a camera base length of 0.4m and located 1m above the lower grid of the control.

This satisfies the geometry of a well conditioned photogrammetric set up, where the base/height ratio is in the order of 1:5, see figure 5.3 below.

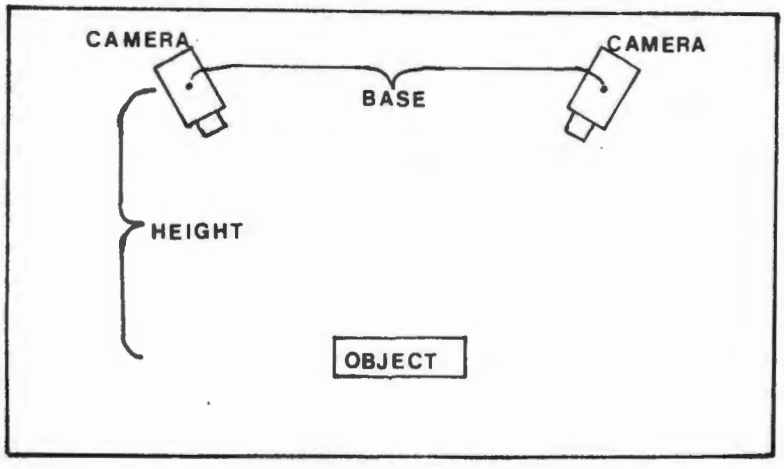

FIGURE 5.3 ILLUSTRATING BASE/HEIGHT RATIO

Stereo photographs were taken at the extremes of the breath-in and breath-out epochs. Once the photographs were developed, the stereo pairs were observed in stereo mode in a Steko 1828 stereo comparator. The three dimensional co-ordinates of object points, points on the patient's chest, were obtained using the projective transformation equations using the measured two dimensional photo co-ordinates of control and object points.

The residual uncertainties obtained from redundant control points showed that the error in spatial vectors seldom exceeded 0.5mm

Results were also shown as computer generated contours and tridimensional plots.

A pictorial demonstration of a patient with funnel chest deformity best shows the effect of the different plots, as can be seen in figures 5.4 through 5.8.

FIGURE 5.4 STEREOGRAM : BREATH IN EPOCH
(from Adams et al 1986)

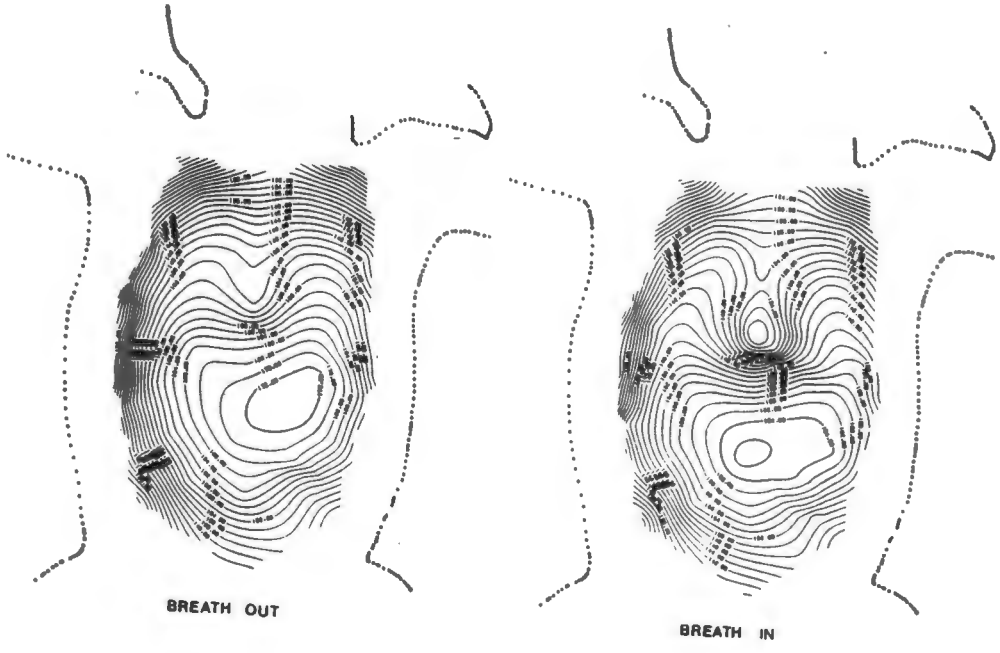

FIGURE 5.5 ABSOLUTE PLOTS : CONTOUR INTERVALS 2mm
(from Adams et al 1986)

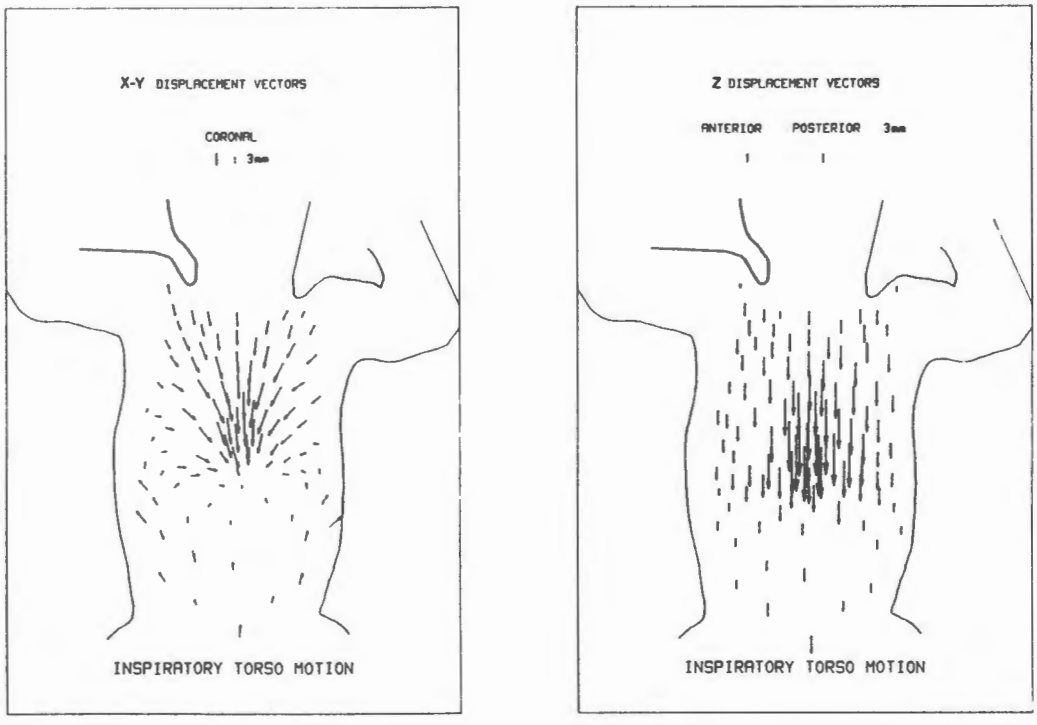

FIGURE 5.6 VECTOGRAPHS

(from Adams et al 1986)

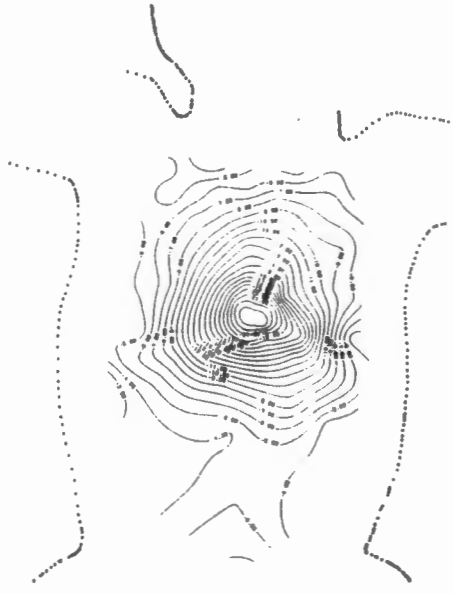

FIGURE 5.7 CONTOUR PLOT - INTERVAL 1mm

(from Adams et al 1986)

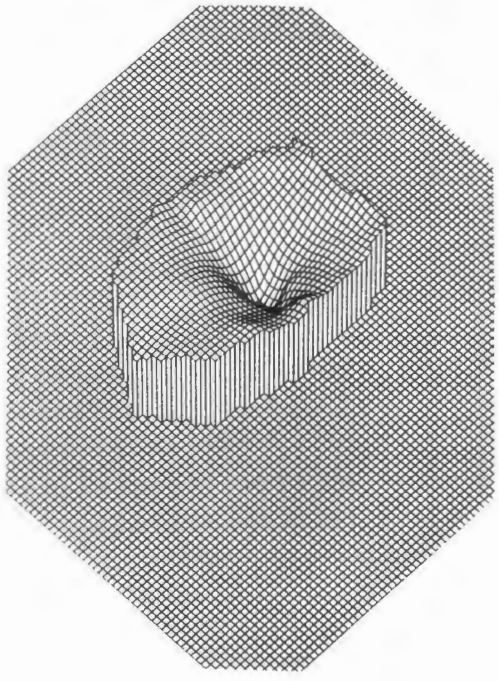

FIGURE 5.8 3-D VIEW - GRID INTERVAL 5.5 mm. Z AMPLIFIED * 5

With the traditional photogrammetric system it was possible to study the motions of specific regions of interest of the chest surface, thereby learning more about the function of the principal muscles of respiration - the intercostals and the diaphragm. But the traditional method is highly labour intensive and time consuming, with the developing process of the photographs alone taking up a full day. The work could only be done by people with photogrammetric experience. This has limited the field of study using traditional photogrammetry for research studies only.

Developments in computer hardware and image processing technology have made it possible to develop a near real time photogrammetric system that can be used in the hospital

environment by hospital personnel. In this context near real time refers to the obtaining of results in a relatively short time spanning only minutes. Thus it is logical to proceed with the development of the traditional system to a more automatic system making it a feasible tool not only for research work but also for diagnostic purposes.

6. THE EQUIPMENT SET UP FOR THE NEAR REAL TIME COMPUTERISED SYSTEM

The computerised system consists of the following equipment:

1. JVC video cassette recorder, Model HR-D700E
2. 2 JVC video tapes
3. Image mixer, AVICOM, Model HVB 100
4. 2 Ikegami B/W CCD cameras, Model ICD-290
5. 2 Computar TV zoom lenses, Model M6Z 1212, 12.5-75mm, F1.2
6. 20" CCTV picture monitor, Model PM-205A
7. 9" CCTV picture monitor, Model PM-930
8. 2 control frames with three dimensional co-ordinated control points
9. 1 lamp with 100 watt globe and dimmer
10. 1 matt black sheet
11. Customized hospital bed
12. 2 image processing cards, Matrox PIP-512 and Matrox PIP-1024
13. IBM Personal System/2 model 30 computer
14. Two Phillips RG3 monitors CM8833
15. Tektronix image printer 4693D
16. Star Nx-15 dot matrix printer
17. Hewlett Packard HP 7475A plotter

6.1 EQUIPMENT SET UP AT THE RED CROSS CHILDRENS' HOSPITAL

In designing the equipment set up at the Red Cross Children's Hospital the factors, that the system had to be both user friendly and time efficient, had to be considered. More importantly the user friendly factor does not only apply to the hospital personnel, who will operate the system, but also to the patients.

As patients suffering from respiratory ailments may be chronically ill, a special hospital bed was designed to serve as a photogrammetric rig, see figure 6.1 below.

FIGURE 6.1 THE COMPLETE RIG IN FRONT OF THE EQUIPMENT SET UP AT UCT

The rig can be wheeled to the patient, the patient transferred to the bed/rig for the video session to record the patient's respiratory motion. This process occurs with minimal discomfort to the patient and the patient remains recumbent throughout the session.

The rig differs from the normal bed in that a gantry is mounted above the bed which supports the cameras, a cupboard and a shelf are mounted below the bed to hold the equipment. The cupboard, depicted in figure 6.2 below, contains the JVC video recorder, the Avicom mixer, a plugboard and, the cameras and video tapes when not in use. The shelf is used to transport control frames.

FIGURE 6.2 CUPBOARD STORING EQUIPMENT - VCR, MIXER, etc.

6.1.1 THE DESCRIPTION OF THE CCD CAMERAS

The two Ikegami cameras are black and white CCD (charge-coupled devices) solid state matrix cameras and generally are recognised as the best type of video camera for photogrammetric work.

The "eye" of the CCD camera can be likened to an insect's multifaceted eye, with each facet being composed of a diode (light sensor). These diodes on a silicon wafer chip (10-20 micrometers thick) form the silicon imaging chip. The light energy falling on the sensors is transferred as a series of voltages, which are analogue quantities. The way the analogue signals are transported depends on the type of sensors used, either interline transfer (IL) or frame transfer (FT) sensors.

IL sensors use a two line-interfaced TV fields, which are read out alternately. One line is used for charge accumulation (charge generated from light from object) and one line for charge transportation (charge transportation to output mode - monitor or video tape). In FT sensors the same capacitors are used for charge accumulation and transportation. The IL sensor is used in most consumer cameras, such as the Ikegami cameras and has reached a high quality standard.

The pixels (picture elements - sensors) in a CCD camera do not have uniform proportions but scale differences between the vertical and horizontal axes. These scale differences had to be determined in order to measure an image accurately . The determination of the pixel size is discussed in chapter 7. The size of the Ikegami imaging chip is also not uniform with 500 horizontal and 580 vertical pixels making an image field of approximately 290000 pixels.

Through the use of image processing hardware and different processing techniques it is possible for the computer to convert the analogue to digital quantities and display these as an image on a CCTV (closed circuit television) monitor

The cameras have the advantage of being small, light weight and virtually maintenance free. The sensors are stable over time in position and sensitivity; that is the cameras do not produce any change in image capturing over long periods of time as the temperature of the cameras varies (El-Hakim 1986).

Two Computar TV zoom lenses are mounted on the CCD cameras, with the lenses having a variable focal distance (1m to infinity), zoom (focal distance 12.5-75mm) and aperture (f stop 1.2-22,C) settings. The general term camera hereafter refers to both the CCD camera and its zoom lens.

6.1.2 THE MOVEMENT CAPABILITIES OF THE CAMERAS AND CAMERA CABLE CONNECTIONS

The gantry that supports the cameras above the bed is not a fixed structure and can be raised and lowered. The gantry's height is changed by using the notches on the vertical bars, as shown in figure 6.3 below.

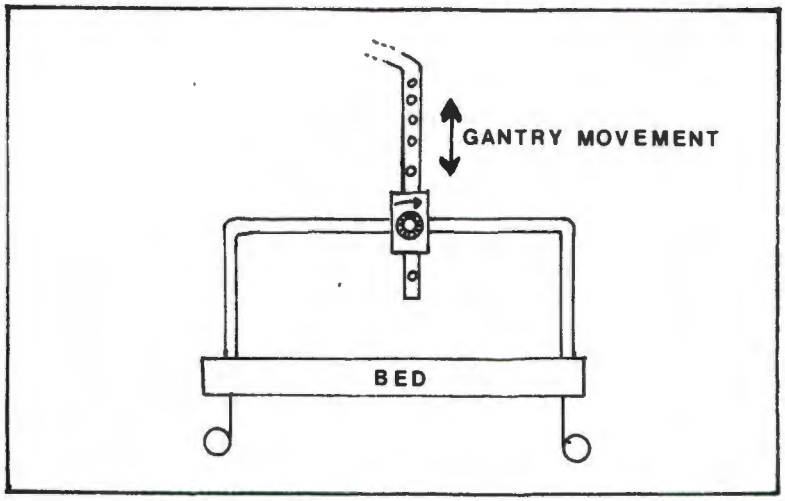

FIGURE 6.3 GANTRY MOVEMENT ABOVE BED

The gantry's height above the bed can vary approximately between 0.9m and 1.5m to allow for patients of different sizes. A bar, mounted on top of and perpendicular to the gantry (see fig. 6.6), holds a garbler rail (see fig. 6.5) above the patient's torso area.

The cameras are held aloft above the patient's torso by snug fitting camera holders, which are attached to garbler clips, which are clipped onto the garbler rail, see figure 6.4 and 6.5.

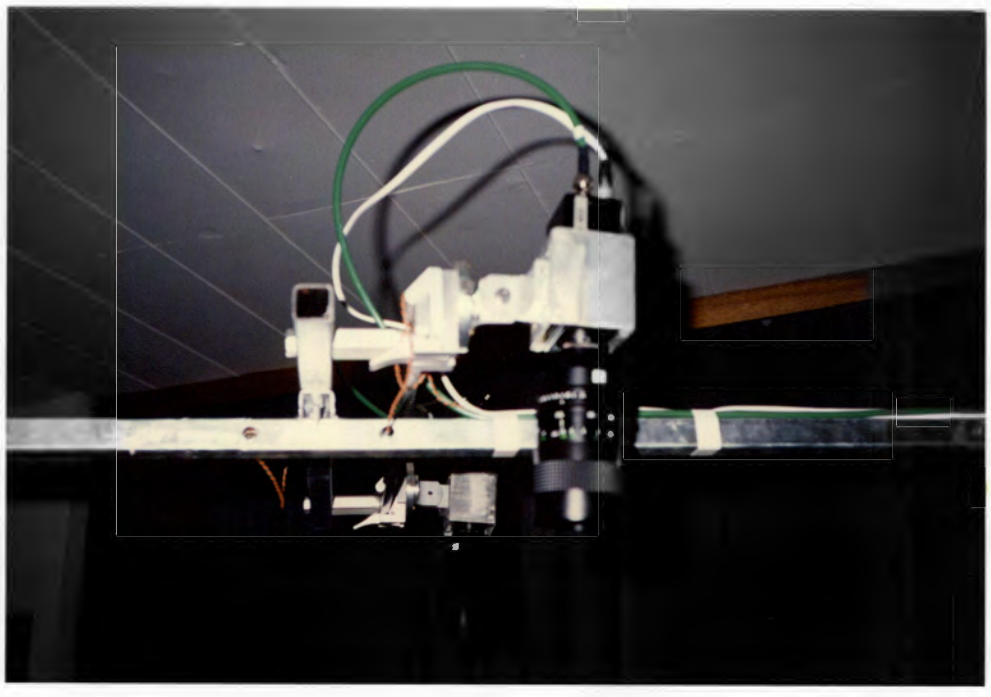

FIGURE 6.4 CAMERAS IN CAMERA HOLDERS SUSPENDED ABOVE THE BED

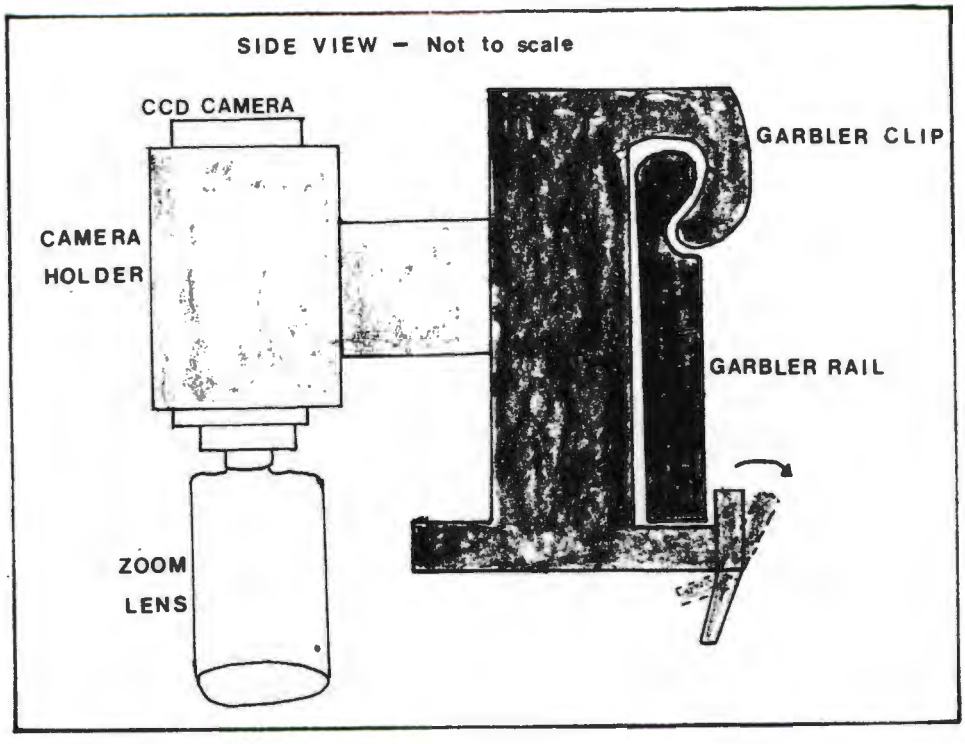

FIGURE 6.5 GARBLER RAIL AND GARBLER CLIP

The garbler rail holds three garbler clips. One garbler clip is positioned in the centre of the garbler rail and holds a swivel lamp with a 100w light bulb. The light intensity is controlled by a dimmer switch. The other two garbler clips are positioned at either end of the garbler rail.

The adjustment of the cameras' positions and orientations is done by changing:

Z by raising or lowering the whole gantry

X by moving the garbler clips on the garbler rail

the rotations about the X & Y axis by swiveling the camera holders

and is shown in figure 6.6 below.

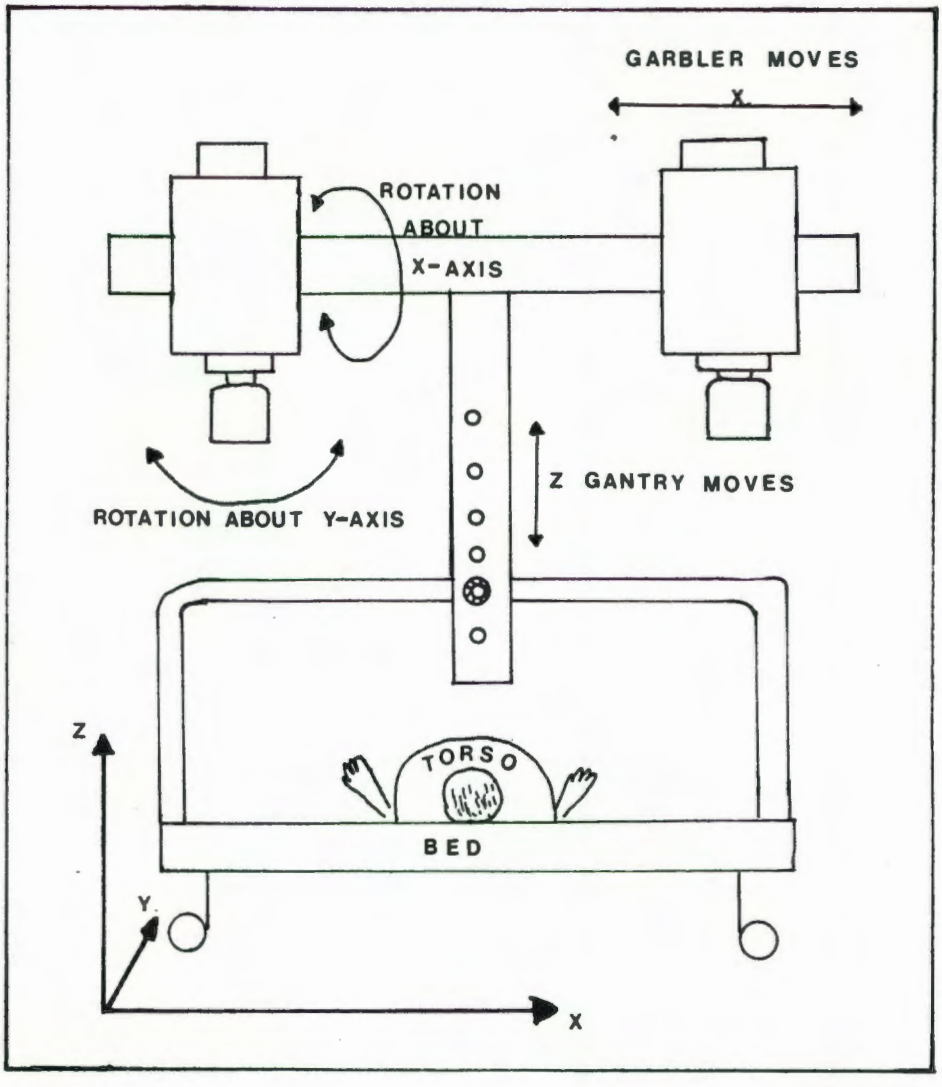

FIGURE 6.6 MOVEMENTS OF THE CAMERAS

Five cables connect the two cameras to an image mixer. Two cables supply power to the cameras and two cables send the video images to the mixer. The fifth cable is a sync (synchronisation) cable. The signal from one camera is used to synchronise it with the other camera through the mixer, so that video frames from both cameras are taken at the same instant in time.

6.1.3 THE IMAGE MIXER, THE RECORDING AND DISPLAYING OF "MIXED" FRAMES

The image mixer, as it's name implies, "mixes" images or more correctly single video frames. The video frames taken by the two cameras at the same instant of time are "overlayed", a portion of each frame from the cameras is taken and a "mixed" frame is formed, see figure 6.7 below.

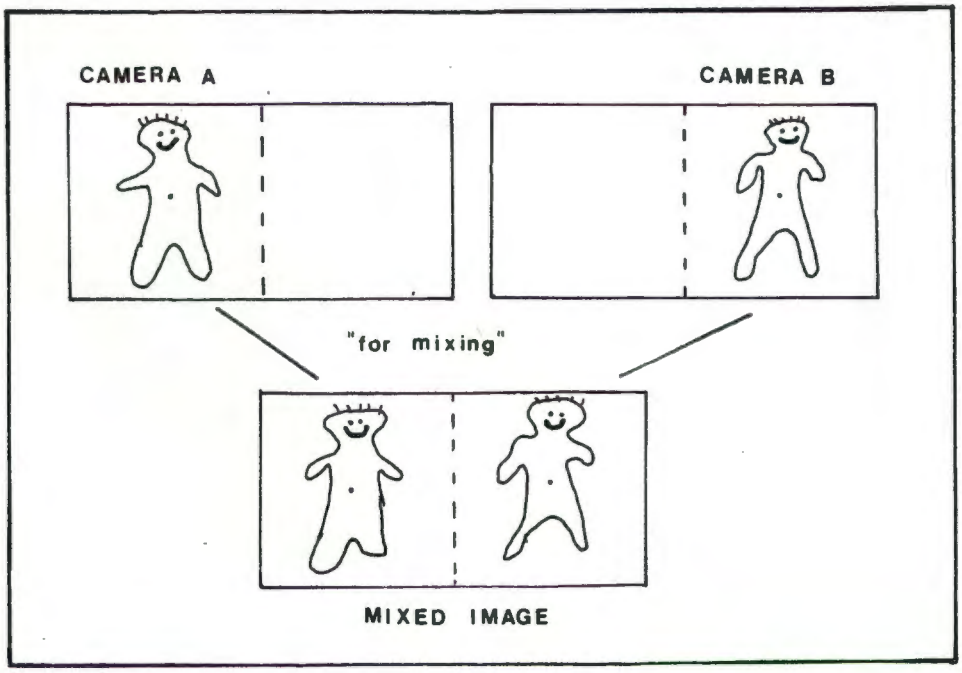

FIGURE 6.7 HOW IMAGES ARE MIXED

The mixer has two different types of control settings allowing the user to decide the type of "mix" that is required. One set of settings controls whether camera A (left) or camera B (right) will be on the left or right side of the "mixed" image. The second set of settings control what proportions are required from each image. The

proportions settings have two controls, one setting the proportions in a vertical and one in a horizontal direction. The mixer capabilities are shown in figure 6.8 below.

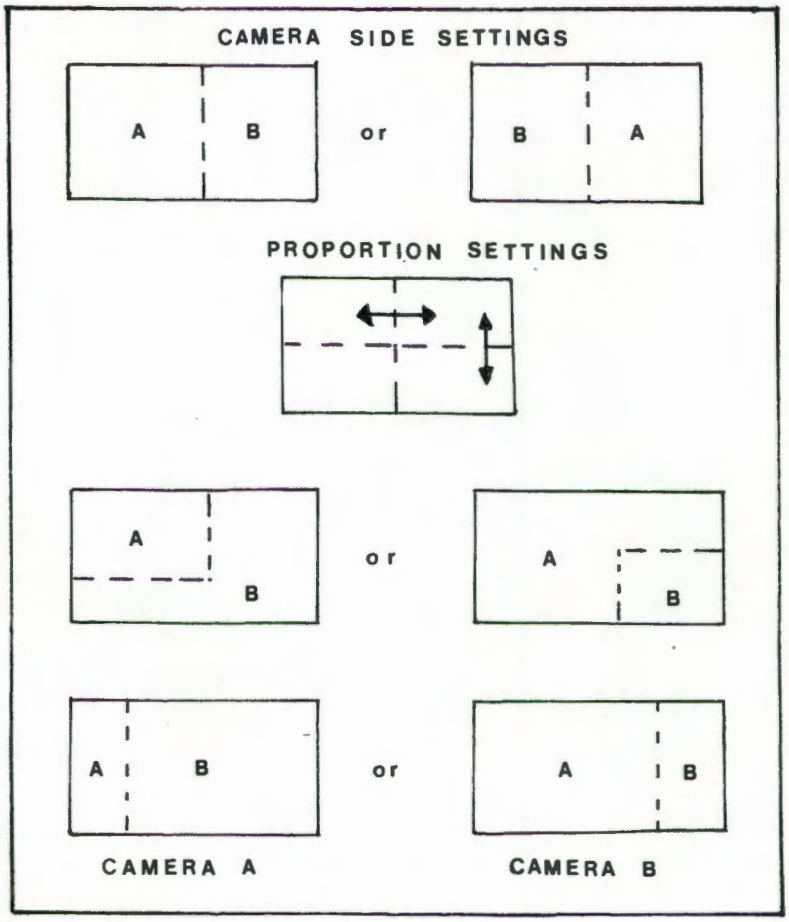

FIGURE 6.8 MIXER SPLITTING POSSIBILITIES

For the set up at the Red Cross Hospital, a pseudoscopic image is required. A pseudoscopic image is similar to the mixed image in figure 6.7, but the left camera image appears on the right and the right camera image on the left of the mixed image. The pseudoscopic image, when viewed in stereoscopic mode with the naked eye using the cross-eyed axis method or by using a special personal viewing stereoscope developed at UCT, is perceived as a three dimensional image.

The "mixed" frames are being updated twenty-five times a second from the cameras. The images are recorded on high quality video tape with the JVC video recorder, which is linked to the mixer. The JVC video recorder has the capabilities of freezing an image without noise (disturbances in the image), which is normally found in the home video machine. The video recordings serve as a permanent record and images can be retrieved for viewing and analysis.

To be able to set up the cameras and position the patient and control frames correctly, the "mixed" images received from the cameras are displayed on a 9" monitor, linked to the JVC video machine. The monitor is mounted on a swivel table at the head rest end of the bed, see figure 6.9 below.

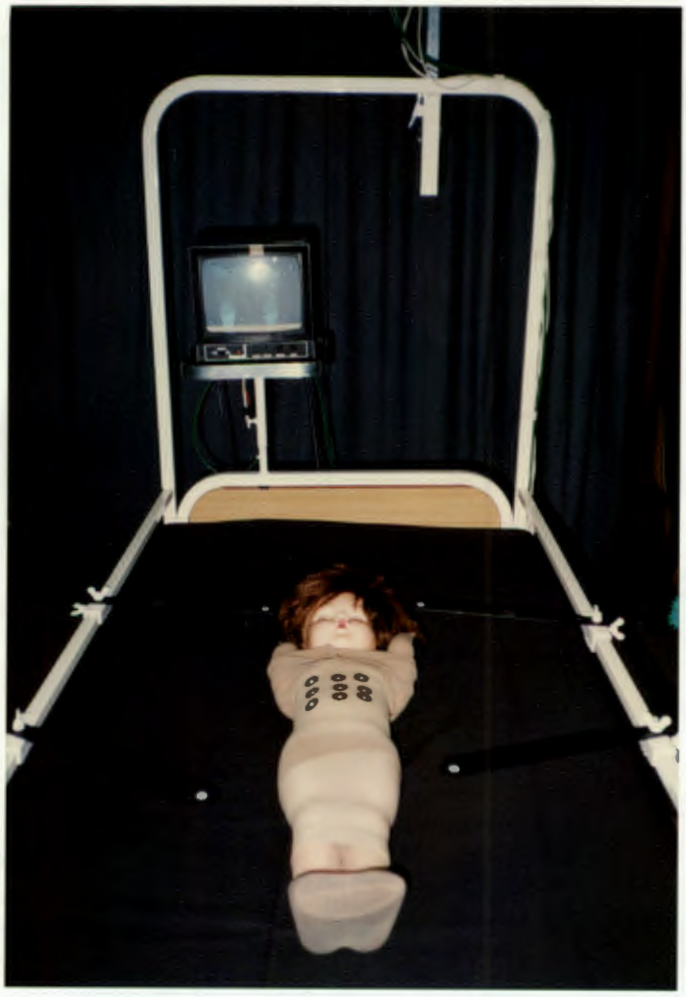

FIGURE 6.9 MONITOR ON SWIVEL TABLE AND FIDUCIAL MARKS

6.1.4 THE NEED FOR FIDUCIAL MARKS ON THE PHOTOGRAMMETRIC RIG

Suspended above the bed "surrounding" the test subject "Debby" are four matt black metal bars with four white markers/fiducial marks on their ends, as seen in figure 6.9.

The metal bars are mounted on clamps, which are clamped to the bed railings. The bars can be moved in a plane parallel to the surface of the bed to surround the patient's torso, as shown in figure 6.10 below.

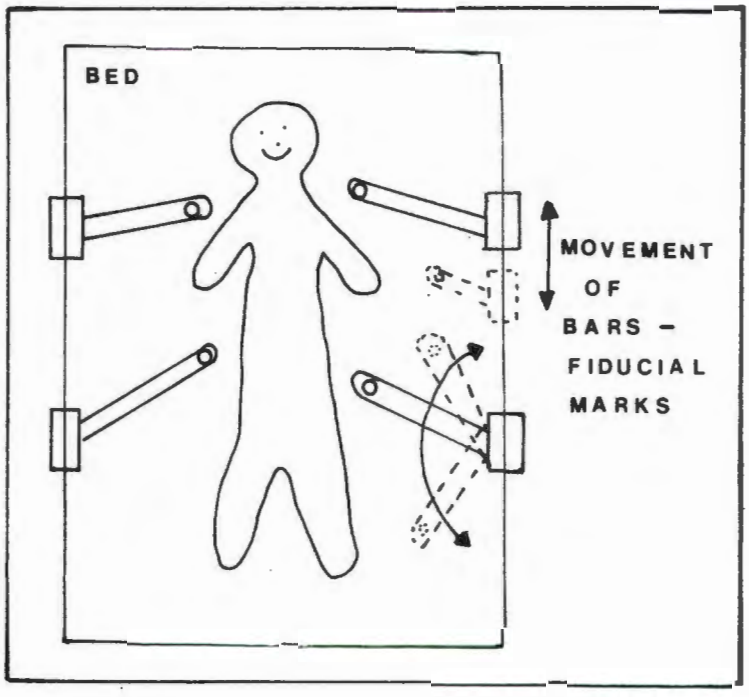

FIGURE 6.10 MOVEMENTS OF BARS WITH THEIR FIDUCIAL MARKS

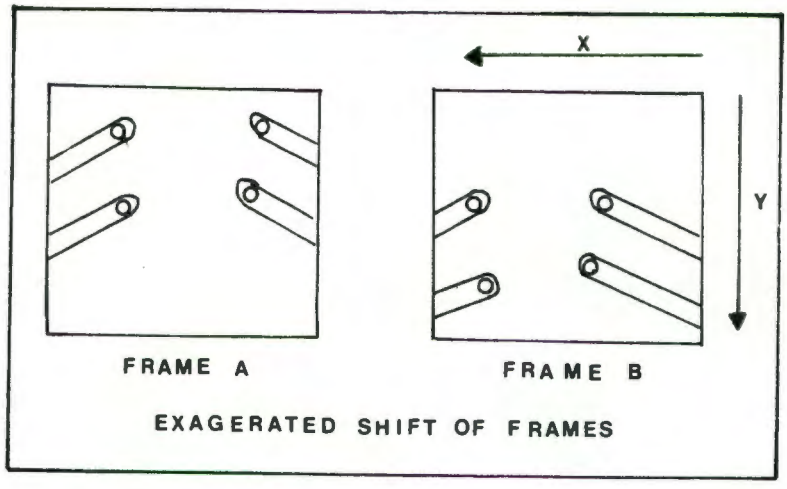

FIGURE 6.11 FIDUCIAL MARKS RELATING THE TWO FRAMES

The fiducial marks are needed to align subsequent image frames making up the video session of a patient's torso. As shown in figure 6.11 above a shift of the frames occurs on the monitor when single frames of the video tape are viewed with the JVC video's pause function.

Although this does not cause problems in viewing the frames it does cause problems in digitising, measuring and analysing different frames. The change of position of subsequent frames and therefore of the patient's torso due to the videos pause function, would be incorrectly analysed as part of the respiratory motion of the patient. This shift is caused by the JVC video's pause function. The frame shift can be likened to a rigid body, for example a square, which has been moved from it's original position. To move the square back to it's original position is impossible unless the original position has

been clearly marked. This can be done by placing four markers at the corner of each square before it is removed from it's original position.

The square in its original position represents the first/original frame of the patient's torso. The new position of the square, once it has been moved, represents the subsequent frames of the patient's torso with the frames having shifted due to the video's pause function. The corners of the square represent the fiducial marks. Thus the shifted video frames can be moved back to the first/original frame by using the fiducial marks as markers.

The four fiducial marks and a two point linear transformation are used to calculate the movement/shift of the frame. This shift is applied to all xy co-ordinates of digitised object points so that the object points are "shifted" back into the original frame's position. The two point linear transformation is discussed in chapter 8.

6.2 EQUIPMENT SET UP AT THE UNIVERSITY OF CAPE TOWN

The equipment set up at the University of Cape Town is needed for digitising, calculating and analysing video sessions and consists of an IBM PS2 computer, with 20 megabyte harddrive and two Pip Matrox image processing cards, two Philips television monitors, a Star Nx-15 printer and a Tektronix color image printer 4693D.

6.2.1 THE PIP MATROX IMAGE PROCESSING CARD

PIP is a plug-in card that allows the IBM computer to perform frame grabbing/digitising operations on a video signal from an external source - the JVC video machine.

The image processing cards can either be addressed through their interpreter or by using the Microsoft Fortran or Microsoft C commands in the software libraries provided. The images, when transferred to the computer, are converted from analogue to digital signals by the Pip card. The digital images are displayed on one of the Philips monitors. The frame grabber board has a resolution of 512 by 512 pixels, with eight bits per pixel. Each pixel consists of a grey-level/value ranging between 0 and 255, where 0 is black and 255 is white and values between 0 and 255 represent different levels of grey. The position

of any pixel can be measured on this 512 by 512 grid thus the Pip card can be used to digitize an image.

An image once snapped by the PIP frame grabber can be stored on the hard drive, using 262 K bytes of memory (an image consists of 512 by 512 pixels, 8 bits/1 byte per pixel, 262144 pixels/ 262144 bytes per image). The image can be recalled at any time by the PIP card for digitising or printing on the Tektronix image printer.

6.2.2 The Tektronix image printer

" The Tektronix 4693D color image printer produces high quality fast prints using a thermal wax printing process. It can print pictures of screen or data files from host computers, terminal display screens, graphic work stations, IBM-compatible AT personal computers and rasterizers. .....

The 4693D color image printer has a thermal wax print engine and a high speed digital color data communication interface. It completely floods image data from source devices in less than three seconds, freeing graphic systems for other use, and prints images at a resolution of 300 dots per inch." (Tektronix 1988)

The images captured and stored on the IBM computer are black and white images and have to be "modified" before printing on the Tektronix colour image printer. From the black and white images digital values are simulated, using a program written in C language, to form a "colour" image data files. These files can now be sent to the Tektronix colour image printer to print a "colour" image, although the printed image is in black and white. The C program was written by N.Parkyn for Tektronix (copyright - N.Parkyn).

7. PIXEL SIZE DETERMINATION

The pixels in a CCD camera do not have uniform proportions but scale differences between the vertical and horizontal axes, as mentioned in chapter 6.1.1. on the description of the CCD cameras. These scale differences had to be determined before an image could be accurately measured.

To determine the pixel size for the Ikegami cameras, the length and breadth of a pixel had to be compared to a length of known value. This was done by drawing up an accurate grid, with grid intersections at 5 cm intervals, on stiff cardboard that was mounted on a flat surface. The board was mounted perpendicular to the camera's line of sight. The camera settings were only important in so far as that the image was correctly focused and the light setting was correct. The length and breadth ratio is not affected by the changing of the focus or the zoom.

Images from the camera were fed directly into the IBM computer via the Pip Matrox image processing card and displayed on the Phillips monitor. Grid intersections were measured by using a program Phoenix (written by H. Ruther & R.Wildcheck). This process was repeated for both Ikegami cameras.

The grid lengths and breadths for camera A on average were 47.2 (standard deviation 0.2) and 66.7 (std. dev. - 0.4) pixels respectively and for camera B 48.3 (std. dev. +\- 0.2) and 68.5 (std. dev 0.4) respectively, see figure 7.1 below.

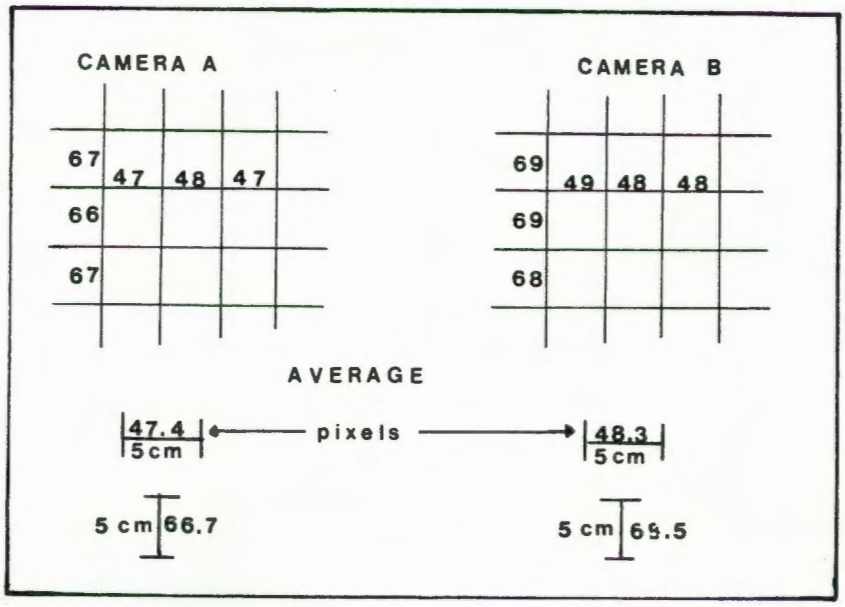

FIGURE 7.1 GRIDS FOR CAMERA A AND B

As each grid section has a length of 5cm, the length and breadth for camera A for that particular set up was 1.1mm and 0.7mm respectively and for camera B for its particular setting was 1.0mm and 0.7mm respectively. (Figures are rounded to two significant figures). The ratio of length to breadth for camera A and B are 1.4.

Two Siemens cameras at the Department of Surveying at the University of Cape Town have the same dimensions for their image processing chips as that of the Ikegami cameras and have had the length and breadth of the pixel more accurately determined than that of the Ikegami cameras. The factors applied for the Siemens cameras are 0.015m and 0.011m for length and breadth respectively giving a length/breadth ratio of 1.36. Thus it was decided to adopt the scale factors determined for the Siemens cameras for the Ikegami cameras, this adoption of the Siemens factors being verified by good accuracies on the control frame co-ordinates.

Thus the scale adjustments on all measurements on images taken by the Ikegami cameras are X co-ordinates * 0.015 and Y co-ordinates * 0.011.

8. TWO DIMENSIONAL LINEAR TRANSFORMATION TO CORRECT FOR VIDEO SHIFT

The four fiducial marks and the two point linear transformation are used to calculate the movement/shift of the video frames due to the pause function of the video machine.

The two point linear transformation is the transformation of one system of co-ordinates to another and assumes a rotation of axes and a scale change, see figure 8.1.

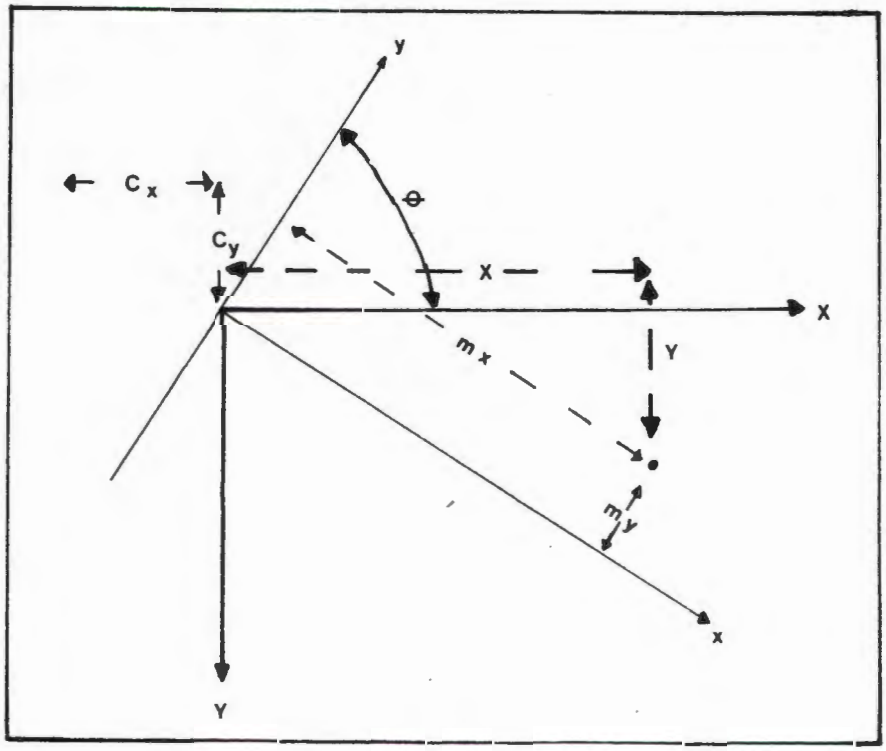

FIGURE 8.1 CO-ORDINATE AXES OF THE TWO SYSTEMS

The equations of the two point linear transformation are:

$$X = a*x + b*y + Cx$$

$$Y = -b*x + a*y + Cy$$

where: x,y - co-ordinates of one system to be transformed to another
i.e. subsequent video frame co-ordinates to be transformed to the original frame co-ordinate system

X,Y - the transformed co-ordinates
i.e. subsequent video frame co-ordinates now in the co-ordinate system of the original video frame

Cx,Cy - Centre of gravity co-ordinates of the original system
i.e. centre of gravity co-ordinates calculated from the four fiducial marks of the first patient frame

a,b - a = m*cos(ɵ)
b = m*sin(ɵ)
where m is the scale change and Í is the rotation of the axes

The fiducial mark co-ordinates (X,Y) of the original frame are used to calculate the centre of gravity Cx and Cy of their original frame.

The fiducial mark co-ordinates (x,y) of a subsequent video frame are used to calculate the centre of gravity of the fiducial marks of the subsequent frame and to reduce the fiducial mark co-ordinates with respect to that centre of gravity. The object co-ordinates of the points on that frame are also reduced with respect to the centre of gravity of the subsequent video frame.

The fiducial mark co-ordinates of both sets are used in the two equations to solve for a and b using a least squares adjustment. The two point linear transformation, as indicated by its name only needs two points for the transformation. This however results in a unique solution. Using four fiducial marks and a least squares solution results in redundancies and a more accurate solution.

Once a and b have been determined it is possible to reduce the object co-ordinates (x,y) to the original frames' co-ordiante system, i.e. to X,Y co-ordinates.

This process is used for both the left and the right image of a stereo video frame. The whole process of the two point linear transformation is calculated in the program D3.C, in appendix 17, that calculates the three dimensional co-ordinates of the object points.

9. DETERMINING THREE DIMENSIONAL CONTROL FOR THE TWO CONTROL FRAMES

Control points, which are required to calculate the b parameters of the projective transformation, are provided by a control frame. Two control frames were constructed, one for children and adults and one for babies.

Both control frames have been constructed with thick metal wire, painted matt black, and with control points consisting of a matt black circle with a white centre, see figure 9.1 below.

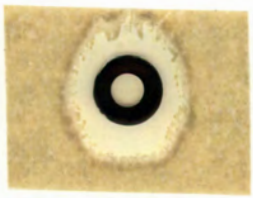

FIGURE 9.1 EXAMPLE OF TARGET

9.1 CO-ORDINATING THE BABY CONTROL FRAME

FIGURE 9.2 BABY CONTROL FRAME

The baby control frame, in figure 9.2 above, consists of eighteen control points. There are 9 points on the lower and 9 points on the upper plane/level of the control frame. The planes are about 14cm apart.

The numbering of the different control points is depicted in figure 9.3 below.

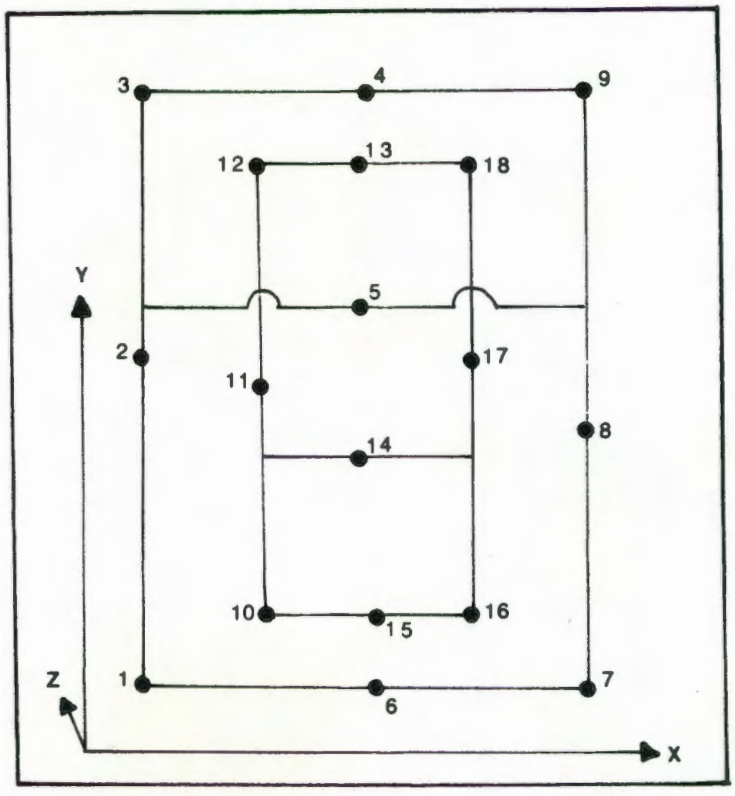

FIGURE 9.3 NUMBERING OF CONTROL POINTS ON THE BABY CONTROL FRAME

The baby control frame was first measured in a reflex metrograph (Scott 1981) linked to a Tektronix 4051 computer. The metrograph has a resolution of 0.1mm in each of the x,y and z axes. The control frame was measured four times by two different observers and the mean co-ordinates calculated.

The positions of the markers on the control frame were also measured twice, using video cameras, an IBM computer,the programs Phoenix (Prof. H. Ruther and R.Wildcheck) and Hittum4 (Prof. L.P.Adams) and the mean of the results calculated.

The IBM values were transformed into the metrograph system using Rodrigues parameters (Thompson 1969, p153-157).

The average accuracy for control point co-ordinates for Dx is 0.2mm, for Dy is 0.2mm and for Dz is 1.0mm. The greater z range discrepancy is due to the single pixel reading accuracy of Phoenix. The discrepancy in z occurs over a z range of 26cm. As the patient's chest rises and falls over a much smaller z range this discrepancy in accuracy is acceptable.

Three dimensional co-ordinates for the baby control frame can be found in appendix 2.

9.2 CO-ORDINATING THE CHILD/ADULT CONTROL FRAME

FIGURE 9.4 CHILD/ADULT CONTROL FRAME

The larger control frame, the child/adult control in figure 9.4 above, consists of 24 control points, 12 points on the lower and 12 points on the upper plane. The two planes are separated by 14cm.

The numbering of the control points is shown in figure 9.5.

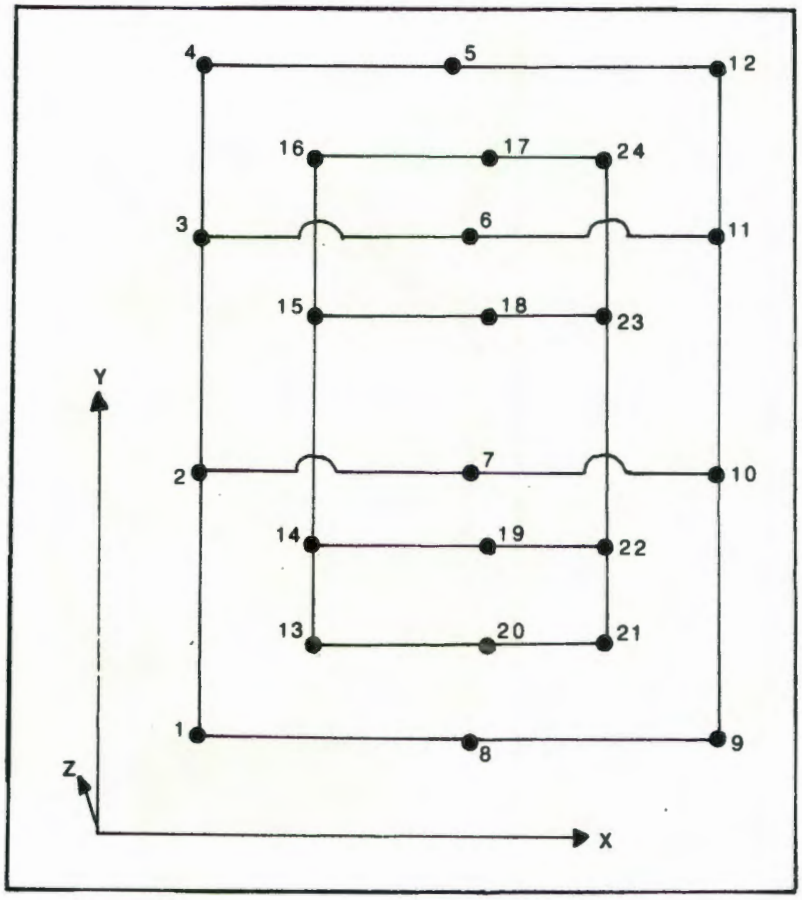

FIGURE 9.5 NUMBERING OF THE CONTROL POINTS OF THE CHILD/ADULT CONTROL FRAME

As the child/adult control frame was too large to be measured in the metrograph, it was placed in the calibration chamber of the Department of Surveying at the University of Cape Town and photographed by a pair

of Zeiss UMK 10/1318 metric cameras (maximum lens distortion of approximately 7 microns), as shown in figure 9.6.

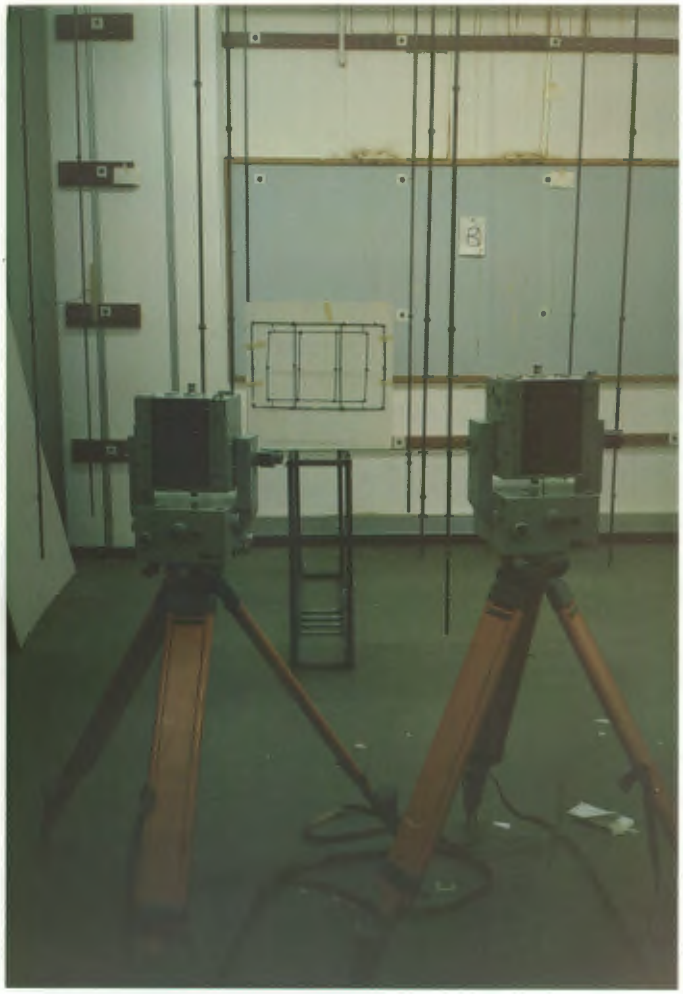

FIGURE 9.6 CONTROL FRAME IN CALIBRATION CHAMBER

The film was not lowered onto the cameras to increase the principal distance and a long exposure was used. The negatives were measured twice in a Carl Zeiss Jena stereocomparator, model 1818.

The heights of the control points were remeasured twice using a Sokisha level and a straight edge. The control was first remeasured when measurements using the control resulted in Dz accuracies of only 6.1mm. These however were not due to faulty control but due to incorrect camera set ups. The control had to be relevelled a second time, due to a defective solder joint which had to be repaired, and which caused slight changes in the z co-ordinates of the control points.

Three dimensional co-ordinates for the child/adult control frame can be found in appendix 3.

10. CAPTURING THE RESPIRATORY MOTION OF A PATIENT ON VIDEO TAPE

Before a video session of a patient can begin the photogrammetric rig has to be correctly set up.

10.1 SETTING UP THE RIG FOR VIDEOING THE RESPIRATORY MOTION

Two requirements have to be met when setting up the rig to record a patient's respiratory motion.

1. The principles of the traditional set up have to be adhered to, that is the geometry of the set up must be correct, i.e. base/height ratio must be correct and the cameras' lines of sight must be parallel or converging.

2. The cameras have to be set up to allow for pseudoscopic (3D) viewing of the patient's respiratory motion.

To satisfy the first requirement the base/height ratio must not be smaller than 1:5. The camera base (the separation between the camera holders on the garbler rail) is usually set at 0.6m. The height of the gantry above the bed varies. For children and adults the child/adult control frame is used and the gantry

height is set at 1.5m, with babies the baby control frame is used and the gantry height is set at 1.0m. Thus the base/height ratio is maintained.

The cameras are set up with slightly converging rays so that the angles of intersection of the cameras' line of sight allow for good co-ordination of object points. The 6.1mm discrepancies in z of the control (see ch. 9.2), when measuring the control, occurred when a shorter camera base and parallel camera rays were used.

The angle of intersection is much greater for a larger camera base and converging lines of sight than for a smaller camera base and parallel lines of sight, as shown in figure 10.1 below.

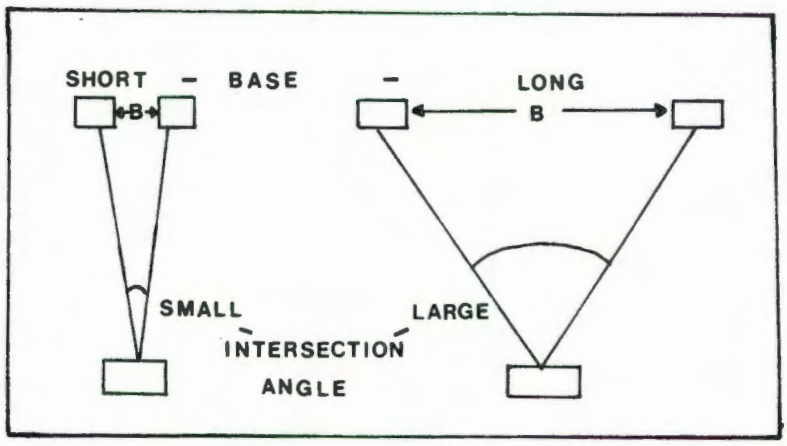

FIGURE 10.1 DIFFERENT ANGLES OF INTERSECTION FOR DIFFERENT CAMERA BASES

The larger angle of intersection in figure 10.1 is more accurate than the smaller angle in determining the co-ordinates of the point of intersection.

The second requirement is satisfied by setting the cameras up correctly above the patient's chest. Both cameras must capture the whole area occupied by the control frame. The ability to rotate the camera holders permits the correct alignment of the stereoscopic pairs and the zoom lenses permit scale matching.

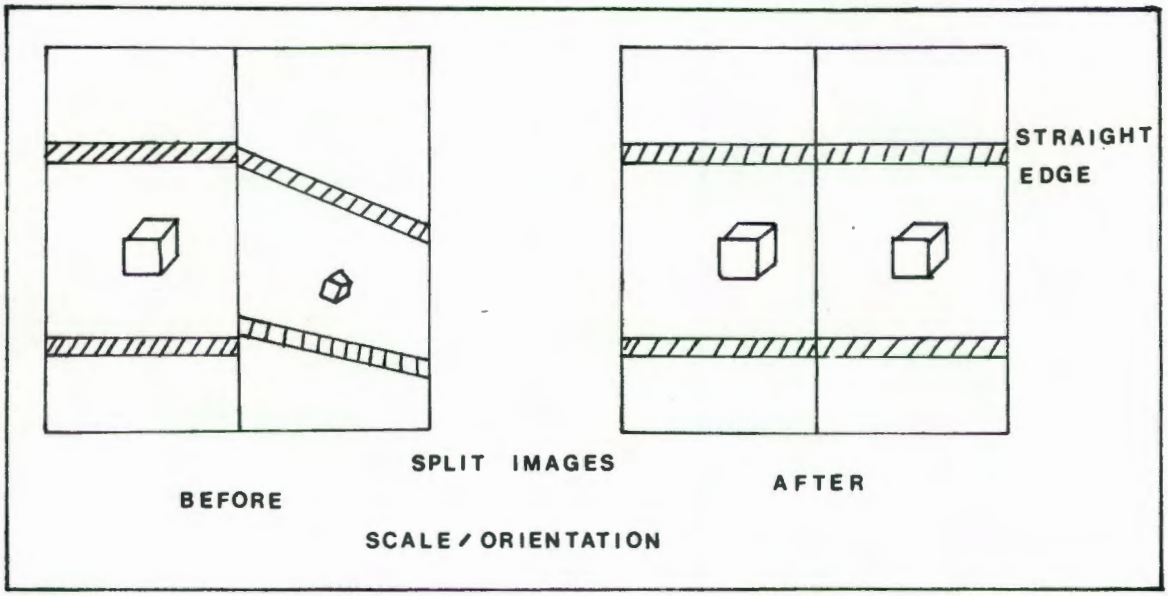

FIGURE 10.2 STRAIGHT EDGES BEFORE AND AFTER ORIENTATION

Correct alignment and scale matching is achieved by using two long straight edges placed on the bed parallel to the headboard, the two straight edges separated by about 40cm. Before alignment, as shown in figure 10.2, the straight edges are "non-parallel" in the stereopair, nor are the "left and right" objects at the same scale. After alignment, using the two rotations of the camera holders, and scale matching, using the zoom lenses, the straight edges are parallel and the "objects" at the same scale. The control frame is then placed on the bed to insure that the whole control frame appears on both the left and right image of the stereopair.

The settings on the mixer must be checked to ensure that the left camera image appears on the right and the right camera image on the left of the mixed image to form a pseudoscopic pair so it can be viewed in stereoscopic mode with the cross-eyed axis method. A pseudoscopic set up is required as the camera base is too large for a normal stereopair to be viewed stereoscopically.

10.2 TARGETING OF THE PATIENT BEFORE A VIDEO SESSION AND TARGET DESCRIPTION

The patient has to have specific areas of interest targeted by the physician, before a video session can begin. Figure 10.3 gives an example of the positioning of the targets.

Two targets, one affixed to each shoulder, are later used to determine whether any patient movement has taken place during a video session. This would result in erroneous results with patient movement being mistaken for respiratory motion.

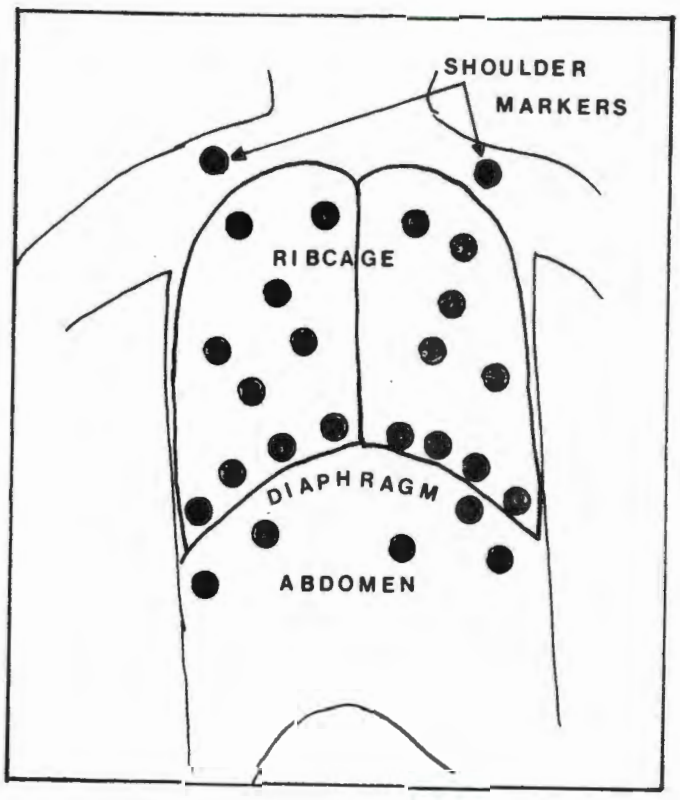

FIGURE 10.3 THE TORSO WITH TARGET POSITIONS

The targets consist of a white centre and a matt black outer circle. The illumination of the targets is controlled by the dimmer switch of the 100 watt lamp and the aperture settings on the two cameras. The targets must be illuminated in such a way so that the inner white circle appears clearly without "blacking" out the black outer circle, i.e. the light from the white centre must not flood the light from the black outer circle. An example of a target and different illumination levels are shown in figure 10.4 below.

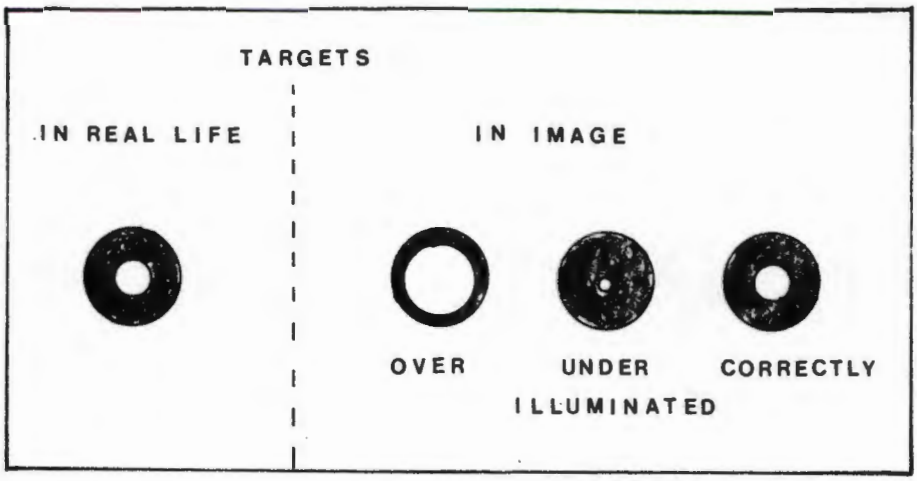

FIGURE 10.4 DIFFERENT ILLUMINATIONS OF THE TARGET

The black and white target is required for the centre of gravity routine, described in chapter 11.3.2, to achieve sub-pixel and thus also sub-millimetre accuracy for determining the target centre.

10.3 THE VIDEO SESSION TO RECORD THE RESPIRATORY MOTION

At the start of a video session the video tape portion pertaining to that patient has to be "labeled". An identification paper - with the patient's name, the date of the recording and any specific information that is required - is placed on the bed and recorded on tape.

The patient is then positioned on the bed so that the torso is viewed by both cameras. When moving the patient on and off the bed care has to be taken that the cameras are not moved.

The metal bars, with the four fiducial markers, are moved in to position around the patient's torso so that they appear in the four corners of the left and right camera images, see figure 10.5 below.

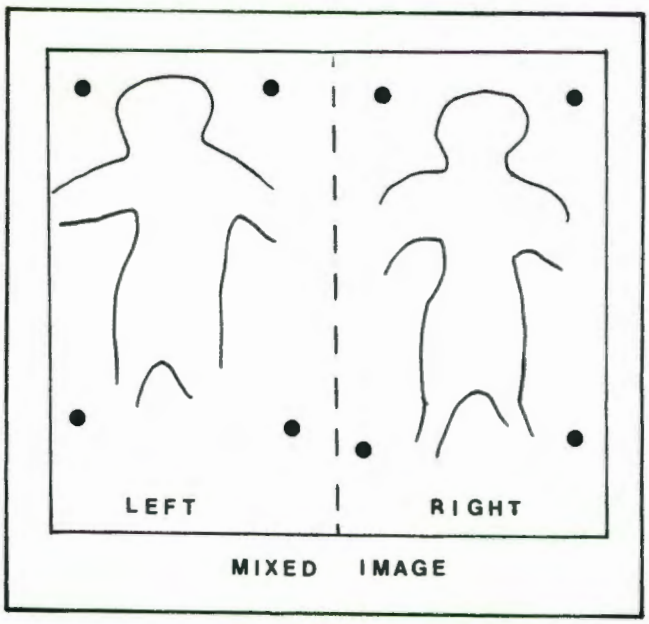

FIGURE 10.5 FIDUCIAL MARKERS AROUND PATIENT

The respiratory motion of the patient is recorded on video tape. This part of the session is controlled by the doctor as different breathing motions may be required to study the patient's respiratory motion.

The patient is then removed from the bed and the control frame is placed in the position previously occupied by the torso so that the whole of the control frame appears in both the left and right camera image. The cameras must not move between the two sections of the video session, i.e. none of the camera constants

are changed. When positioning the control frame on the bed, the x axis of the control frame must be parallel to the base line of the cameras to satisfy conditions of the projective transformation. The control frame can now be captured on video tape.

Once the video session has been completed, the video tape is taken to UCT, where digitising and calculations can begin.

11. TAG - THE PC BASED COMPUTER SYSTEM TO ANALYSE RESPIRATORY MOTION

As a system was being developed as a diagnostic tool for the hospital environment it had to be both functional and user friendly. A computer package had to be developed to allow the user to select and capture the necessary images from the video film, measure the images and view the final results without becoming involved in any of the numerous photogrammetric calculations.

The computer system developed to satisfy these criteria was named TAG (Tregidga-Adams-Gutschow). TAG consists of a series of menu driven programs written in Microsoft C and True Basic.

The programs that make up the TAG computer system are listed below and the appendices in which the computer listings are to be found:

1. Main - contains and runs main menu of TAG
   - appendix 4
2. Dmenu - contains and runs digitising menu
   - appendix 5
3. Pmenu - contains and runs calculating menu
   - appendix 6

4 Gmenu - contains and runs graphics menu
- appendix 7

5 Tod - transfers images from video machine to IBM
- appendix 8

6. Conl - for digitising left control images
- appendix 9

7. Conr - for digitising right control images
- appendix 10

8. Objl - for digitising left object images
- appendix 11

9. Objr - for digitising right object images
- appendix 12

10. Single - for digitising single images
- appendix 13

11. Outline - for digitising patients' outlines
- appendix 14

12. Cursor - runs in conjunction with digitising programs
- appendix 15

13. Bparams - to calculate b parameters
- appendix 16

14. D3 - to calculate three dimensional co-ordinates
- appendix 17

15. Main/Vec - to superimpose vectograms on images
- appendix 18

16. Tplt.tru - to plot xy displacement vectors, z displacement vectors and three dimensional space vectors
    - appendix 19
17. Imp - to print images on the Tektronix printer
    - appendix 20
18. Graph1 - subprogram that runs most of the graphics functions and graphic on screen instructions
    - appendix 21

11.1 USER FRIENDLINESS OF THE TAG SYSTEM

TAG is a user friendly computer system in that :

(i): a readme document exists, which the user can read before running the TAG system, which will give him an overview of the different programs and their operations in TAG. (Readme.doc in appendix 22)

(ii): the user is guided throughout the programs with on screen instructions from connecting and operating the video machine with the Pip Matrox card, operating the TAG "digitiser", calculating and outputting

results and printing images

(iii): the system generates all file name extensions from the minimal information, i.e. the patient's name and the breath number, which the user has to enter

(iv): the programs only respond to the correct input, e.g. "Press the SPACE BAR to continue" will only allow the program to continue if the space bar is pressed, any other key entered will be rejected and the instruction repeated.

(v): all single key entries, e.g. " Do you wish to continue y/n : ", where only a single letter such as "y" or "n" has to entered, is immediately responded to by the computer without using the enter key; here, as with the example of the space bar above, the computer will only respond to the correct entry of either "y" or "n" or else repeat the instruction until the input is correct

A video session of a patient named Mary is used to demonstrate the various functions of the programs that make up the TAG computer system. Mary, a patient at the Red Cross Children's Hospital, is an eleven year old girl

who has one non-functioning and one diseased lung due to a severe post measles pneumonia.

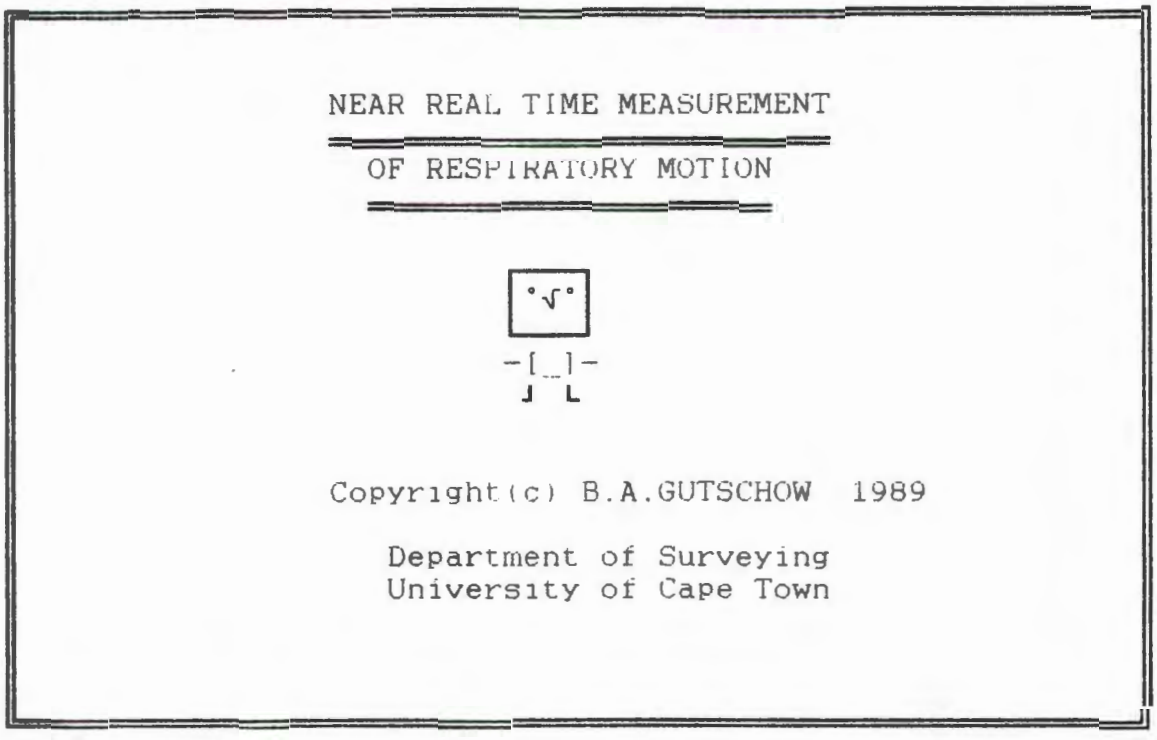

FIGURE 11.1 TAG TITLE PAGE

When initialising the TAG computer system the 'Title page', figure 11.1 above, is called by the program Main from the program Graph1.

The program Main contains and runs the TAG main menu, displayed in figure 11.2 below.

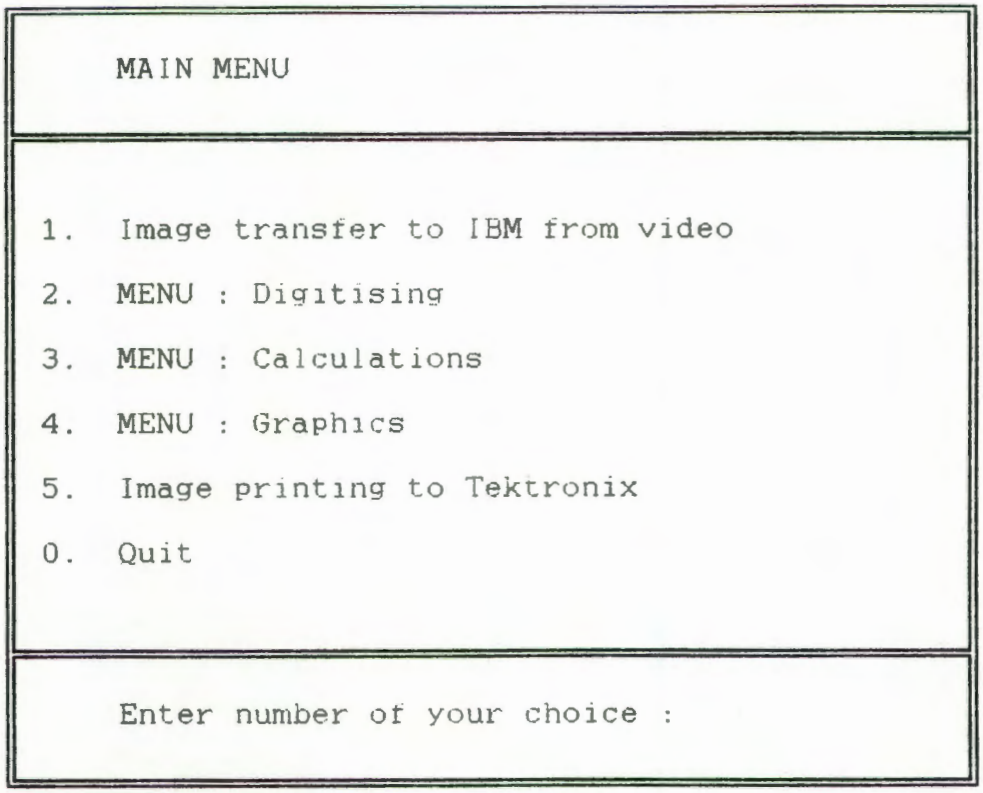

FIGURE 11.2 MAIN MENU OF TAG

Options 1 to 5, once called up and completed, return the user to the main menu. Option 0 returns the user back to the DOS system.

11.2 SELECTING AND TRANSFERRING IMAGES TO THE IBM COMPUTER

Option 1 of the main menu - " 1. Image transfer to IBM from video " - run by the program Tod, allows the user to select specific video frames from a video session of a patient and transfer them to the IBM computer via the Pip Matrox image processing card.

On screen instructions describe how to connect the video machine to the PIP Matrox card. The name of the patient "Mary", whose video session is to be worked on, has to be entered. The user is then given the option to transfer a control image (image of the control frame) or object images (images of Mary's torso).

The playback of Mary's video session is displayed on the Philips monitor via the Pip Matrox card and the IBM computer.

First a good control image, as shown in figure 11.3 below,is selected. Only one control image is needed for digitising, and calculating the b parameters for the projective transformation.

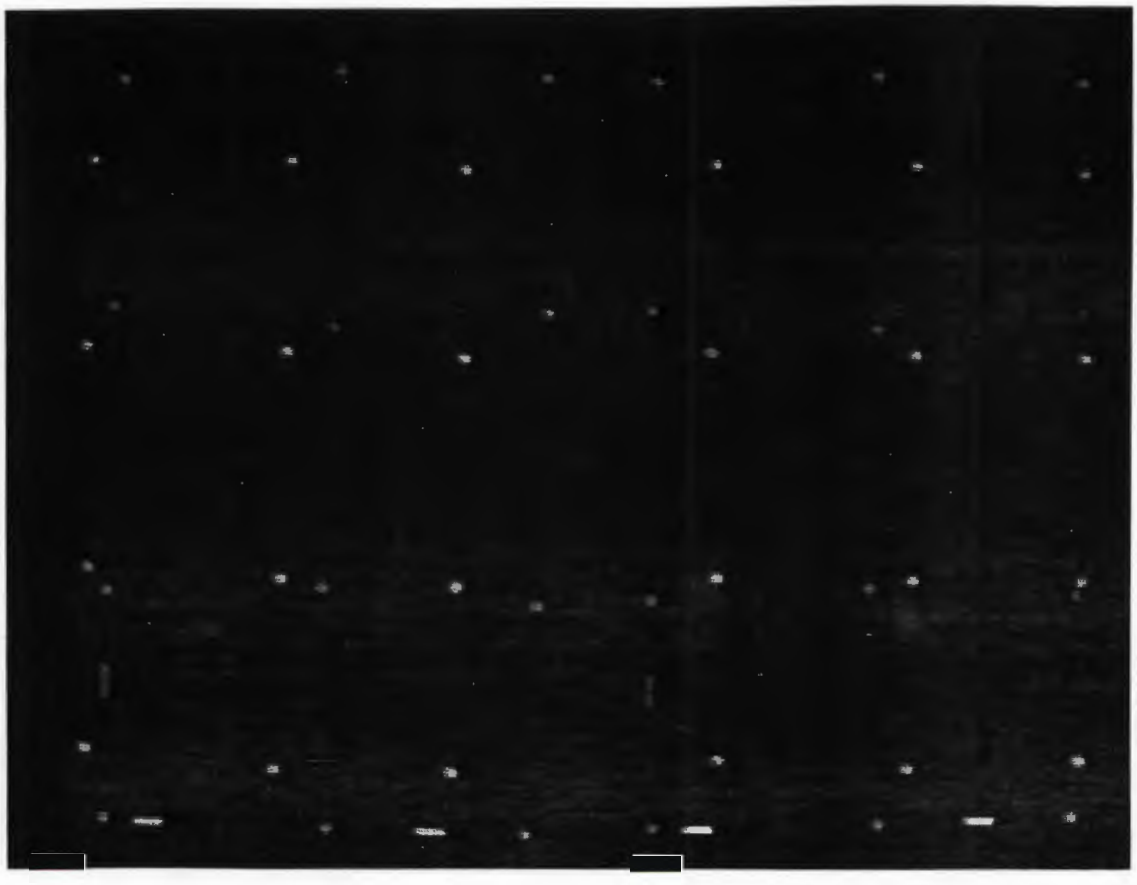

FIGURE 11.3 STEREO VIDEO FRAME IMAGE OF THE CHILD/ADULT CONTROL FRAME

By stereoscopically viewing the playback of Mary's respiratory motions in slow motion it is possible to

select video frames at the extreme breath in and breath out of different respiratory cycles, one such extreme breath in frame depicted in figure 11.4 below.

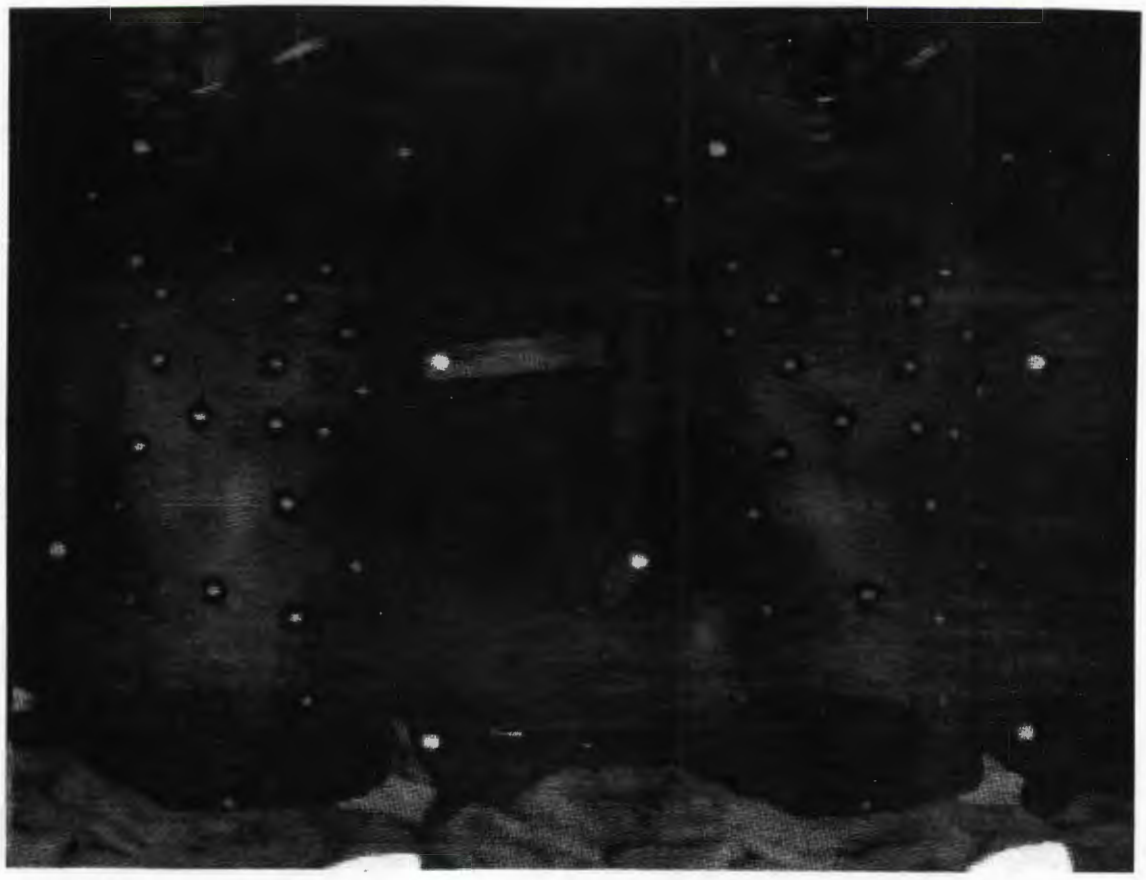

FIGURE 11.4 STEREO VIDEO FRAME IMAGE OF MARY'S TORSO AT THE EXTREME BREATH IN

The three dimensional view of the patient's torso is achieved by either crossing one's eyes or by using the viewer developed by Prof. L.P.Adams (Adams et al 1990).

The breath number of an image and the position in the respiratory cycle must be entered, e.g. 1i - breath 1 at full inspiration. Once the image has been selected by freezing the playback, the video frame/image is transferred to the IBM harddrive. The process of capturing object images is repeated until all the required images of the patient's respiratory cycles have been captured.

The file names for the frames are created by the program using the patient's name "Mary", the breath number, and position in the respiratory cycle, e.g. "1i", and the extensions created by the program. The image names are created in the following manner:

1. control frame image: "Mary" + "c" = "Maryc"
2. stereo object image: "Mary" + "o" + "1i" = "Maryo1i"

Once the transfer of images of that particular patient's video session is completed, the user is given the option of processing another patient's video tape or quitting the option and returning to the main menu.

11.3 DIGITISING THE TRANSFERRED IMAGES

Option 2 of the main menu - " 2. Menu : Digitising " - run by the program Dmen, calls the digitising menu, which consists of seven different suboptions, as shown in figure 11.5 below.

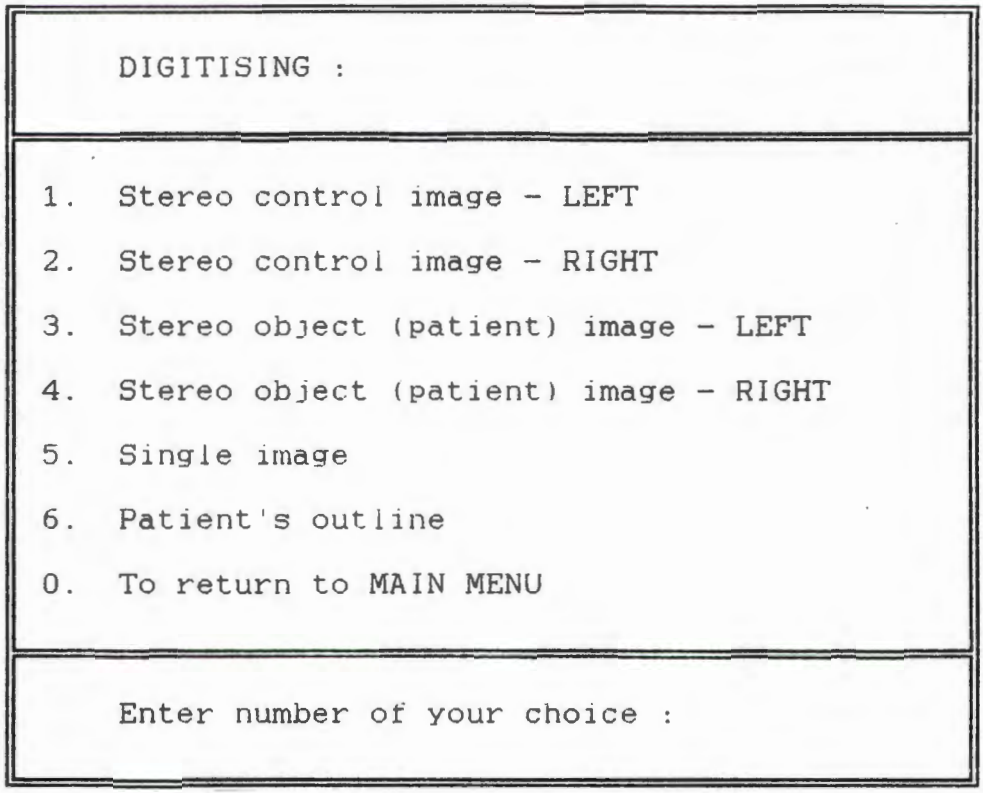

FIGURE 11.5 DIGITISING MENU

Suboptions 1 to 5 are "shell" programs that all make use of the same subprogram Cursor. Suboption 6 is a modified version of the Cursor program. Suboption 0 returns the user to the main menu.

11.3.1 CURSOR - THE DIGITISING PROGRAM

The TAG computer system can be seen as a glorified DO-IT-ALL-IN-ONE digitiser and thus the main core of TAG is the Cursor program which enables the TAG system to digitise digital images.

A digitiser is used to determine xy co-ordinates of points on a two dimensional image in the co-ordinate system of the digitiser. The TAG digitiser consists of a 512 * 512 pixel co-ordinate system, predetermined by the Pip Matrox card, with the origin (0,0) of the system in the left hand upper corner of the monitor. The image/video frame is retrieved from the harddrive and displayed on a Philips monitor in the 512 * 512 pixel co-ordinate system.

To determine xy co-ordinates of specific points on the image a pointer in form of a cursor is created by the subprogram Cursor and superimposed on the digital image.

The cursor consists of two perpendicular lines, where the intersection area of the two lines has

been removed and replaced by a single dot the size of one pixel, see figure 11.6. The intersection area was removed as it obliterated too much of the area it was "pointing" to. The single dot, now representing the centre of the cursor, can be positioned with much greater ease on a point whose xy co-ordinates are to be determined.

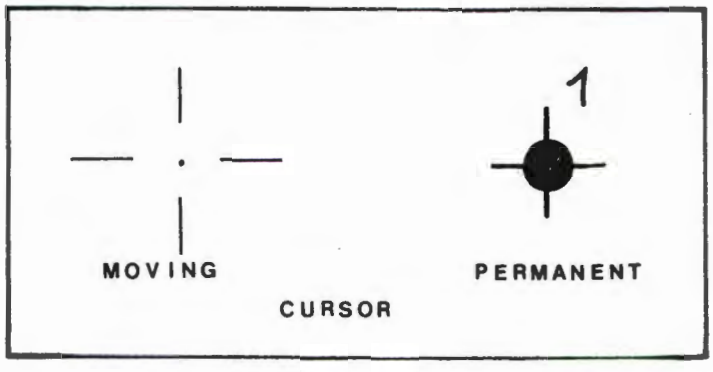

FIGURE 11.6 CURSOR ON SCREEN AND MARKED TARGET NO.1

The points whose co-ordinates need to be determined by the TAG system are the control targets on the control images and the fiducial marks and object targets on the patient's torso images. Once the cursor has been positioned on a target and the enter button pressed the targets are marked by a permanent cross and labelled with a number, as shown in figure 11.6 above.

The xy co-ordinates of the centre of the target are determined to sub pixel accuracy (0.1 of a pixel) using a centre of gravity routine in the Cursor program and this process is described in chapter 11.3.2 below.

The Pip Matrox card does not have any cursor features or digitising functions, except for the ability to read a position on the xy grid of the Pip Matrox card. The cursor, as described above, had to be created using graphics features (lines and dots) of the card. Before the cursor is drawn on the image, the grey values of the pixels that are going to be overwritten by the cursor have to be read and saved or else they are permanently lost. When the cursor is moved to a new position, the grey values of the old cursor position are replaced and the image is restored.

The grey values of the new cursor position are read before the new cursor is drawn. Thus the cursor can be moved over the image without destroying the original image. The permanent crosses, that mark points that have been co-ordinated, are simply created by not replacing the cursor with the original grey values, but by grey values of 255 creating a permanent white

cross.

The xy co-ordinates of the cursor position are kept in memory and updated with each move (1 or 25 pixel jump) of the cursor. The xy co-ordinates of the cursor position are used to calculate what pixels make up the cursor so that their grey values can be read and stored in memory.

The movement of the cursor on the image is controlled via the computer keyboard using the arrow keys, the insert, delete, home, end, page up and page down keys and the numeric key five, as shown in figure 11.7 below.

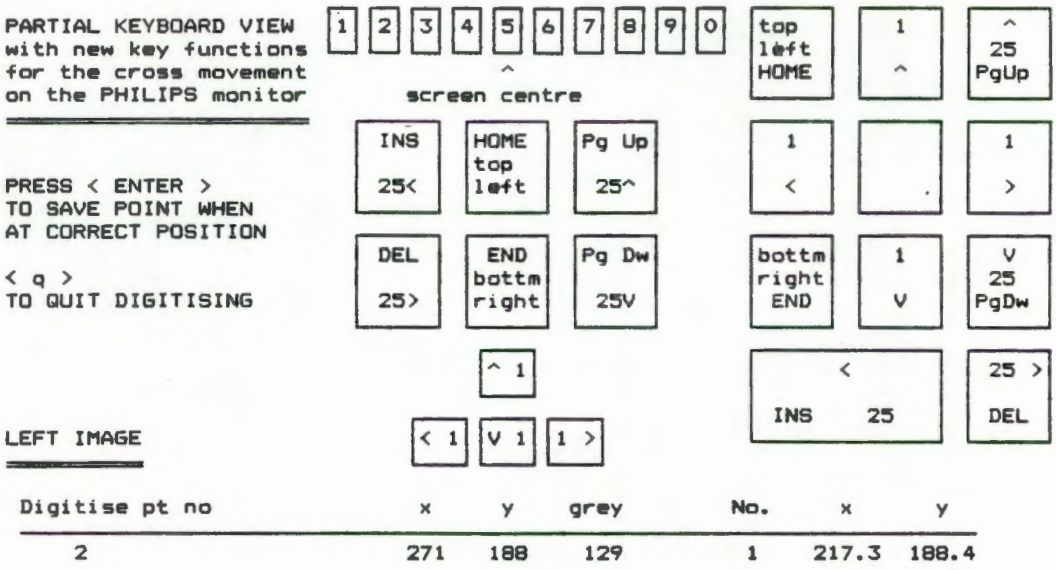

FIGURE 11.7 LAYOUT FOR CURSOR MOVEMENT KEYS

Figure 11.7 is a computer display on the IBM computer monitor which serves as a guide to the user when digitising.

The lower portion of the digitising guide in figure 11.7 consists of:

1. an indicator to show whether the left or the right image of the stereo image is to be digitised
2. the number of the next point to be digitised
3. the current position of the cursor on the screen and its grey level
4. the number and xy co-ordinates, to sub pixel accuracy, of the last point that was digitised.

The different digitising options call up the image to be digitised and the subprogram Cursor. Cursor positions the cursor in the centre of the Philips monitor at the start of every new digitising session. The user, using the keyboard keys, moves the cursor to the centre of each target he wants to digitise and presses the enter button. The xy co-ordinates, in pixel units, are then written to a text file designated by the digitiser options.

A text file of a digitised image consists firstly of the number of points that were digitised on that image and then the number and x and y co-ordinates of those points. The digitising option returns the user to the digitising menu.

11.3.2 CENTRE OF GRAVITY ROUTINE IN CURSOR TO DETERMINE THE CENTRE OF A TARGET

The Pip Matrox image processing card has only single pixel accuracy. Thus when the Cursor program was first developed single pixel accuracy was accepted for the the centre co-ordinates of the target. The cursor was simply placed on what the user deemed to be the centre of the target and the cursor co-ordinates accepted as the centre co-ordinates of that target.

Problems were caused by the centre of the target not lying in the centre of a pixel but on the border between two pixels or at the intersection point of four pixels as shown in figure 11.8 below. This negatively affected the accuracy with which the control frame co-ordinates could be measured and thus the accuracy of the whole system. Discrepancies in control co-ordinates were as large as 4mm in DZ, with an average of about 1.5mm.

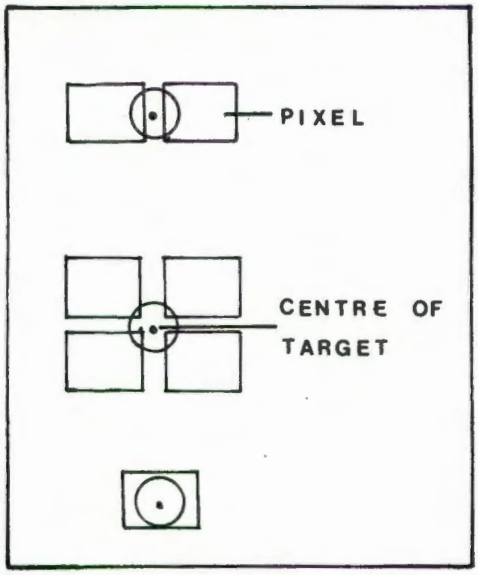

FIGURE 11.8 CENTRE OF TARGET FALLING BETWEEN PIXELS

The traditional Cape Town method showed that the spatial residual errors obtained for discrete targeted torso points seldom exceeded 0.5mm

vector displacement. As the chest movements, specially of infants, during breathing are small, such accuracies are required.

Sub pixel accuracy was required to improve the accuracy of the TAG system. As the pointing accuracy of the cursor, with which the centre of the target was determined, could not be improved, the centre of the target had to be determined by mathematical means using the grey level values of the target. A mathematical routine, a centre of gravity algorithm, was used to determine the centre of the target to sub pixel accuracy.

The centre of gravity algorithm (Swokowski 1983), or otherwise known as the centre of mass algorithm, defines the centre of gravity P(xc,yc) of points P1, P2 ...Pi...Pn as:

$$m * xc = My \quad \text{and} \quad m * yc = Mx$$

where m is defined as the total mass of the system where:

$$m = \sum_{i=1}^{n} m_i$$

Mx and My are defined as the moments of the system with respect to the x-axis and y-axis and respectively are:

$$Mx = \sum_{i=1}^{n} m_i y_i$$

$$My = \sum_{i=1}^{n} m_i x_i$$

where n is the number of particles of masses m1, m2, ...mi ...mn of particles P1(x1,y1), P2(x2,y2),.... ..Pi(xi,yi)...Pn(xn,yn) respectively in the co-ordinate plane

On average the size of the white inner circle of the torso targets span 7 to 8 pixels in length and breadth on the image. Therefore, in the centre of gravity algorithm, a 10 by 10 pixel area is used, whose centre is determined by the user's cursor and deemed to be the approximate centre of the target. In this 10 by 10 area lies the white centre and part of the black outer circle of the target as shown in figure 11.9 below.

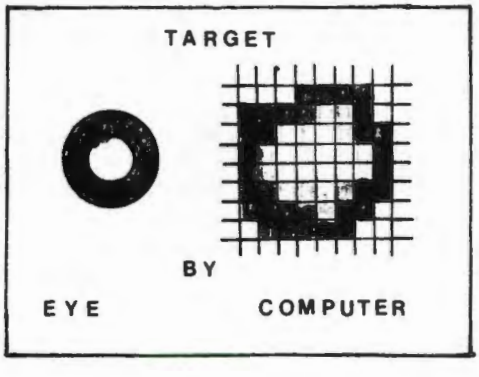

FIGURE 11.9 TARGET IN FORM OF PIXELS

As seen in figure 11.9 the white centre, made up

of "white" blocks/pixels is not uniform. This is due to the fact that light is not uniformly reflected from the target. Thus the brightest pixels, which are at the physical centre of the target, do not lie in the centre of the target. This cannot be detected by the human eye as it cannot distinguish between the grey levels of the white target centre. The computer however can detect the difference and so is instructed to search the 10 by 10 pixel area and find the largest grey value, i.e. the brightest pixel. This pixel's co-ordinates are accepted as the provisional centre co-ordinates of the target for the centre of gravity algorithm.

Included in the 10 by 10 pixel area are also the pixels in the outer black concentric circle of the target. They may appear black to the human eye but in reality vary between black and a dark grey with grey values as high as 70 to 80. As the grey values are used as masses in the centre of gravity algorithm, the grey values in the dark outer concentric circle would adversely effect the calculations. To counteract this problem the grey levels of all pixels with grey levels of less than a hundred are made zero, i.e. they are made black. Thus only the grey levels of the

white centre of the targ et affect the centre of gravity calculation.

Once the centre of gravity for the target has been calculated, the co-ordinates have to be adjusted by 0.5 pixels in both the x and y co-ordinate. This is due to the fact that the co-ordinate system of TAG does not deal in single points but in pixels, which cover a definite area. The xy co-ordinate of the pixel does not refer to the centre of that area but to it's edge, as shown in figure 11.10 below.

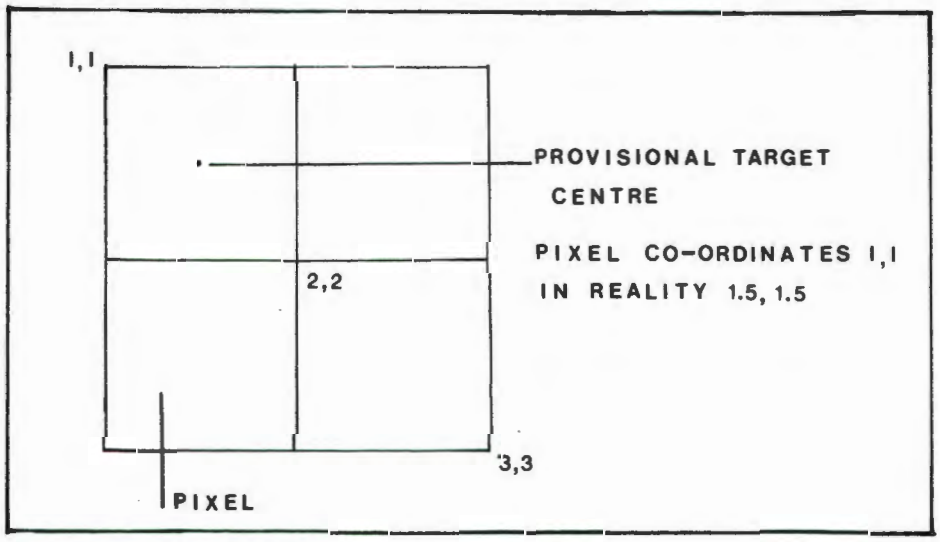

FIGURE 11.10 ADJUSTING CENTRE CO-ORDINATES

The centre of gravity algorithm was first tested by it's repeatability. Several targets were

measured repeatedly with the results being accurate to 0.1 of a pixel.

The use of the algorithm was further verified by the accuracies achieved by the TAG system. A typical set of average uncertainties of accuracy obtained when comparing the surveyed values of the targets on the control frame with those determined by the TAG system are shown in figure 11.11 below for the set up for the baby and child/adult controls and verify the use of the centre of gravity algorithm in the Cursor program,

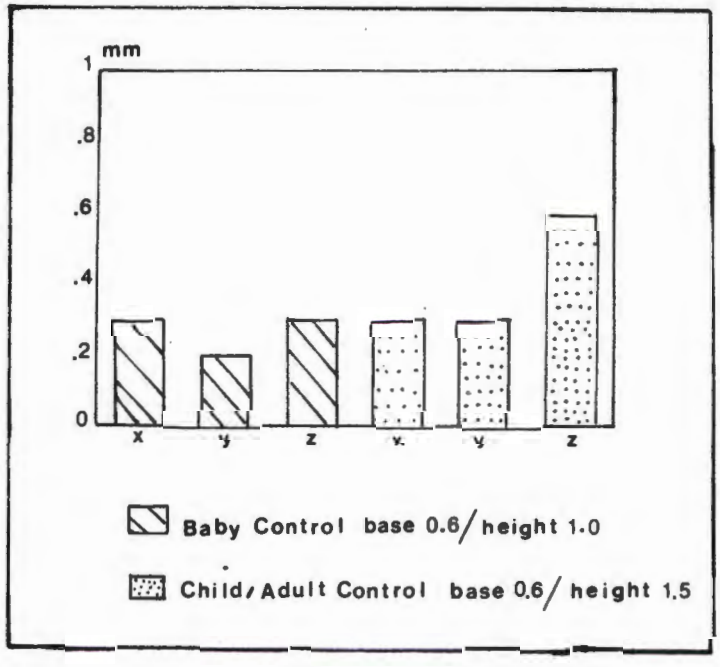

FIGURE 11.11 ACCURACY COMPARISONS AVERAGE X Y Z UNCERTAINTIES

11.3.3 DIGITISING STEREO CONTROL IMAGES

Suboptions 1 and 2 of the digitising menu -

"1. Stereo control image - LEFT " and

"2. Stereo control image - RIGHT "

are used to digitise the control frame image of the video session of a patient. Suboption 1 and 2 digitise the left and right stereo control image respectively.

In both suboptions the user has to enter the name of the patient. The control frame image for that patient is called up from the harddrive and displayed on the left Philips monitor.

To calculate the b parameters of the projective transformation, the xy co-ordinates of the control frame targets of the left and right image of the stereo control image are required in a specific order. The computer must be able to match target number 1 of the left image to the target number 1 on the right image and to the previously surveyed target number 1.

Thus the targets on the left and right control frame images must be digitised in consecutive order to allow for target matching. Due to the design of the control frame of an upper and lower level it does occur that lower control targets may be obscured by the upper part

of the control frame. Therefore it must be possible for the user to identify the obscured target and enter the number of that target into the computer, so that it can delete that number in its consecutive numbering process; e.g. the computer numbers targets from 1 to 18 or 24 depending on what control is used. If the user enters numbers 3,5, and 7 as non visible targets, the computer numbers the targets that have been co-ordinated as 1,2,4,6,8,9,10 etc. . Thus the subsequent text file that is created consists of the correctly numbered and co-ordinated control frame targets.

To assist the user to identify what control frame targets have been obscured, the user is prompted to enter what control frame was used for that patient, i.e. the baby or child/adult control frame. A "help" image of the relevant control frame is displayed on the right hand Philips monitor in which all the control frame targets are numbered. The numbers of non-visible targets on the patient's control image are entered into the computer by the user. (Non-visible targets rather than visible targets were used, as there are usually only a few non-visible targets on each image, thus speeding up the number entering process.)

The differences between suboptions 1 and 2 are that the

part of the stereo control image which is not to be measured is blocked out by a grey square. This ensures that the correct part of the image is measured and written to the correct file. For example if the left image is to be measured the right part of the image on both monitors is blocked out to avoid confusion. The measured data from the the two options, i.e. for the left and right control image are written to different files.

The text files for a particular patient are created as:

1. left control frame image data:

   "Mary" + "cl.txt" = "Marycl.txt"

2. right control frame image data:

   "Mary" + "cr.txt" = "Marycr.txt"

and consist of the number of control frame targets digitised, the relevant numbers and xy co-ordinates of those targets.

11.3.4 DIGITISING OBJECT IMAGES

Suboptions 3 and 4 of the digitising menu -

"3. Stereo object (baby) image - LEFT " and

"4. Stereo object (baby) image - RIGHT "

are used to digitise object images, i.e. images of the patient's torso during respiration. Suboptions 3 and 4 digitise the left and right stereo object images respectively.

In both options the user has to enter the name of the patient, the breath number and the position in the respiratory cycle. The relevant stereo object image is displayed on the Philips monitor.

As the transfer in images from the video machine to the IBM resulted in a shift of the frames, the fiducial marks were introduced. The fiducial mark co-ordinates, with a two point linear transformation, are used to transform the target co-ordinates of subsequent object images to the original object image. Thus the user is asked, before digitising begins, whether the object image he is about to digitise is the first, i.e. the original, object image of that patient. If so, the digitised co-ordinates of the fiducial marks are written to a separate file, which serves as a "control" file. This file is later used to correct possible

shifts in subsequent object images. The text file for the fiducial marks consist of the number of fiducial marks, i.e. 4, and the numbers and co-ordinates of the fiducial marks. The text files are created as follows:

1. left fiducial mark file:

   "Mary" + "4l.txt" = "Mary4l.txt"

2. right fiducial mark file:

   "Mary" + "4r.txt" = "Mary4r.txt"

When digitising the four fiducial marks, they must always be observed in the same order and the convention is to start digitising at the bottom left fiducial mark and continue to the other fiducial marks in a clockwise direction.

The fiducial mark co-ordinates are written to the head of the object text file for that patient and that particular image, i.e.

1. left object file:

   "Mary" + "o" + "1i" +"l.txt" = "Maryo1il.txt"

2. right object file:

   "Mary" + "o" + "1i" +"r.txt" = "Maryo1ir.txt"

Only for object points appearing on both the left and the right image of the stereo object image can three dimensional co-ordinates be calculated. Thus, before

digitising begins, the user has to examine the left and right image and number the object points common to both images. The targets must be numbered and digitised in the same consecutive order in both the left and the right object image due to the format of the projective transformation calculation. Common object points and their numbering should be noted by the user as subsequent object images of that patient should be digitised in exactly the same manner to allow for later comparison between the object images.

The text files for the left and right object images consist of the number of targets and fiducial marks observed on an image, the co-ordinates of the four fiducial marks and the co-ordinates of the object targets.

Suboptions 3 and 4, once completed, return the user to the digitising menu.

11.3.5 DIGITISING SINGLE IMAGES

Suboption 5 of the digitising menu -

" 5. Single image "

was created as a general nonspecific digitiser, as the other digitising options have very specific purposes. Suboption 5 allows the user to measure a single target or check the grey levels on different targets etc..

Suboption 5, unlike other options in the TAG system, is not as user friendly due to its versatility. The user has to enter the full name of the image he wants to digitise e.g. "Maryoli". The point numbers and the co-ordinates observed are written to a file name of the user's choosing. The text file's full name and extension must be entered by the user, e.g."Trash.txt".

In other aspects suboption 5 is the same as the other digitising options in that it uses Cursor for its digitising and thus operates in the same manner as the other digitising options.

On quitting suboption 5 the user is returned to the digitising menu.

11.3.6 DIGITISING THE PATIENT'S OUTLINE

Suboption 6 of the digitising menu -

"6. Patient's outline"

is used to digitise the patient's outline and consists of a modified version of the Cursor program.

The outline serves as a "reference frame" for the plotted vectors on a vectogram. This is more clearly demonstrated in figure 11.12. Part 1 only shows the vectors with no reference to where they are situated on the patient's torso. Part 2 consists of the vectors and the outline, the outline serving as a reference to the vectors. The vectors, now no longer floating in free space as in part 1, can be clearly related to their positions on the patient's torso.

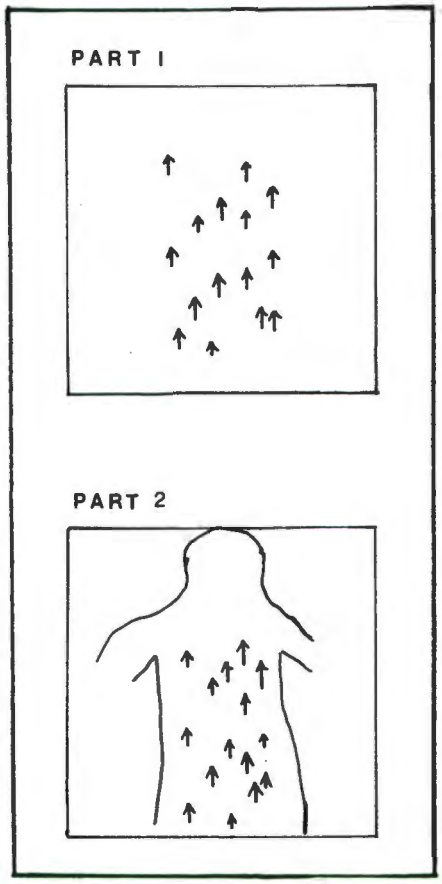

FIGURE 11.12 VECTORS WITHOUT AND WITH OUTLINE

Suboption 6, when called up by the user asks for the patient's name and which stereo object image of that patient is to be used to digitise the patient's outline.

As the outline and the vectors have to lie in the same co-ordinate system, it is necessary to transform the outline from the Pip Matrox co-ordinate system to the patient's co-ordinate system.

A two point linear transformation (described in ch. 8) and the three dimensional co-ordinates of two diagonally opposite points on the patient's torso are used to transform the outline. As the outline only serves as a guide, the minimum of two points needed for the transformation is used.

As three dimensional co-ordinates are required for the two points, the program checks whether three dimensional co-ordinates have been calculated for the targets of that particular stereo object image. If they have not been calculated the program allows the user to go to the calculating menu, calculate the missing data, and return to continue with suboption 6 to digitise the outline.

The user is informed that only one outline is required for the vectograms. Therefore only one outline, i.e. the left or the right outline of a stereo object image of a patient's torso, is to be digitised. The user is then prompted to enter the numbers of the two diagonally opposite points he will digitise, one target in the shoulder region and one target in the opposite hip region.

The user is then given instructions on how the outline digitiser operates. The outline digitiser either

operates in point or stream mode. In point mode only those points that are digitised and entered by the user are written to file. In stream mode every cursor position is written to file. The user can toggle between point and stream mode by using the keyboard keys "z" and "x", where z and x are stream and point mode respectively.

Where suboption 6 differs from the normal digitising options and Cursor is in the movement of the cursor on the monitor. The movement of the cursor is controlled via a mouse.

The user is given on screen instructions, shown in figure 11.13, on how to use the mouse for digitising. The normal movements of the cursor via the keyboard are still operational as an alternative to the mouse. The user can change the speed of the cursor movement by using the left hand mouse key. Either single or five pixel jumps can be used for the cursor movement.

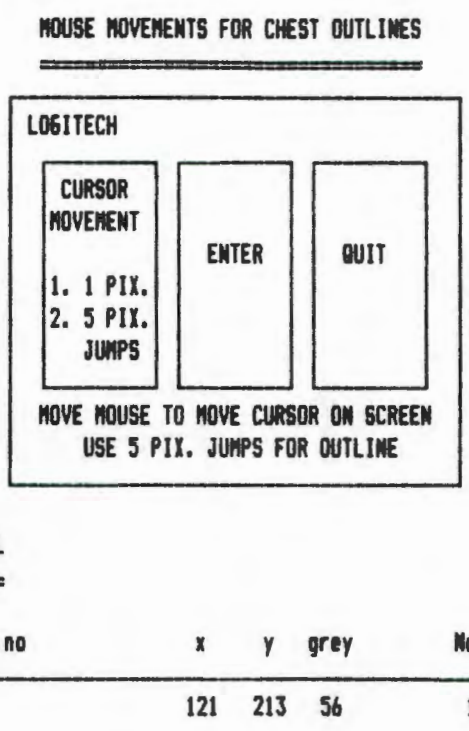

FIGURE 11.13 INITIAL MOUSE INSTRUCTIONS

As can be seen in figure 11.13 above, the user is first prompted to digitise the two diagonally opposite points. The cursor movement is set to point mode at the start of the digitising session.

Once the two points have been digitised the on screen instructions are refreshed. The user can then begin to digitise the patient's outline. The altered on screen instructions are shown in figure 11.14 below.

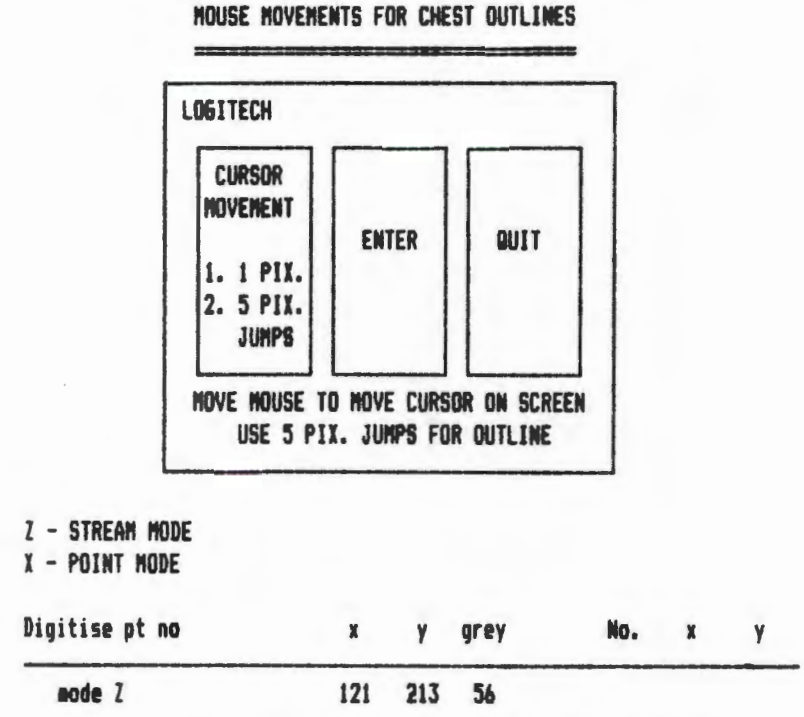

FIGURE 11.14 MODIFIED INSTRUCTIONS FOR MOUSE

The cursor can now be moved to the first section of the outline and its movement set to stream mode. The user can now start to digitise the outline. As the outline is not continuous, for example where only a portion of the upper arm is captured on the frame, the user has to change to point mode, move to the next portion of the outline, set the cursor movement back to stream mode and continue digitising. Thus the patient's whole outline is digitised.

Five pixel cursor jumps should be used where possible on the outline, that is on portions of the outline that are fairly straight. This speeds up the digitising process and also prevents the outline data file becoming too large.

Once digitising is complete, the outline point co-ordinates are transformed to the patient's co-ordinate system using the two point linear transformation.

The number of points making up the outline of the patient and the point co-ordinates are written to a text file, whose name is created by the program as an outline file: "Mary" + "outl.txt" = "Maryoutl.txt"

11.4 CALCULATING THREE DIMENSIONAL CO-ORDINATES FROM THE DIGITISED DATA

Option 3 of the main menu - " Menu : Calculations " - is used to calculate the b parameters for the camera set up for that patient and the three dimensional co-ordinates for the targets on the patient's torso using the projective transformation.

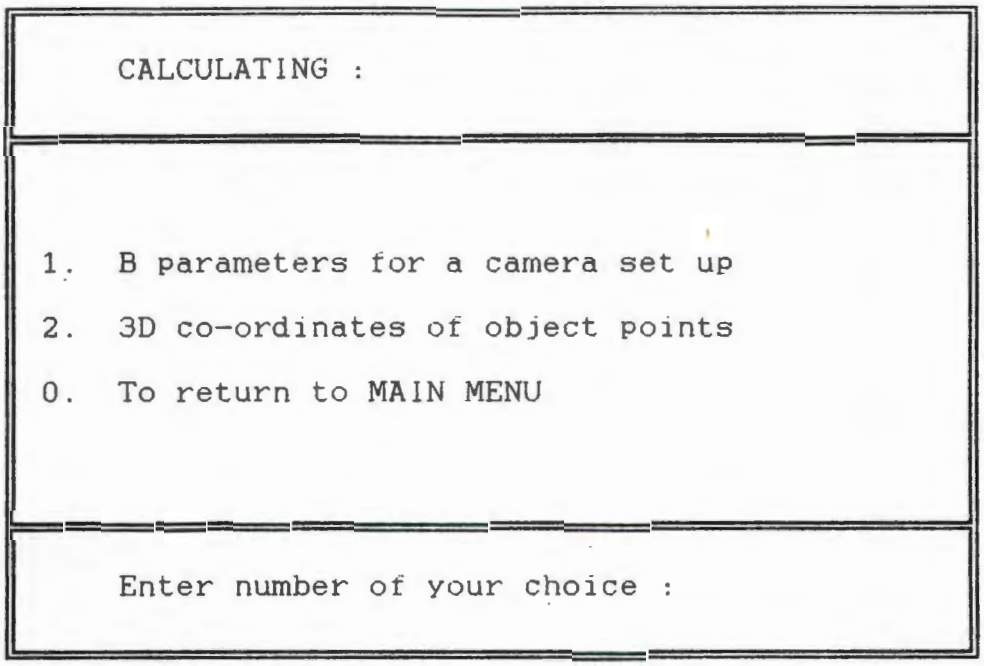

FIGURE 11.15 MENU - CALCULATIONS

Normally the b parameters and the three dimensional co-ordinates are both calculated by the projective transformation in a single program, as only one control and one object image exist.

In the TAG system for a single camera set up there is only one control image but many object images per patient. Thus, to avoid recalculating the b parameters with every object image, the calculations of the projective transformation were split into two programs, one to calculate the b parameters and one to calculate the three dimensional co-ordinates of the targets. Thus for a patient's video session the b parameters only had to be calculated once and then used for all the object images of that patient.

11.4.1 B PARAMETERS FOR THE CAMERA SET UP

Suboption 1 of the calculating menu -

" 1. B parameters for a camera set up "

is used to calculate the b parameters for a particular camera set up for a patient's video session.

On calling suboption 1 the user has to enter:

1. the patient's name, e.g. Mary
2. whether the large or small control was used for that particular camera set up.

3. whether a hard or soft copy of the results is required; hard copy - a print out of results on the printer and soft copy - results are only displayed on screen. If a hard copy is required the user is prompted to switch on the printer.

From the patient's name the computer generates all the necessary file names to call up the digitised data of the left and right control frame images and create new file names for the left and right b parameters, which will be used by suboption 2 of the calculating menu.

File names generated for the b parameters are:

"Mary" + "lhb.txt" = "Marylhb.txt"

"Mary" + "rhb.txt" = "Maryrhb.txt"

To begin the calculations for the b parameters, the two co-ordinate systems, i.e. the Pip Matrox and the patient's system, must be the same.

The co-ordinate system of the Pip Matrox card differs from the patient's co-ordinate system created by the control frame, as the y axes run in opposite directions as shown in figure 11.16 below. The sign on the digitised y co-ordinates of the control frame targets has to be reversed.

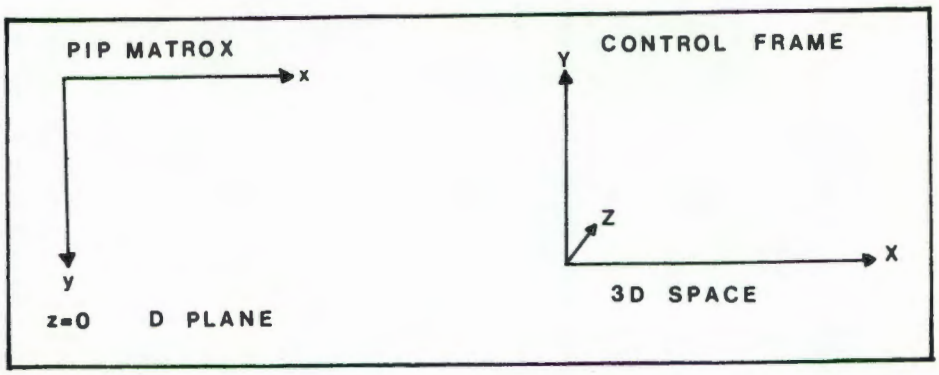

FIGURE 11.16 DIFFERENT CO-ORDINATE AXES

A correction for the scale factor difference in the x and y length of the pixel has to be applied. The digitised two dimensional co-ordinates of the control frame targets are multiplied by the scale factors previously determined, so that the x and y axes are at the same scale.

The two systems are now alike and it is now possible to calculate the b parameters. The digitised left control frame co-ordinates are

used to determine the left camera b parameters, the digitised right control frame co-ordinates the right camera b parameters.

Once the b parameters have been calculated the program calculates the accuracies for determining the three dimensional co-ordinates of the control frame targets. The accuracies of the control frame targets are the differences between a and b, where a and b are:

a: the three dimensional co-ordinates of the control frame targets calculated from the digitised control frame co-ordinates and the b parameters and

b: the accurate three dimensional co-ordinates of the control frame targets previously determined.

The accuracies of the control frame targets reflect the accuracies with which the three dimensional co-ordinates of the targets on the patient's torso can be determined, as these lie within the area that the control frame occupied.

The differences in the two sets of three dimensional co-ordinates is printed out as a check on the accuracy, see figure 11.17 below.

BABY'S NAME : mary

CHECK ON ACCURACY USING COMMON CONTROL POINTS

NO.	DX	DY	DZ	(mm)
1	-0.2	0.4	-1.3	
2	-0.2	0.3	0.7	
3	0.2	1.0	1.5	
4	0.2	0.4	-1.9	
5	-0.0	-0.0	-1.1	
6	-0.0	0.1	1.9	
7	0.0	-0.4	2.0	
8	-0.3	-0.4	0.1	
9	-0.4	-0.3	-0.9	
12	0.4	-0.5	0.4	
13	0.4	-0.6	-0.3	
14	0.3	-0.5	0.0	
15	-0.1	-0.1	-0.2	
16	-0.7	-0.5	-0.9	
17	-0.5	-0.0	-0.7	
18	0.1	0.2	1.9	
19	0.4	-0.2	1.2	
20	0.1	-0.1	-0.4	
21	0.1	0.5	-1.3	
23	0.2	0.7	1.1	
24	-0.6	0.4	0.2	

FIGURE 11.17 PRINTOUT OF THE ACCURACIES OF THE COMMON CONTROL POINTS

The user is then returned to the calculating menu, from where he can begin to calculate three dimensional co-ordinates for the object targets or return to the main menu.

11.4.2 3D CO-ORDINATES OF OBJECT POINTS

Suboption 2 of the calculating menu -

" 2. 3D co-ordinates of object points " -

calculates the three dimensional co-ordinates of the targets on the patient's torso for a particular object image.

On calling suboption 2 the user has to enter:

1. the patient's name, e.g. Mary
2. the breath number and position in the breathing cycle, e.g. 1i
3. whether a hard or soft copy of the results is required. If a hard copy is required the user is prompted to switch on the printer.

The program first ensures that the b parameters for the particular patient have been calculated. If not is returned to the calculating menu so that the b parameters can be calculated.

The program then reads in the following data:

1. the left and right b parameters for the patient's set up
2. the co-ordinates of the fiducial marks of the left and right object image of the first stereo object image of the patient
3. the two dimensional co-ordinates of the fiducial marks and targets of the particular object image of the patient specified by the user.

All text file names are generated from the information the user has entered.

Once all the data has been read, the fiducial mark co-ordinates from the original image and from the object image, e.g. 1i, are used to calculate a two dimensional linear transformation as described in chapter 8. The two dimensional co-ordinates of the targets on the patient's torso are transformed into the system of the original object frame of that patient.

As the two dimensional control frame co-ordinates were adjusted from the Pip Matrox system to the system of the patient, so too have the two dimensional target co-ordinates to be adjusted.

y Co-ordinates have to be multiplied by a negative factor and a scale factor applied to the x and y co-ordinates.

Three dimensional co-ordinates of the targets on the patient's torso for that particular image, e.g.1i, can then be calculated using the b parameters and the transformed digitised two dimensional co-ordinates of the targets. A print out of the set of three dimensional co-ordinates for that particular image are given on the following page in figure 11.18.

BABY'S NAME : mary
BREATH NUMBER AND IN OR OUT : 1i

TRANSFORMED OBJECT POINTS

No.	X	Y	Z	(mm)
1	16.5	456.4	85.1	
2	260.2	479.8	119.9	
3	59.4	408.6	110.2	
4	132.4	417.0	121.0	
5	208.9	401.5	137.6	
6	84.9	385.2	132.7	
7	185.6	382.2	144.4	
8	55.5	364.0	121.8	
9	224.8	359.8	138.7	
10	92.1	340.2	158.2	
11	178.3	338.5	156.9	
12	235.2	322.5	131.8	
13	127.0	303.5	168.5	
14	181.1	299.4	165.8	
15	211.9	295.0	151.0	
16	83.8	283.1	169.2	
17	66.1	244.6	162.1	
18	190.4	248.4	168.5	
19	75.6	182.1	161.0	
20	142.3	192.3	182.1	
21	197.0	175.1	165.8	
22	234.6	208.0	140.5	
23	62.3	121.8	134.1	
24	203.5	120.0	154.3	
25	148.5	51.8	153.8	

FIGURE 11.18 PRINTOUT OF THE THREE DIMENSIONAL CO-ORDINATES OF MARY'S BREATH 1I

The user is returned to the calculating menu. He can continue calculating three dimensional co-ordinates for other object images or return to the main menu.

11.5 GRAPHICAL REPRESENTATION OF THE DATA FROM THE PATIENT'S RESPIRATORY MOVEMENT

Option 4 of the main menu - " Menu : Graphics " - see fig. 11.19 below, is used to represent the 3D co-ordinates of the targets on the patient's torso during a respiratory cycle in graphical form.

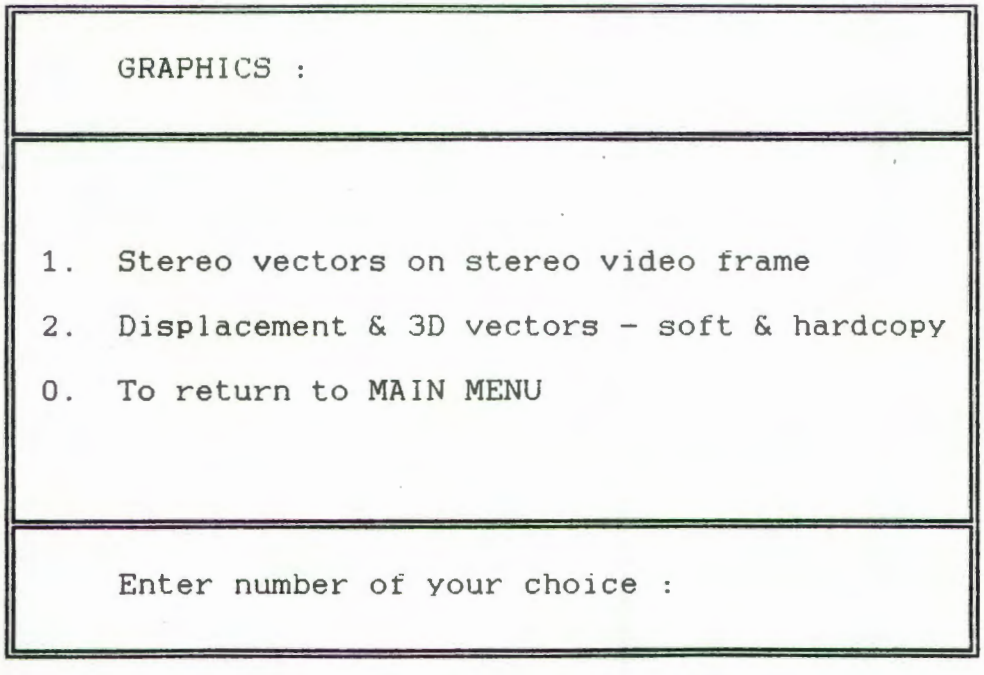

FIGURE 11.19 MENU - GRAPHICS

11.5.1 STEREO VECTORS ON A STEREO VIDEO FRAME

Suboption 1 of the graphics menu -

" 1. Displacement vectors "

produces vectograms, showing how the various targets on the patient's torso move from the extreme breath out to the extreme breath in during a respiratory cycle.

A vector is an obvious way to represent a movement by using a directed line segment, as shown in figure 11.20.

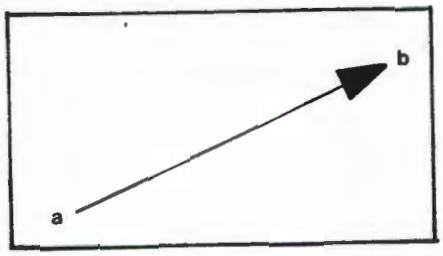

FIGURE 11.20 AN ARROW REPRESENTING A VECTOR

a is the initial and b the terminal point, and ab shows the magnitude and direction of the movement. A can be considered to be a point on the patient's torso at the extreme breath out and B the identical point at the extreme breath in.

A vectogram, see figure 11.21 below, for a particular respiratory cycle is created by joining targets on the left image of a stereopair at the extreme breath out to the identical targets on the left image of the stereopair at the extreme breath in; to complete the vectogram the same procedure is repeated for the right image of the stereopair at extreme breath out and in. The lines joining targets at extreme breath out to extreme breath in are superimposed on the breath out object image.

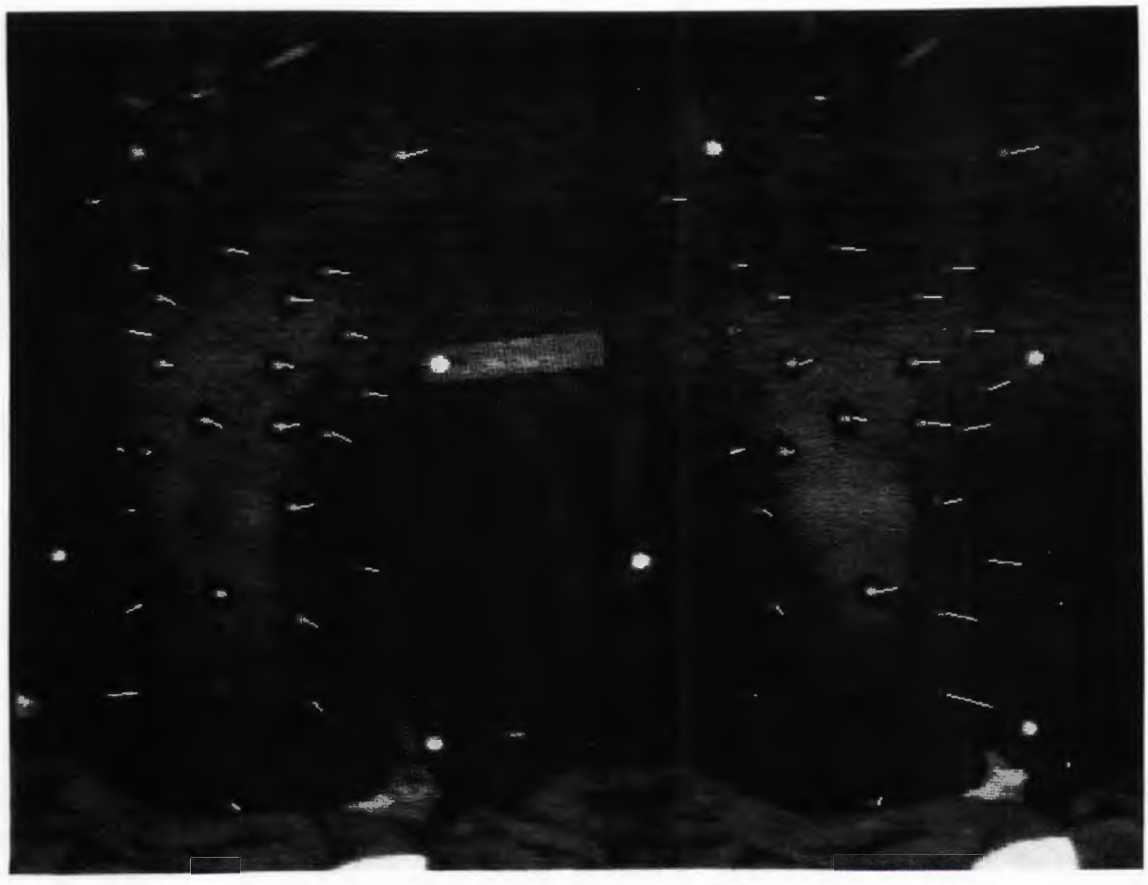

FIGURE 11.21 VECTOGRAM OF THE VIDEO FRAME OF MARY

The vectors can be viewed in three dimensions, when viewed stereoscopically by crossing one's eyes or by using the viewer developed at the University of Cape Town by Prof. L.P. Adams.

On calling suboption 1 the user has to enter:

1. the patient's name
2. the breath in and breath out image of the respiratory cycle, i.e. 1i and 1o of Mary.

When calculating the vectors the frame shift between the extreme breath in and extreme breath out have to be taken into account and corrected. The two point linear transformation and the fiducial mark co-ordinates of the extreme breath in and out are used to transform the extreme breath in co-ordinates into the extreme breath out co-ordinate system.

The vectors, superimposed on the extreme breath out image of the patient's torso, are drawn true to scale. As the true movement of the targets on the torso are very small it is often necessary to view the vectors at a larger magnification. Thus the user is given the option to view the vectors at a scale of his choosing. The image is refreshed and the new vectors plotted. The user

can repeatedly change the magnitude of the vectors until they are satisfactory. To obtain a hardcopy the user can store the vectogram on the IBM's harddrive for a later printout on the Tektronix image printer.

Suboption 1 allows the user to process as many respiratory cycles of a patient as necessary. Once the user has specified that no more respiratory cycles of that patient are to be processed, the user is given the option to produce vectograms for another patient or return to the graphics menu.

11.5.2 DISPLACEMENT AND THREE DIMENSIONAL VECTORS IN GRAPHICAL FORM

Due to the high cost of producing images and vectograms on the Tektronix image printer, suboption 2 was included in the TAG computer system as an alternate form of representing the results in graphical form.

Suboption 2 of the graphical menu plots:

1. XY displacement vectors

2. Z displacement vectors

3. 3D vectors

4. Anaglyph

using two and three dimensional co-ordinates for targets of the extremes of the breath in and out of a respiratory cycle of a patient, with options for on screen viewing and hardcopy outputs on the HP plotter.

Suboption 2 consists of the program Tplt.tru, written in True Basic programming language by A. Tregidga and Prof. L.P.Adams. Tplt.tru, was compiled and bound to form an executable program Tplt.exe, which is run by the TAG system using a C program DOS interrupt.

As the HP plotter and the IBM computer are not compatible, a package Crib, consisting of hardware and software components, had to be used to bridge the gap between the two.

The package Crib (Compusurvis redirection interface board designed by Compusurvis CC.) is a General Purpose Interface Bus (GPIB) interface. The interface card (hardware) is built around the NEC 7210 interface controller chip. The interface driver (software) is designed to allow output

directed at the standard devices supported by the operating system (DOS) - (COM1 to COM2 and LPT1 to LPT2) - to be redirected to the devices on the GPIB . The driver utilises the interface as a controller in order to set up the GPIB and then outputs data as a talker. Thus it is possible, using the driver and the interface, to send plotting instructions, understandable to the HP plotter, to the plotter and obtain the plots required.

On calling suboption 2 the user is prompted to enter:

1. the patient's name
2. the breath number
3. whether the patient's outline is required in the plots.

The text files for the breath in, the breath out and the outline are read in by the program Tplt. From the data the minimum and maximum X and Y are determined by a subroutine Minimax and are used to set up reference frames and windows for screen and printer.

A change in magnitude is also calculated for the displacement and 3D vectors (3*magnitude) by

altering the XYZ co-ordinates of the breath in co-ordinates.

The program gives the user the option of whether or not to view the displacement vectors. For the XY displacement vectors, the vectors are produced by the subroutines vecxy and arrxy which draw the displacement lines and arrow heads respectively.

For the Z displacement vectors, the vectors are produced by the subroutines vecz and arrz which draw the lines and arrow heads respectively.

Both the XY and Z displacement vectors are produced using the three dimensional co-ordinates of the targets. Using the XY co-ordinates of the torso targets at breath out and in the XY displacement vectors are plotted, using their Z co-ordinates the Z displacement vectors are plotted.

The user is then given an option to obtain a hardcopy of the displacement vectors on the HP plotter. Hardcopy outputs of the XY displacement vectors are made by the subroutines plotxy and arrxy and of the Z displacement vectors by the subroutines plotz and arrz.

Hardcopy outputs of the XY and Z displacement vectors are shown in figures 11.22 and 11.23.

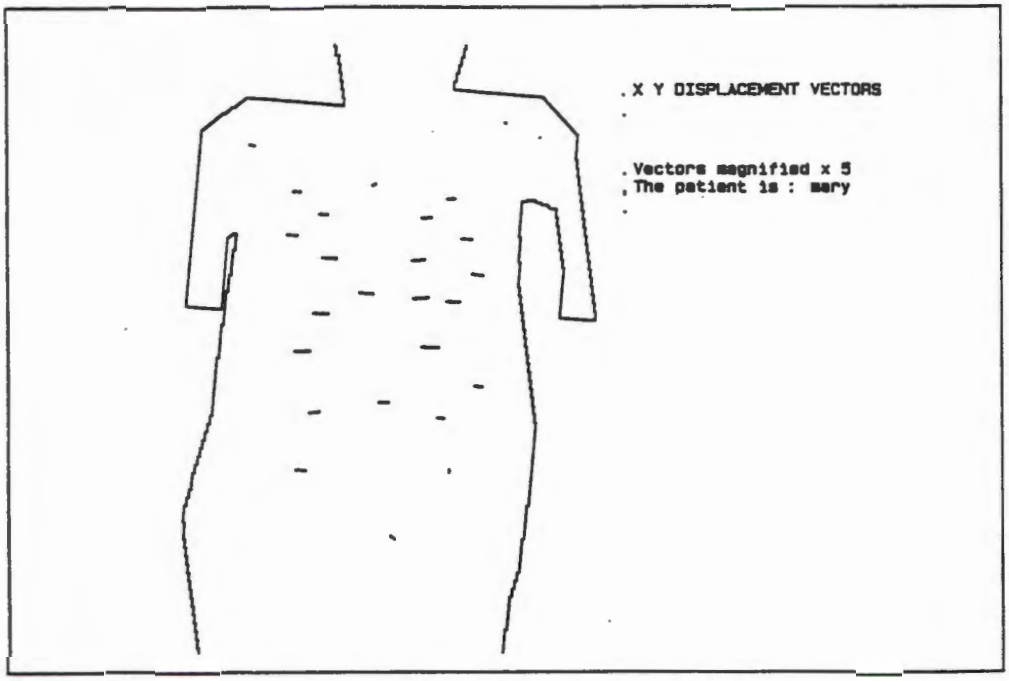

FIGURE 11.22 DISPLACEMENT VECTORS XY

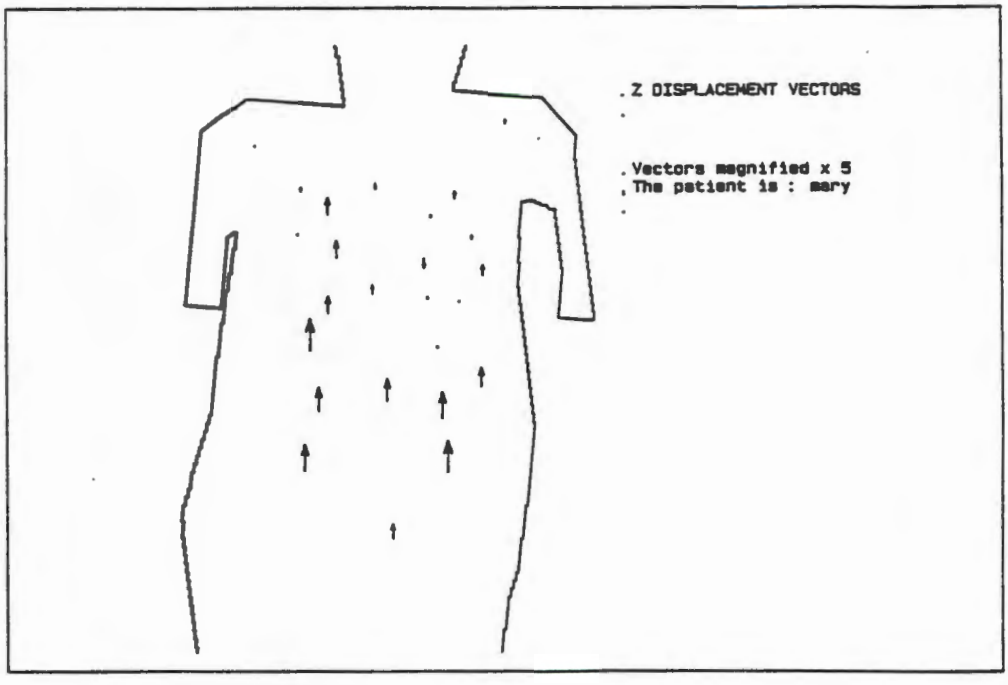

FIGURE 11.23 DISPLACEMENT VECTORS Z

Once the displacement vectors have been completed the user is asked whether or not he wishes to view the stereo vectors.

The stereo vectors plotted by Tplt are not true vectors but pseudo vectors. True vectors are calculated from the true camera set up, i.e. in the TAG system cameras have converging lines of sight. Pseudo vectors, on the other hand, are calculated from "false" camera positions,

cameras set up with parallel lines of sight. Pseudo stereo vectors simplify the viewing of the vectors in three dimensions, but should not be used for accurate measurements. They should only be considered as a visual aid.

The pseudo vector are derived from the parallax bar equations:

$$Px = xl - xr \quad \text{and} \quad Px = (f * B) / (H - h) \qquad (1)$$

Px - absolute parallax of point P

- full definition of absolute parallax is "a ground point lying in the area covered by the overlap of a pair of photographs will possess an image on each photo. The absolute stereoscopic parallax of the point is the algebraic difference, in the direction of the air base, of the distance of its two images from the principle points of their respective photos. The difference between two such absolute measurements is known as a Parallax Difference."

P - point P on the ground having three dimensional co-ordinates X,Y,Z

- point P on the left and right image of a stereo pair having co-ordinates xl,yl and

xr,yr for the left and right image respectively

f - focal length of the two cameras, the two cameras being of the same make

B - camera base, i.e. distance between the two camera stations

H - camera height above the datum (sea level)

h - height of point P above datum

- h of P = Z of P

The equations for the points making up the vectors on the left pseudo stereo image are:

$$xl = (X * f) / (H - Z) \quad (2)$$

$$yl = (Y * f) / (H - Z) \quad (3)$$

and for the points making up the vectors on the right pseudo stereo image are:

$$xr = (xl - f * B) / (H - Z) \quad (4)$$

$$yr = yl \quad (5)$$

Equations two and three are derived from the same figure from which the parallax bar equations are derived, i.e. figure 11.24.

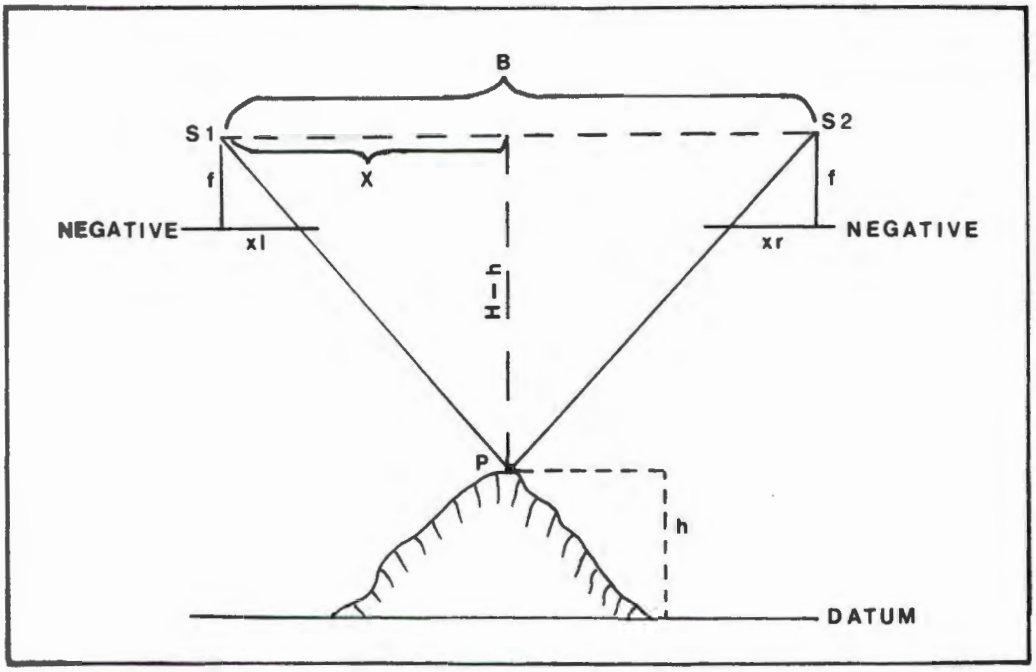

FIGURE 11.24 PARALLAX BAR DIAGRAM

In figure 11.24, S1 and S2 are the camera stations, with S1 being the origin of the system. By using similar triangles it can be shown that:

$$xl / X = f / (H-h)$$

By rearranging this equation, equation (2) is derived.

By changing the perspective in figure 11.24 by 90 degrees, i.e. instead of looking down the y axis onto the x axis one is now looking down the x axis on to the y axis,it is possible to obtain equation (3) in the same manner as equation (2). The different perspectives are illustrated in figure 11.25 below.

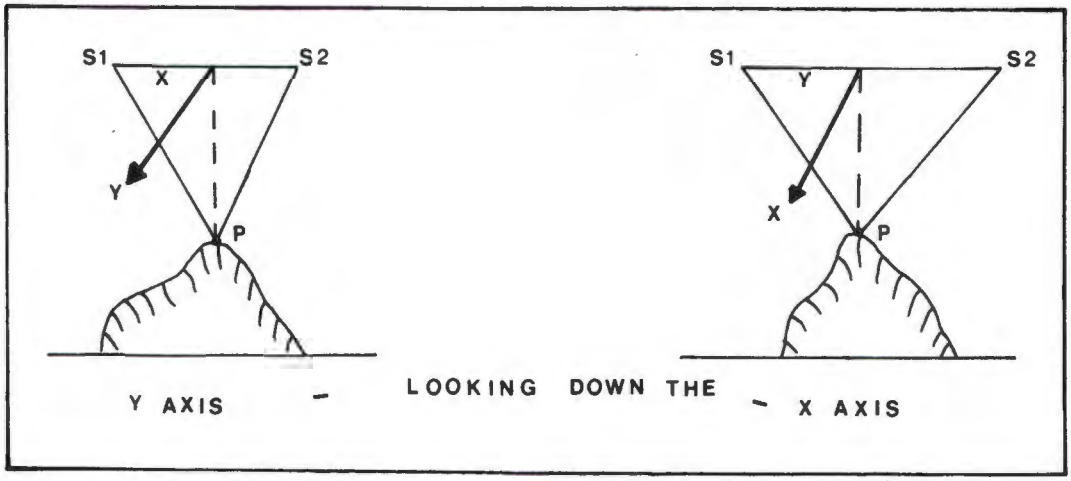

FIGURE 11.25 DIFFERENT PERSPECTIVES OF PARALLAX SET UP

Once xl and yl have been calculated, it is possible to use equation (4) to solve for xr. As the stereo vectograms serve as visual aids, yl is made equal to yr in equation (5). Thus the vector depicted on the left and right image lie at the same eye level allowing the viewer to view the vector in three dimensions.

As only the right or the left image of the stereo pair can be plotted on the screen, a hardcopy of the two images is produced. Thus it is possible to place the two images under a stereoscope and perceive the plotted vectors in three dimensions.

The stereo vectors are shown in figures 11.26 and 11.27 on the following page and can be viewed in three dimensions.

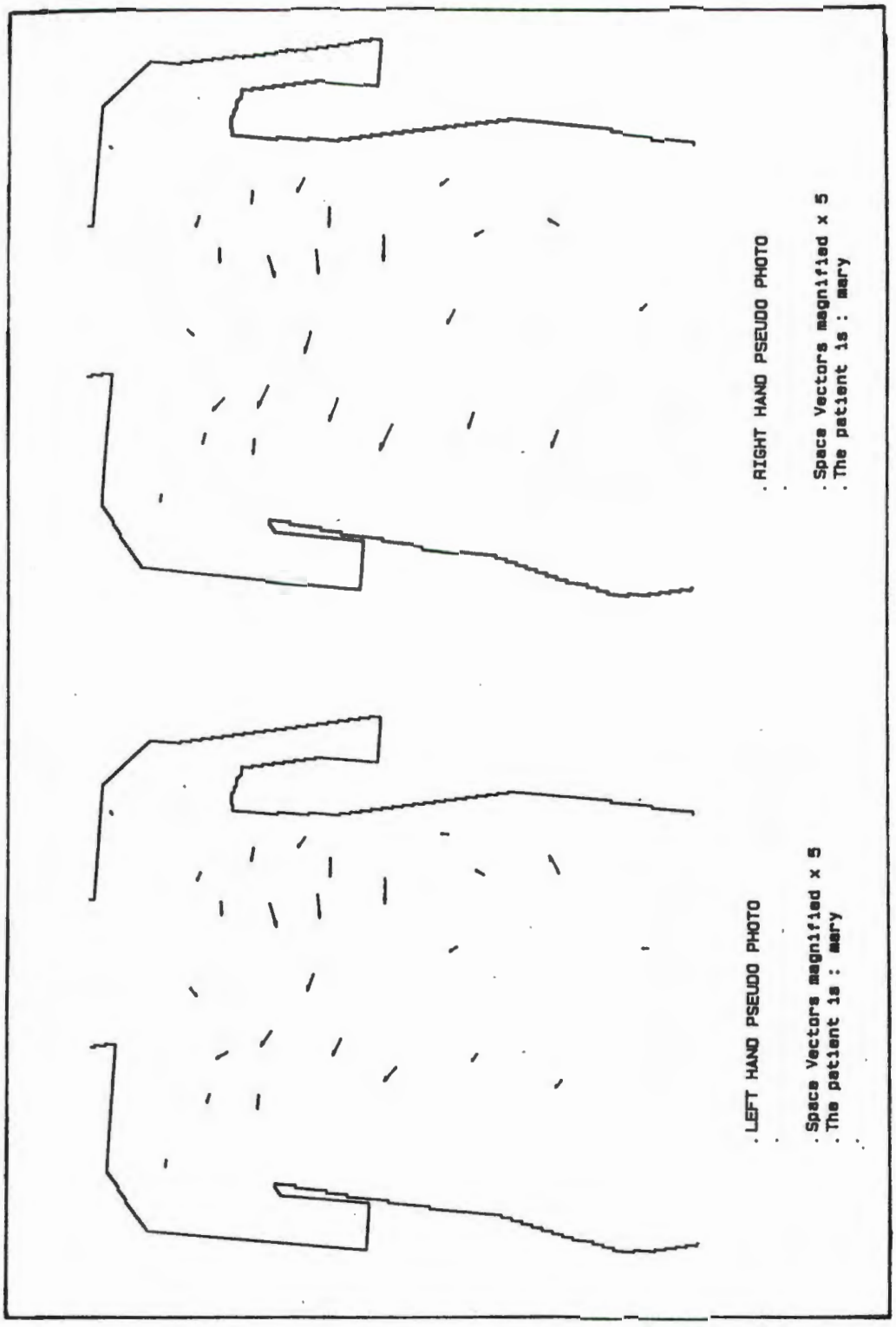

FIGURE 11.26

LEFT HAND PSEUDO PHOTO

FIGURE 11.27

RIGHT HAND PSEUDO PHOTO

The last option in Tplt consists of an anaglyph. An anaglyph consists of the left and the right pseudo vectors, plotted in red and green respectively, on the same screen. A special pair of spectacles, the two lenses tinted the same green and red colour as the green and red vectors respectively, is used to view the vectors in three dimensions. The human eye can only perceive the green vectors through the red lens and the red vectors through the green lens. Thus the red and green vectors are viewed stereoscopically and the brain perceives a single set of vectors in three dimensions.

There is no hardcopy option for the anaglyph, as the green and red ink will never have the same shades from batch to batch. If the green shade of the plotted vectors differs from the green shade of the lens, the eye will see both the green and red vectors through one eye. Thus the stereoscopic effect would be destroyed and therefore also the three dimensional perception of the vectors.

Once suboption 2 has been completed, the user is returned to the graphics menu.

11.6 IMAGE PRINTING ON THE TEKTRONIX IMAGE PRINTER

Option 5 of the main menu -

" 5. Image printing to Tektronix " -

allows the user to send images, stored on the IBM via the Pip Matrox card, to the Tektronix image printer for printing.

Problems that had to be overcome to print the images were:

1. the IBM computer and the Tektronix printer are not compatible
2. the Tektronix printer is a colour printer and the images produced by the Ikegami cameras are black and white.

A program, written in C language to run on DOS for the company Tektronix by N.Parkyn, specifically overcomes these problems. An image on the harddrive is called up by the program and the data altered to suit the Tektronix image printer. A set of DOS commands then sends the altered data of the image to the Tektronix image printer.

A "shell" program was written so that both the program and the sending of the altered data of the image to the image printer could be run by the TAG system,

without the user having to know the operating process.

On calling option 5 of the main menu, the user has to enter the patient's name. An image menu is then displayed on the screen, see figure 11.28 below.

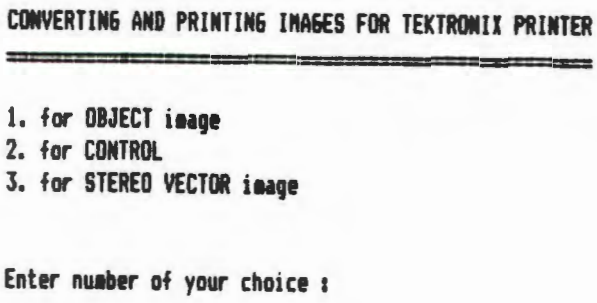

FIGURE 11.28 IMAGE MENU IN PROGRAM TIMP

The user selects the image to be printed and follows the simple on screen instructions on how to operate the image printer, e.g. connect and switch on the printer.

Once the image is printed out the user is returned to the image menu to select another image or return to the main menu.

12. ANOTHER PATIENT EXAMPLE OBSERVED USING TAG

Another example is given by the patient Veronica. In the case of Veronica we the photogrammetrists did not know the state of her health and the condition of her respiratory muscles.

It can be clearly seen from the XY and Z displacement vectors (fig. 12.1 & 12.2), from the stereo pseudo photos (fig. 12.3) and the vectogram (fig 12.4) that the diaphragm is pulling from left to right.

Although it is possible to make such observations as photogrammetrist and take an interest in the results, it is up to the medical staff to draw medical conclusions from the results; the photogrammetrists' work is to develop the system so that the medical staff can obtain those results quickly and efficiently for evaluating the patient's condition.

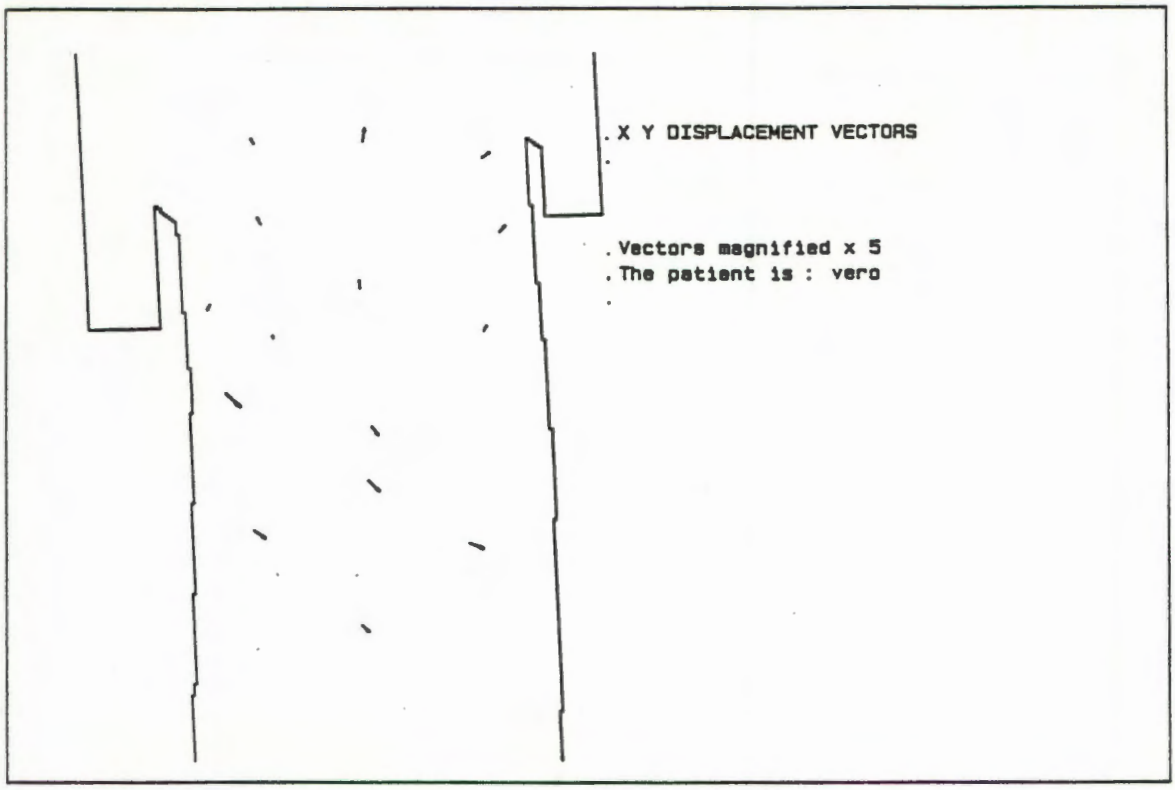

FIGURE 12.1 XY DISPLACEMENT VECTORS OF VERONICA

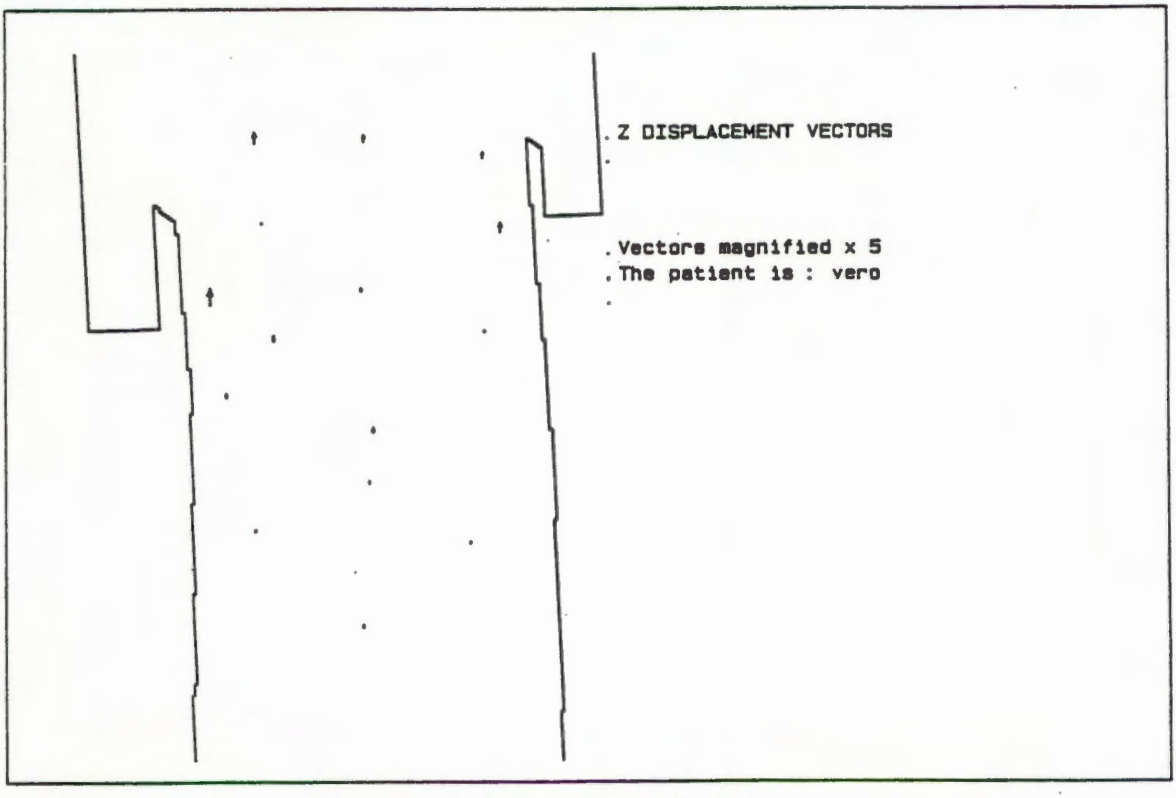

FIGURE 12.2 Z DISPLACEMENT VECTORS OF VERONICA

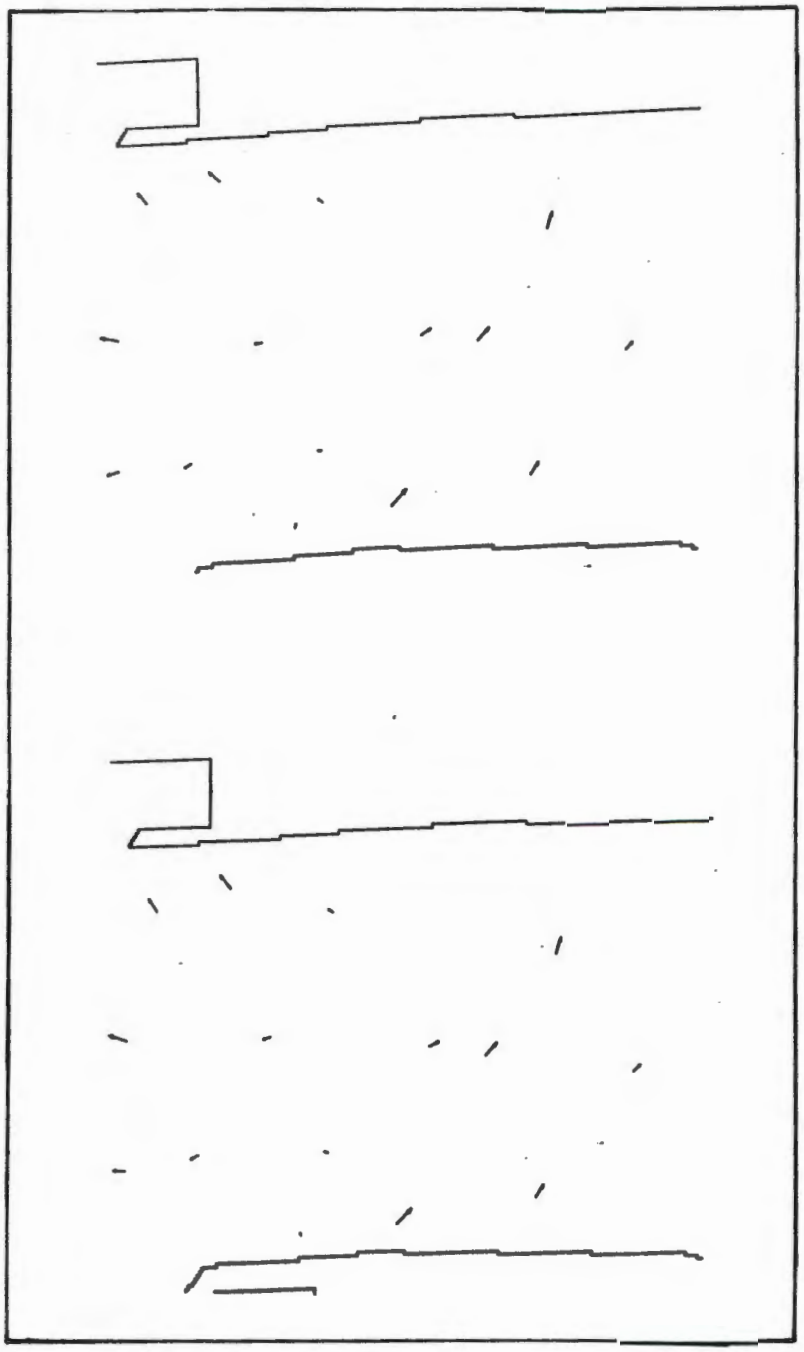

FIGURE 12.3 LEFT AND RIGHT PSEUDO PHOTOS OF VERONICA

(set up to be viewed in stereoscopic mode)

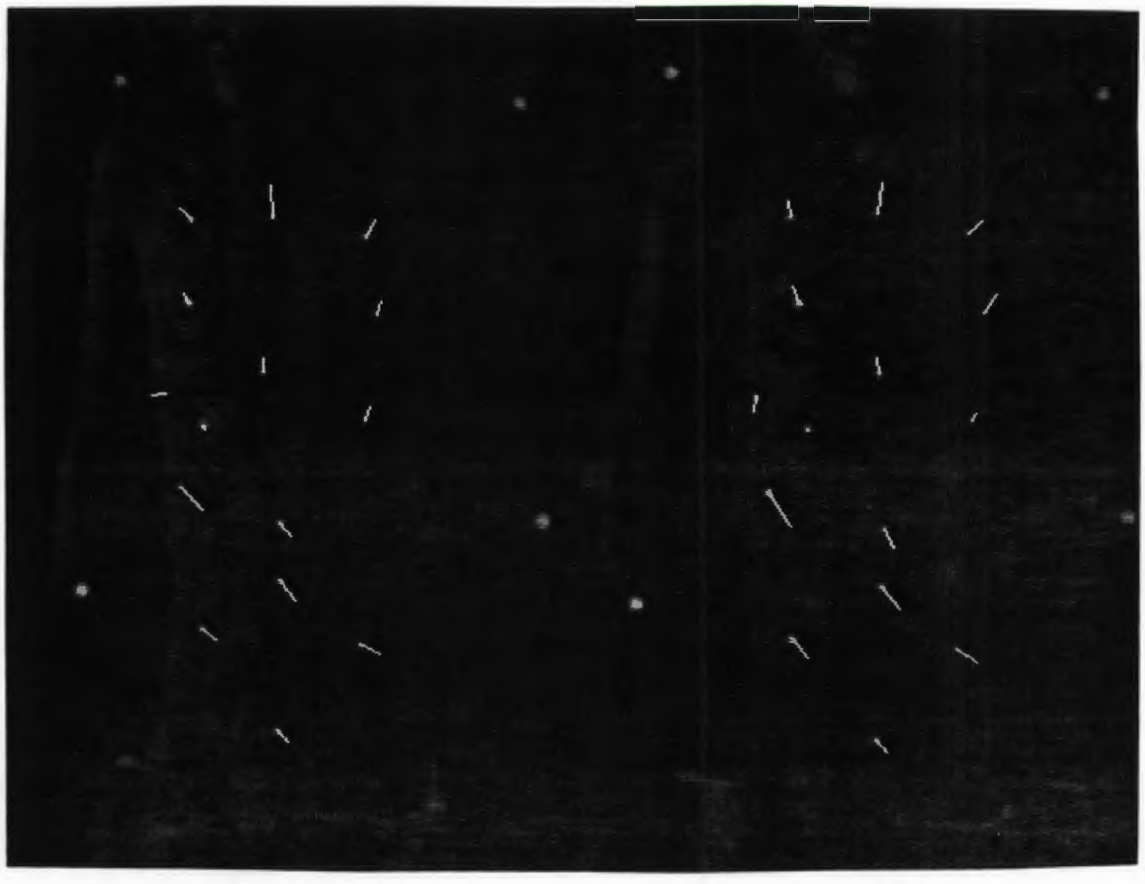

FIGURE 12.4 VECTOGRAM OF VERONICA

13. PROBLEMS EXPERIENCED IN THE RESEARCH

Several problems were experienced during the making of the TAG system. Several ideas were implemented, only to be discarded at a later stage. Some of the ideas had to be altered to fit in with the developing TAG system. Other areas proved to be areas of difficulty and concern. All these are discussed in this chapter.

13.1. EXPERIMENTATION WITH DIFFERENT TARGET TYPES

A suitable target for marking the discrete points on the patient's torso had to be found.

The first targets were small black dots, about half a centimeter in diameter, which were attached to the test subject's chest - a doll "Debby". The targets were not visible on the monitors, as the black targets were "swamped" by white light reflected from her light coloured chest. The target size was increased to prevent the "swamping" effect. The target however had to be very large to appear on the monitor and as the white light did not uniformly cover the black target, it was difficult to determine the centre of the target accurately.

The second type of target used were retro-reflective targets. Retro-reflective targets can be thought of as mirrors that reflect the light back to its source and are made of similar material to that of red reflectors used on a bicycle that reflect the light of a car's headlamps back to the driver so that he is aware of the cyclist. The retro-reflective targets affixed to "Debby's" appeared as bright white light against her light skin.

The advantage of using such targets is that the lighting is not critical. Even in normal lighting the targets appear clearly on the monitor, with some exceptions. The surface of the retro-reflective target must be approximately perpendicular to the beam of light for it to reflect the light back correctly. Due to the patient's torso shape this caused several problems, which were partially overcome by using a lamp set up midway between the two cameras to illuminate the retro-reflective targets.

The targets on the very edge of the patient's torso still caused problems. The light being reflected off the target at an oblique angle made the target appear too dimly on the video image for the centre of gravity routine to operate.

The intensity of the lamp used was increased. The outer targets now were bright enough for the centre of gravity routine. Targets however that had previously been reflecting the light correctly now appeared as a blazing circle of light. This light was not reflected evenly in all directions away from the target and thus the centre of the blazing circle of light was not always the centre of the target.

The third type of targets to be used consisted of white painted inner concentric circles and a matt black painted outer concentric circle.

No matter what amount of light is reflected from the surface to which the targets are attached, the light from the white inner concentric circle is not "swamped" due to the matt black outer concentric circle. The white inner circle reflects the light well, without having the glaring intensity of a retro-reflective target. Thus both the white inner circle and matt black outer circle appear on the video image under the correct lighting conditions.

When working with CCD cameras and targets the lighting is always a critical factor. White is dominant, reflecting the light with a much larger intensity than the black. The light from a white target spreads away

from the target making the target appear much larger and in the case of a retro-reflective target it can appear huge. A black target on the other hand can be considered as non-reflective.

The direction of the light also plays a major role. In determining the path of light rays, it is a known fact that the angle of incidence is equal to the angle of reflection. A white surface illuminated from light rays perpendicular to the surface will make the surface appear a brilliant white as the light is reflected directly back from the surface. Light rays reflecting off the same surface at an acute angle will make the surface appear off white, grey or nearly non visible depending on the angle of the light when it strikes the surface.

The effect of the different directions of incident light is shown in figure 13.1 below.

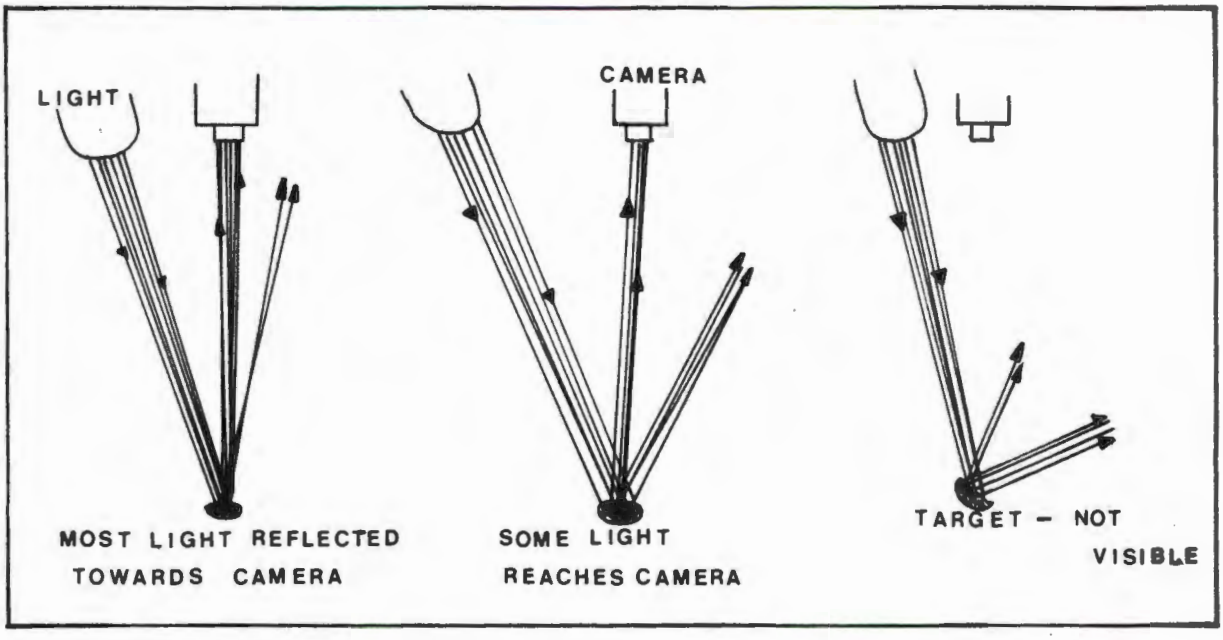

FIGURE 13.1 DIFFERENT ANGLES OF LIGHT AFFECTING TARGET VISIBILITY

With the correct targets - black and white targets - and the correct lighting - controlled by the dimmer light switch - it is possible to achieve a good video image of the patient's torso and the targets on the torso.

13.2 IMAGE SATURATION DUE TO SYNCHRONISATION COMPRESSION

Image saturation due to synchronisation compression is caused by surfaces reflecting a large amount of very bright light. The diodes in the image processing chip of the camera receive so much white light that sensory overload occurs and the synchronisation signal from the camera is distorted. The resultant stereo image can be as badly affected as figure 13.2 shown below.

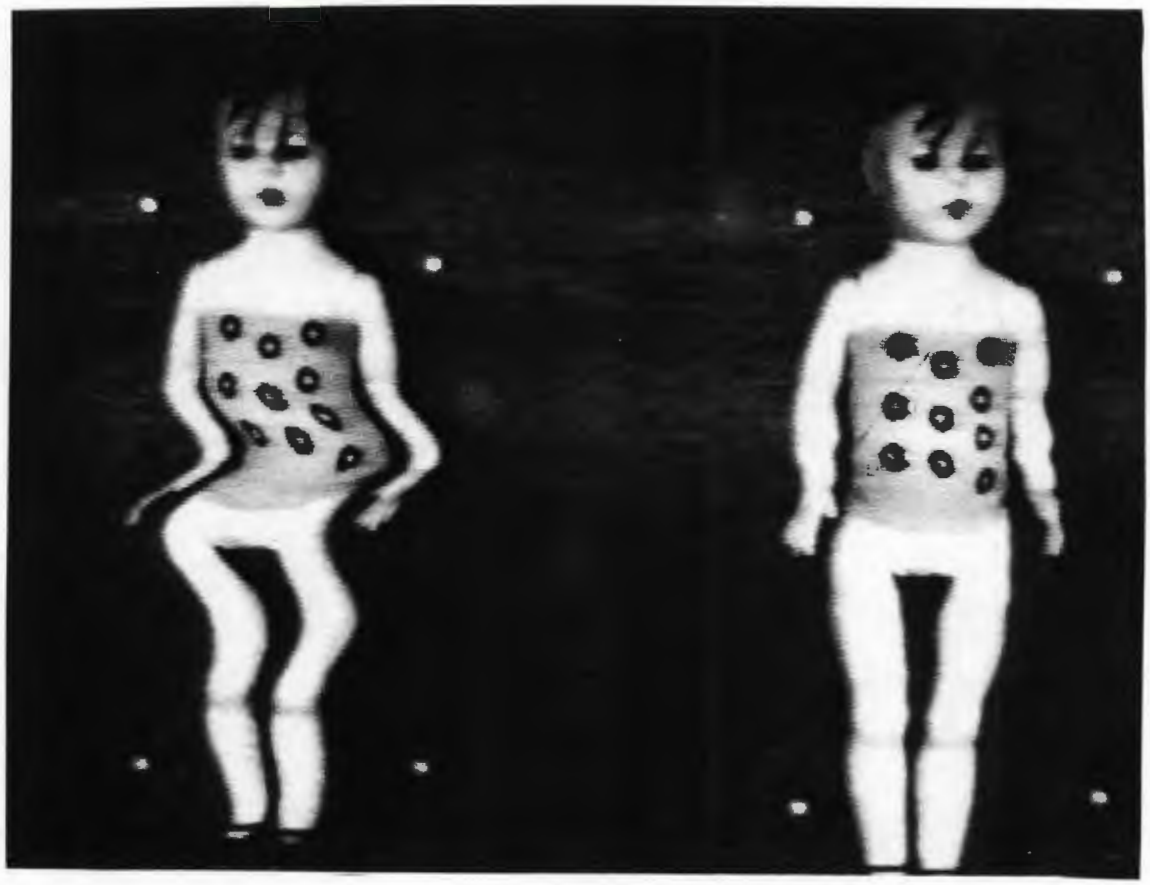

FIGURE 13.2 DEBBY DOLL IMAGE DISTORTED DUE TO IMAGE SATURATION

The first time the problem was encountered it was caused by light reflecting off the aluminum ladder used to reach the cameras on the gantry. As the light from the ladder appeared in the part of the image not shown on the mixed stereo image, it was difficult to deduce the source of the problem.

Once the ladder had been moved, the glare from the aluminum ladder and thus the synchronisation compression was removed.

Other substances that have caused synchronisation compression due to reflecting large amounts of light are white materials such as the white clothing medical staff wear.

As precautions during video sessions the cameras are shielded partially from peripheral bright light by lens shields. Any synchronisation compression that still occurs is immediately visible due to the distortion of the video image on the monitor. The object causing the glare must be moved. It is not necessary in all cases to remove the object, such as the aluminium ladder, but only to turn the object so that the glare of the object is deflected away from the camera.

13.3 LIGHTING PROBLEMS DURING A VIDEO SESSION

Light plays a critical role when using CCD cameras to capture the patient respiratory motions.

When working with light and CCD cameras several factors always have to be taken into account:

1. light intensity
2. surfaces from which the light reflects
3. the angles of incidence and reflection of that light

The light intensity is partly controlled and partly uncontrolled. The natural light forms the uncontrollable element. Usually in a ward the light from the windows is coming from one direction only and thus reflects more light into the one camera than the other making the one image appear much brighter.

The overhead lighting and the light suspended midway between the two cameras forms the controllable element. These lights and the aperture settings of the two cameras can be manipulated until the lighting on both the left and right image are approximately equal and satisfactory. Due to the synchronisation of the two cameras, setting the aperture on the one camera affects the amount of light received by the other

camera. Thus the aperture settings on the two cameras should be set in conjunction with one another and not as separate entities.

The surfaces that scatter light affect the images received by the camera. White surfaces which reflect a large amount of light affect the images adversely as this light often overshadows other objects. Thus a normal white hospital sheet to cover the mattress is not feasible. A matt black sheet is used, which is non reflective and thus does not affect the images.

An example of the power of white surfaces is a white hospital blanket that was used to cover a patient's legs during a video session. It was found that when the blanket was placed in the camera's field of view, the rest of the image immediately darkened due to the strong light reflected off the blank.

The only objects that are of interest during a video session are

1. the patient's torso
2. the targets on the torso
3. the fiducial marks
4. the targets on the control frame

All other surrounding objects and surfaces should not affect the image. Thus the mattress is covered in a matt black sheet, the fiducial marks are attached to bars that are painted matt black and the control frame is painted matt black and thus can not effect the video image adversely as matt black is "non-reflective".

13.4 ALTERING THE PROGRAM TO DIGITISE THE PATIENT'S OUTLINE

The outline of the patient is presently digitised by the program Outline. The outline digitised however does not fully follow the outline of the patient and at times has a rather rectangular appearance due to the coarse image grid of the Pip Matrox card. The outline obtained from the program Outline does however serve its purpose as a reference frame for the plotted vectors.

The Department of Surveying has however recently obtained a sonic digitiser, which in future can be used to digitise the patient's outline. The sonic digitiser operates on the sound which a computer mouse sends out. The distance that the sound travels to each

one of the two sonic receivers is used to calculate the mousses position (x,y) by intersection.

The sonic digitiser can be suspended directly over the monitor and the patient's outline digitised directly from the patient's image on the monitor. On the other hand the sonic digitiser could be laid flat on a table and the patient's outline digitised from a Tektronix image print of the patient.

With the sonic digitiser a much smoother curve for the patient's outline can be achieved.

13.5 MOVEMENT OF THE BED AND THUS THE CAMERAS WHEN LIFTING THE PATIENT ON AND OFF THE BED

When moving the patient on and off the bed it is difficult not to disturb the bed. A test was carried out to see whether the movement of the bed had effected the cameras and thus altered the cameras' b parameters.

The control frame was videoed before and after a video session of a patient. The first control was used to calculate the b parameters of the cameras. The second

control was treated as an object and the co-ordinates of the targets calculated using the b parameters from the first control.

From comparing the three dimensional co-ordinates calculated for the control and those previously determined, it was apparent that no significant movement of the cameras had taken place. It was therefore established that the small movement of the cameras that occurs when a patient is moved on and off the bed, not only affects the cameras but the whole unit, i.e. the bed, gantry and cameras all move as a rigid unit. Thus as the whole unit moves as one, the relation between the cameras and the surface of the bed remains the same. The accuracy of the system was therefore not affected.

13.6 DESIGN ADJUSTMENTS TO FACILITATE BETTER OPERATION

In designing such a system for the first time, there are always elements that should be changed on future models.

1. To set the cameras up correctly, the orientation often has to be changed. At present the clamp mechanism used to hold the cameras in place is tightened and loosened by an Allen key. This is not satisfactory, as the cameras are usually positioned above the operators' head and make the tightening and loosening with the Allen key very difficult. One should be able to fasten and loosen the cameras with a single handgrip by using a mechanism such as a large wing nut. This could be operated with a single handgrip instead of using an Allen key.

2. The cameras are mounted on a cross bar, which is welded to the gantry. As the patients are not all of the same size, the patient often has to be moved down the bed so that the torso of the patient is directly below the cameras. If the patient is chronically ill all movement should be avoided. It would thus be far better if the bar holding the cameras could be moved along the gantry to position the cameras above the patient's torso.

3. A minor problem is the cupboard that is built below the bed. One always underestimates the amount of space that is required for equipment. Therefore it would be better if the cupboard spanned the whole length of the bed.

14. OTHER APPLICATIONS FOR THE TAG SYSTEM

Although TAG has been specifically designed for evaluating regional body surface motion during respiration, it has a wide range of potential applications.

Possible applications, also in the medical field, include the studies of the effects of growth and development on spinal and chest deformities and of lung and heart disease on the mechanics of breathing.

At present the TAG system is being modified to study the motion of brain damaged subjects and to analyze gait.

14.1 TAG IN THE STUDY OF BRAIN DAMAGED SUBJECTS

The study of brain damaged patients can include a study to determine a specific pattern of motion for brain damaged subjects.

It is hypothesized that a normal subject and a brain damaged subject will stand up and sit down in different ways and that from the pattern in which a subject stands up and sits down that presence of brain

damage can be determined and also the type of damage that exists.

Previous studies on this subject do show that the hypothesis may be true, but as in the study of respiratory motion the time consuming traditional photogrammetric method has seriously limited the number of studies on which to base this hypothesis. With a modified version of the TAG system it will be possible to study rapidly a large number of patients from which to derive an accurate hypothesis.

The pattern of movement of a subject is studied by attaching a target to the patient's forehead. In the traditional method the target, a flashing LED (light emitting diode) attached to the subject's forehead by a sweatband, was photographed by two cameras in stereo. The two Mamiya cameras used cut film with only four negative holders. After each patient had been photographed, the negative holders had to be taken into the dark room and reloaded. For the camera settings a long exposure and a low aperture setting were used. The resultant pair of negatives consisted of a series of white dots which showed either the standing up or sitting down movement of the patient. Also depicted on the negatives were the control frame targets also made of LEDs. The dots on the left image

had to matched to their stereo counter parts on the right image using a stereoscope and the matched pairs numbered. Three dimensional co-ordinates for the dots were obtained by placing the negatives in the stereo comparator, linked to the HP computer. This process had to be repeated for every subject and is very time consuming.

Using a modified version of the TAG system, it will be possible to set up the control at a centre such as a cerebral palsy school and a large number of subjects processed within a very short time. The video tape could then be transported to the Department of Surveying for quick analysis.

The advantage, as with the respiratory motion, is that the whole motion is captured as a permanent record on tape. The motion can be repeatedly observed by simply rewinding the video tape. The viewing speed can also be altered to facilitate analysis. The viewing of the actual movement of brain damaged subjects compared to normal subjects alone should bring a new insight to this field of study and may even in future be of use in determining the type and extent of the damage to the brain.

14.2 TAG IN THE STUDY OF GAIT ANALYSIS

Sport has always interested man and with the large amount of interest shown in running as demonstrated by such marathons as the Comrades and the Two Oceans, more and more indepth study is being done on the actual running motion.

As with the motion study of brain damaged subjects, so has the running motion been studied by the traditional photogrammetric methods. The pattern of the study resembles that of the brain damaged subjects, with some exceptions.

In the gait analysis retro-reflective targets attached to the back of the leg were illuminated using a strobe light. The subject was photographed running on a tread mill within a large control frame. The light from the LED control targets and the light from repeatedly illuminated retro-reflective targets during a single stride were captured on a single negative using a long exposure. The method for obtaining the three

dimensional co-ordinates of the leg targets was the same as in the motion study of brain damaged subjects

As with the motion study of brain damaged subjects so will the gait analysis be enhanced by using a modified version of the TAG system. The study of the gait will be enhanced by being able to review the motion again and again at any speed desirable.

CONCLUSIONS

The anatomy textbook for studying the shape and make up of the body is correct in every aspect, except in that it depicts the body as static without movement. From a single flat page it is impossible to convey the intricate type of movement of which the human body is capable.

It is often the very motion of the body that will show up the injuries and damage it has sustained either from illness or from the very movement of the body itself.

In the case of the study of regional body surface motion, it is the movement of the torso, due to the intercostal muscles and the diaphragm, that show what injuries the lungs have sustained due to illnesses such as pneumonia. And it is the movement of the torso, viewed in three dimensions, that can give insight into what diagnosis of the patient's ailment should be made and therefore his treatment.

To diagnose the patient's condition the patient only has to be moved to the special hospital rig and the video session of his respiratory motion completed. The patient does not have to be disturbed again, whereas the doctor can study the respiratory motion of the patient in three dimensions repeatedly from the video tape. The video tape

also serves as a permanent record for the patient and can be compared with future video sessions of that patient after treatment is underway or completed.

In the motion study of brain damaged subjects the movement of the subject, viewed in three dimensions, may give insight into different types of brain damage and how these effect the individual's motion and how this may be used in future as a diagnostic tool.

In the gait analysis the movement of the runner, viewed in three dimensions, could give valuable insight into the running style faults, such as supernation or pronation, which cause so many injuries in runners. The effect of changed foot wear, such as built up inlays would be immediately apparent by watching the runner's motion, in three dimensions in slow motion.

From the actual study of this thesis it is clear that photogrammetry can play a major role in the medical world. This has been verified by the successful development of the TAG computer system, that has proven to be an aid in analysing the respiratory motion of humans, and also by TAG's present and future developments in the study of gait analysis and of brain damaged subjects. With todays modern technology and with the PC computer it is no longer a tool

that the medical fraternity cannot afford to use, but has now become a viable possibility in todays world.

"Biological form and its motion to both external and internal forces has been and is one of the most engaging subjects in the history of human thought." (Sheffer and Herron 1989)

The power of three dimensional vision made possible by modern photogrammetry not only makes this subject engaging but utterly fascinating. It gives a whole new dimension to motion, especially to the motion of the human body.

LIST OF REFERENCES

Adams, L.P., B.A.Gutschow, A.Tregidga, M.Klein (1990) Near real time biostereometric studies of the regional body surface motion in respiration. Int Archiv Photogrammetry Remote Sensing; (not available as paper was only presented in September 1990)

Adams, L.P., M.Klein (1986) Biostereometric methods for the study of body surface motions during breathing.Int Archiv Photogrammetry Remote Sensing; 6(part5):263-270.

Adams,L.P., H.Ruther, M.Klein (1990) Evaluating regional body surface motion during breathing using stereophotogrammetry. ISPRS;45,152-160

Adams,L.P. (1981) X-ray stereo photogrammetry locating the precise, three dimenssional position of image points. Med Biol Eng Comput 19:569-578.

Adams,L.P. (1974) Stereoscopic viewing of image pairs with the naked eye. Photogramm Rec 8(44): 229-230

Denison, D.M, A.J Peacock, M.D.L. Morgan, M.A. Branthwaite and A.R. Gourlay (1982) Does the lung work?-3.Shedding light on the subject. Br J Dis Chest, 76:20-34

El-Hakim, S.F. (1986) A real-time system for object meaurement with CCD cameras. Int Archiv Photogrammetry Remote Sensing; Vol 26, Part 5:363-373.

Gourlay, A.R., G.Kaye, D.M.Denison, A.J.Peacock,M.D.L.Morgan (1984) Analysis of an optical mapping technique for lung function studies. Comput Biol Med Vol.14 No.1 pp47-58

Hierholzer, E., W.Frobin (1989) Raster Photogrammetry: Systems and Applications. Non-Topographic Photogrammetry (2nd edition), edited by H.M.Karara, American Society of Photogrammetry, Falls Church, Virginia 1989;16:265-278.

Klein, F. (1908) Elementary mathematics from an advanced standpoint. Geometry. Translated from the German 1939. Dover Publications Inc., USA

Klein, M. (1989) Biostereometric studies of regional body surface motion in respiration. Proceedings of the ninth conference of Southern African Surveyors, Cape Town, Paper 3.3: 5 pages

Konno, K., J.Mead (1967) Measurement of the separate volume changes of rib cage and abdomen during breathing. J Appl Physiol 22:407-422

Kováts, F. (1974) Morphometrical study of breathing movements using a stereometric method. Biostereometrics '74. American Society of Photogrammetry, Virginia 340-377.

Mikhail, E.M., J.C.Mcglone, F.C.Padres Jr. (1989) Introduction to metrology concepts. Non-Topographic Photogrammetry (2nd edition), edited by H.M.Karara, American Society of Photogrammetry, Falls Church, Virginia 1989;2:7-14.

Morgan, M.D.L, A.R.Gourlay, D.M.Denison (1984) An optical method of studying the shape and movement of the chest wall in recumbent patients. Thorax 39:101-106

Pekelsky, J.R., M.C.Van Wijk (1989) Moiré Topography: Systems and Applications. Non-Topographic Photogrammetry (2nd edition), edited by H.M.Karara, American Society of Photogrammetry, Falls Church, Virginia 1989;15:231-263.

Saumarez, R.C. (1986) Automated optical measurements of human torso surface movements during breathing. J Appl Physiol 60:702-709.

Scott, P.J. (1981) The reflex plotters: measurement without photographs. Photogramm Record, 10:435-446

Swokowski, E.W. (1983) Calculus with analytical geometry, Third Edition. PWS Publishers, Boston, Massachusetts, USA, 436-442

Tektronix (1988) User manual 4693D Color Printer. Tektronix, Inc. Wilsonville Industrial park, P.O.Box 1000, Wilsonville, Oregon 97070, USA

Thompson, E.H. (1969) An introduction to the Algebra of Matrices with some Applications. Adam Hilger, London, UK

Thompson, E.H. (1971) Space Resection without Interior orientation. The Photogrammetric Record, 23 (2),67-75

Wade,O.L. (1954) Movements of the thoracic cage and diaphragm in respiration J Physiol. 124,193-212

APPENDIX 1 THE PROJECTIVE TRANSFORMATION

The theory of the projective transformation has been developed over the years, starting with references to Klein (1908) and Thompson (1969 and 1972).

It was Thompson, a great photogrammetrist, who discovered the potential in the photogrammetric contest. He showed how the projective transformation formula could be used in single photograph geometry. He however did not forsee the extension of the theory past a single photograph.

By extending the theory to two or three cameras, the full potential of the projective transformation was realised.

The method developed by Adams (1981), extracted from his paper, is described.

PROJECTIVE TRANSFORMATIONS:

Since the picture is a special case of central projection it is possible to make use of central projection geometry and in particular, the somewhat involved mathmatics of projective transformations when we measure image co-ordinates in the plane of the picture.

Analytical definition of the projective transformation
We state tha (according to Klein, 1908)

$$X' = \frac{a_{11}X + a_{12}Y + a_{13}Z + a_{14}}{a_{41}X + a_{42}Y + a_{43}Z + a_{44}}$$

$$Y' = \frac{a_{21}X + a_{22}Y + a_{23}Z + a_{24}}{a_{41}X + a_{42}Y + a_{43}Z + a_{44}} \qquad (1)$$

$$Z' = \frac{a_{31}X + a_{32}Y + a_{33}Z + a_{34}}{a_{41}X + a_{42}Y + a_{43}Z + a_{44}}$$

That is to every point X,Y,Z there corresponds a definite finite point X',Y',Z', provided only that the common denominator is *not* zero.
In the study of projective transformations it is very convenient to use homogeneous co-ordinates; that is in place of the three point co-ordinates we use four quantities x,y,z,w defined by the eqns.

$$X = \frac{x}{w} \qquad Y = \frac{y}{w} \qquad Z = \frac{z}{w}$$

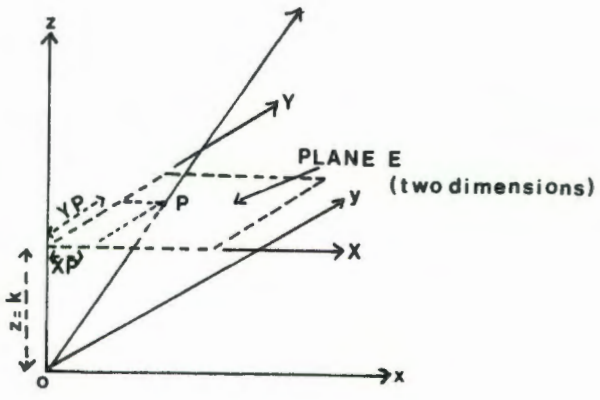

Fig 2. Diagram illustrating homogeneous co-ordinates for two-dimensional space

HOMOGENEOUS CO-ORDINATES - EXPLANATION

Consider first two-dimensional space - a plane E say. An equivalent would be a picture plane in which each image point has rectangular co-ordinates X,Y.
In plane E, P has co-ordinates X,Y in two dimensions. We now interpret x,y,z as rectangular co-ordinates in space and in this space we choose the plane z = k parallel to the x,y plane as the plane E. If we now join the point X,Y of E to O by a straight line then for points on this line x/z and y/z are constant. If we make z = k = 1 we may write

$$\frac{x}{z} = X \qquad \frac{y}{z} = Y$$

Accordingly the introduction of homogeneous co-ordinates signifies the representation of the plane E into that space pencil of rays with the origin O as centre of which E is a section.
That is, the homogeneous co-ordinates of a point are the space co-ordinates of the points of the projecting ray of that point.
The example gien represents the introduction of homogeneous co-ordinates into two-dimensional space.
We can form similar representation if we introduce homogeneous co-ordinates into three-dimensional space, that is we think of the space as a section of w = 1 say of a four-dimensional auxiliary space and we relate it to the space pencil which projects it from the origin of auxiliary space. In this the use of four-dimensional spce is only a convenient means of expression.
In terms of homogeneous co-ordinates eqn.1 can be expressed as:

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = A \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \qquad (2)$$

where matrix A is non-singular.
When the accented and unaccented co-ordinates are referred to the same system, eqns.1 and 2 represent a linear mapping of space upon itself which is known as collineation - non-singular since A is non-singular.
In the photographic case a difference occurs in that photography introduces a singular transformation for, to distinct planes in space there corresponds one, and only one, plane, the picture plane (Fig.3).
Such a transformation cannot be uniquely inverted for to every picture point there corresponds an infinity of space points.

HOMOGENEOUS CO-ORDINATES FOR THE X-RAY PICTURE CASE

We showed that two-dimensional co-ordinates X,Y could be expressed as homogeneous co-ordinates referred to an origing O in which the point X,Y fell on a space ray projection f from O such that, for points on the ray, $X = \frac{x}{z}$ and $Y = \frac{y}{z}$ are constant.

The position of the plane containing the co-ordinated point X,Y from the origin O is therefore

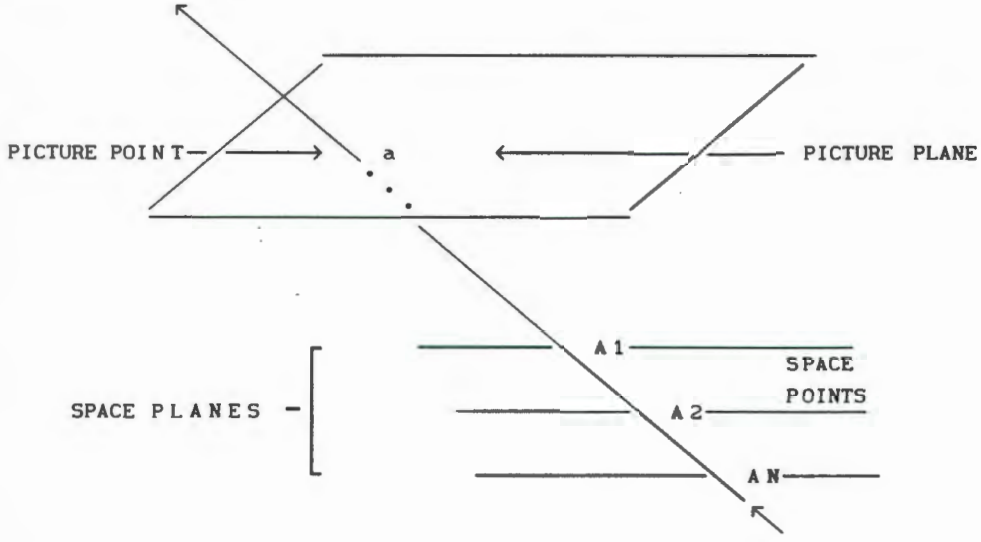

Fig. 3 *Illustration of singular transformation*

arbitrary but, in terms of the camera O is the perspective centre (assumed point focus) and Z = k is the perpendicular distance form the perspective centre to the plane of the film. In photogrammetric terms this is referred to as the principal distance.
The collinearity eqn.1 is in terms of X′,Y′,Z′ but, for two-dimensional co-ordinates x,y say and using homogeneous co-ordinates X′,Y′,Z′, we have $x = \frac{X'}{Z'}$, , $y = \frac{Y'}{Z'}$, where the xy plane is parallel to the X′Y′ plane and Z′ is arbitrary and constant.

Substituting in eqn. 1

$$x = \frac{X'}{Z'} = \frac{a_{11}X + a_{12}Y + a_{13}Z + a_{14}}{a_{41}X + a_{42}Y + a_{43}Z + a_{44}}$$

$$x \frac{a_{41}X + a_{42}Y + a_{43}Z + a_{44}}{a_{31}X + a_{32}Y + a_{33}Z + a_{34}}$$

therefore

$$x = \frac{a_{11}X + a_{12}Y + a_{13}Z + a_{14}}{a_{31}X + a_{32}Y + a_{33}Z + a_{34}}$$

Similarly:

$$y = \frac{a_{21}X + a_{22}Y + a_{23}Z + a_{24}}{a_{31}X + a_{32}Y + a_{33}Z + a_{34}}$$

which is a convenient way of expressing x and y (the plate co-ordinates) in therms of X,Y,Z (the space co-ordinates) when we consider the picture case.

We can eliminate one of the transformation parameters $\left(a_{34}\right)$ by dividing all terms on the right hand side of the equations by a_{34} and introducing new transformation parameters b_{ij} where $b_{ij} = \frac{a_{ij}}{a_{34}}$ so that we may now write:

$$x = \frac{b_{11}X + b_{12}Y + b_{13}Z + b_{14}}{b_{31}X + b_{32}Y + b_{33}Z + 1}$$

$$y = \frac{b_{21}X + b_{22}Y + b_{23}Z + b_{24}}{b_{31}X + b_{32}Y + b_{33}Z + 1} \qquad (3)$$

DERIVATION OF SPACE CO-ORDINATES FROM PLATE CO-ORDINATES

Rewriting eqn. 3 and gathering terms

$$b_{11}X + b_{12}Y + b_{13}Z + b_{14}$$

$$-b_{31}xX - b_{32}xY - b_{33}xZ = x \qquad (4)$$

Similarly from eqn. 3:

$$b_{21}X + b_{22}Y + b_{23}Z + b_{24}$$

$$-b_{31}yX - b_{32}yY - b_{33}yZ = y \qquad (5)$$

Now, provided that we have sufficient control points co-ordinated in terms of the space X,Y,Z system and suitably distributed and we measure comparator (x,y) co-ordinates of their image points in the plane of the picture we can set up solution eqns. in the form of 4 and 5 and hence solve for the transformation parameters b_{ij}.

Two pictures from different view points give us sufficient information to solve for two different sets of b_{ij} terms, i.e. b_{ij} and $\bar{b}_{ij}$ – and hence from measurements of x,y and $\bar{x},\bar{y}$ we can solve for X,Y,Z of a new point by back substitution.

From eqns. 4 and 5 we get:

$$(b_{11} - x.b_{31})X + (b_{12} - x.b_{32})Y + (b_{13} - x.b_{33})Z + b_{14} = x$$

$$(b_{21} - y.b_{31})X + (b_{22} - y.b_{32})Y + (b_{23} - y.b_{33})Z + b_{24} = y \qquad (6)$$

$$(\bar{b}_{11} - \bar{x}.\bar{b}_{31})X + (\bar{b}_{12} - \bar{x}.\bar{b}_{32})Y + (\bar{b}_{13} - \bar{x}.\bar{b}_{33})Z + \bar{b}_{14} = \bar{x}$$

$$(\bar{b}_{21} - \bar{y}.\bar{b}_{31})X + (\bar{b}_{22} - \bar{y}.\bar{b}_{32})Y + (\bar{b}_{23} - \bar{y}.\bar{b}_{33})Z + \bar{b}_{24} = \bar{y}$$

where the unbarred elements refer to the left hand picture and the barred elements to the right hand picture.

SOLUTION OF EQUATIONS

Once the b_{ij} terms have been solved the remaining calculations are straightforward. Eqns. 4 and 5 however are non-linear functions involving the b_{ij} parameters and the methods to be adopted to solve the equations are debatable. Initial values for the parameters can be determined if there are at least six control points imaged in a photograph but with the following important restrictions.
If using a minimum of six control points - giving one redundancy:

(i) no more than four space points should be coplanar;

(ii) no more than three space points should be colinear.

Various 'least squares' iterative solutions to the problem have been suggested in which space control co-ordinates are assumed error free and the comparator co-ordinates assumed subject to errors of observations - a somewhat debatable point since for very short-range photogrammetry the comparator observations are probably determined to a higher accuracy than the space determinations. It seems sensible therefore to assume that both objects and image co-ordinates are error free and to solve for mean 'b_{ij}' values using normal equations formed from 'quasi'-oberservation equations as follows:

b_{11}	b_{12}	b_{13}	b_{14}	b_{21}	b_{22}	b_{23}	b_{24}	b_{31}	b_{32}	b_{33}	= l
X_1	Y_1	Z_1	1	0	0	0	0	$-x_1X_1$	$-x_1Y_1$	$-x_1Z_1$	x_1
X_2	Y_2	Z_2	1	0	0	0	0	$-x_2X_2$	$-x_2Y_2$	$-x_2Z_2$	x_2
X_n	Y_n	Z_n	1	0	0	0	0	$-x_nX_n$	$-x_nY_n$	$-x_nZ^n$	x_n
0	0	0	0	X_1	Y_1	Z_1	1	$-y_1X_1$	$-y_1Y_1$	$-y_1Z_1$	y_1
0	0	0	0	X_2	Y_2	Z_2	1	$-y_2X_2$	$-y_2Y_2$	$-y_2Z_2$	y_2
0	0	0	0	X_n	Y_2	Z_2	1	$-y_nX_n$	$-y_nY_n$	$-y_nZ_n$	y_n
'A' matrix											'L' matr

From these 'quasi' observation equaions the 'b' matrix can be solved from

$$B = (A^TA)^{-1}A^TL$$

Substitution of the calculated b_{ij} and $\overline{b_{ij}}$ values and observed comparator values for image points in the left and right pictures into eqn. 6 leads to four solution equations for three unknowns (X,Y,Z) in the form:

X	Y	Z	= l
C_{11}	C_{12}	C_{13}	L_1
C_{21}	C_{22}	C_{23}	L_2
C_{31}	C_{32}	C_{33}	L_3
C_{41}	C_{42}	C_{43}	L_4
C matrix			L matrix

where, for example,

$$C_{11} = (b_{11} - xb_{31}),\ L_1 = (x - b_{14}) \text{ etc.}$$

and X,Y,Z are the space co-ordinates of the image point which is to be co-ordinated.
In matrix notoation the solution is given by:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = (C^TC)^{-1}C^TL$$

APPENDIX 2 THREE DIMENSIONAL CO-ORDINATES FOR THE BABY CONTROL FRAME

Three dimensional co-ordinates for the baby control frame

	X	Y	Z (mm)
1.	10.00	10.00	10.00
2.	3.99	209.45	5.60
3.	3.36	396.51	6.59
4.	107.86	400.91	44.29
5.	107.53	274.19	43.42
6.	113.96	9.39	47.88
7.	216.93	13.32	83.47
8.	216.48	210.65	82.99
9.	210.06	400.71	80.19
10.	8.65	80.47	172.33
11.	6.49	206.39	168.64
12.	9.06	347.41	168.69
13.	55.96	349.81	184.59
14.	55.87	166.18	187.54
15.	53.08	78.84	188.55
16.	106.32	81.43	205.99
17.	108.76	207.04	204.72
18.	103.36	348.71	202.49

APPENDIX 3 THREE DIMENSIONAL CO-ORDINATES OF THE CHILD/ADULT CONTROL FRAME

Three dimensional co-ordinates for the child/adult control

	X	Y	Z (mm)
1.	10.00	10.00	13.10
2.	10.50	177.10	13.60
3.	12.20	384.60	10.80
4.	18.35	549.60	11.80
5.	193.75	554.80	10.80
6.	191.50	371.20	11.30
7.	184.65	183.30	10.10
8.	190.70	8.85	10.90
9.	347.60	9.90	10.00
10.	354.20	174.15	10.50
11.	358.65	382.20	13.00
12.	356.50	548.65	14.70
13.	42.30	83.90	152.70
14.	43.20	202.70	153.20
15.	40.25	347.90	153.20
16.	44.70	469.25	153.30
17.	188.20	468.20	151.40
18.	186.60	345.20	149.70
19.	183.80	198.00	150.00

20.	180.00	73.10	148.00
21.	307.10	74.20	145.10
22.	309.3	194.40	146.50
23.	312.40	341.20	147.20
24.	312.05	462.60	147.20

APPENDIX 4 Main - MAIN PROGRAM OF TAG

(computer system)

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>

main()
{

 void gotto(int,int);
 void mode();
 void graph();
 void menu();
 void tod();
 void dmenu();
 void pmenu();
 void gmenu();
 void imp();

 int x,y;
 char kill;
 char ch='a';

 kill='t';

 mode();
 graph();
 mode();
 menu();

 while( kill != 'q') {

   switch(ch) {
     case'1':
        mode();
        tod();              /* ttod - images to disc */
        mode();
        menu();
        break;
     case'2':
        mode();
        dmenu();            /* calls up digitising menu */
        mode();
        menu();
        break;
     case'3':
        mode();
        pmenu();            /* calls up calculating menu - b params & 3D */
        mode();
        menu();
        break;
    case'4':
        mode();
        gmenu();           /* calls graphics menu */
        mode();
        menu();
        break;
     case'5':
```

```
        mode();
        imp();              /* printing images on Tektronix */
        mode();
        menu();
        break;
     case'0':
        kill = 'q';         /* to quit program */
        mode();
        continue;
        default:
        ;
     }
  gotto(22,38);
  ch=getch();    /*  getchar();              reads keyboard selection */
  }
}
/******* END OF MAIN ****************************/


/******* MENU ***********************************/

void menu()
{
  printf("╔═══════════════════════════════════════════════════════╗\n");
  printf("║                                                       ║\n");
  printf("║     MAIN MENU                                         ║\n");
  printf("║                                                       ║\n");
  printf("╠═══════════════════════════════════════════════════════╣\n");
  printf("║                                                       ║\n");
  printf("║                                                       ║\n");
  printf("║  1.  Image transfer to IBM from video                 ║\n");
  printf("║                                                       ║\n");
  printf("║  2.  MENU : Digitising                                ║\n");
  printf("║                                                       ║\n");
  printf("║  3.  MENU : Calculations                              ║\n");
  printf("║                                                       ║\n");
  printf("║  4.  MENU : Graphics                                  ║\n");
  printf("║                                                       ║\n");
  printf("║  5.  Image printing to Tektronix                      ║\n");
  printf("║                                                       ║\n");
  printf("║  0.  Quit                                             ║\n");
  printf("║                                                       ║\n");
  printf("║                                                       ║\n");
  printf("╠═══════════════════════════════════════════════════════╣\n");
  printf("║                                                       ║\n");
  printf("║     Enter number of your choice :                     ║\n");
  printf("║                                                       ║\n");
  printf("╚═══════════════════════════════════════════════════════╝");
}
```

APPENDIX 5 DMEN.C - DIGITISING MENU OF TAG

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>

void daenu()
{
 void mode();
 void menu2();
 void conl();
 void conr();
 void objl();
 void objr();
 void single();
 void outline();
 void gotto(int,int);

 char kill;
 char ch='a';

 kill = 't';

 mode();
 menu2();

 while( kill != 'q') {

   switch(ch) {
    case'1':
       mode();
       conl();          /* tconl - digitising LEFT stereo control */
       mode();
       menu2();
       break;
    case'2':
       mode();
       conr();          /* tconr - digitising RIGHT stereo control */
       mode();
       menu2();
       break;
    case'3':
       mode();
       objl();          /* tobjl - digitising stereo objects - left */
       mode();
       menu2();
       break;
    case'4':
       mode();
       objr();          /* tobjr - digitising stereo objects - right */
       mode();
       menu2();
       break;
    case'5':
       mode();
       single();          /* tsingle - digitising single image */
       mode();
```

```
        menu2();
        break;
      case'6':
        mode();
        outline();          /* toutline - digitising babies outlines */
        mode();
        menu2();
        break;
      case'0':
        kill = 'q';         /* to quit program */
        mode();
        continue;
        default:
        ;
      }
  gotto(22,36);
  ch=getch();    /*  getchar();              reads keyboard selection */
  }
}
/******* END OF MAIN ******************************/


/******* MENU *************************************/

void menu2()
{
  printf("╔═══════════════════════════════════════════════════╗\n");
  printf("║                                                   ║\n");
  printf("║     DIGITISING :                                  ║\n");
  printf("║                                                   ║\n");
  printf("╠═══════════════════════════════════════════════════╣\n");
  printf("║                                                   ║\n");
  printf("║  1.  Stereo control image - LEFT                  ║\n");
  printf("║                                                   ║\n");
  printf("║  2.  Stereo control image - RIGHT                 ║\n");
  printf("║                                                   ║\n");
  printf("║  3.  Stereo object (patient) image - LEFT         ║\n");
  printf("║                                                   ║\n");
  printf("║  4.  Stereo object (patient) image - RIGHT        ║\n");
  printf("║                                                   ║\n");
  printf("║  5.  Single image                                 ║\n");
  printf("║                                                   ║\n");
  printf("║  6.  Patient's outline                            ║\n");
  printf("║                                                   ║\n");
  printf("║  0.  To return to MAIN MENU                       ║\n");
  printf("║                                                   ║\n");
  printf("╠═══════════════════════════════════════════════════╣\n");
  printf("║                                                   ║\n");
  printf("║     Enter number of your choice :                 ║\n");
  printf("║                                                   ║\n");
  printf("╚═══════════════════════════════════════════════════╝");
}
```

APPENDIX 6 PMEN.C - CALCULATING MENU OF TAG

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>

void menu()
{

 void mode();
 void menu1();
 void bparams();
 void d3();
 void gotto(int,int);

 char kill;
 char ch;

 kill = 't';

 mode();
 menu1();

 while( kill != 'q') {

   switch(ch) {
     case'1':
        printf("  \n");
        printf("  \n");
        mode();
        bparams();              /* calculating b params */
        mode();
        menu1();
        break;
     case'2':
        printf("  \n");
        printf("  \n");
        mode();
        d3();         /* claculating 3D co-ords of object points */
        mode();
        menu1();
        break;
     case'0':
        kill = 'q';        /* to quit program */
        mode();
        continue;
        default:
        ;
     }
  gotto(21,37);
  ch=bdos(7,0,0);    /*  getchar();              reads keyboard selection */
  }
}
/******* END OF MAIN ******************************/


/******* MENU *************************************/
```

```
void menu1()
{
  printf("  \n");
  printf("  \n");
  printf("  \n");
  printf("                                                                  \n");
  printf("                                                                  \n");
  printf("      CALCULATING :                                               \n");
  printf("                                                                  \n");
  printf("                                                                  \n");
  printf("                                                                  \n");
  printf("                                                                  \n");
  printf("                                                                  \n");
  printf("  1.  B parameters for a camera set up                            \n");
  printf("                                                                  \n");
  printf("  2.  3D co-ordinates of object points                            \n");
  printf("                                                                  \n");
  printf("  0.  To return to MAIN MENU                                      \n");
  printf("                                                                  \n");
  printf("                                                                  \n");
  printf("                                                                  \n");
  printf("                                                                  \n");
  printf("                                                                  \n");
  printf("      Enter number of your choice :                               \n");
  printf("                                                                  \n");
  printf("                                                                  \n");

}
```

APPENDIX 7 GMEN.C - GRAPHICS MENU OF TAG

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>

void gmenu()
{
 void mode();
 void menu3();
 void gotto(int,int);
 void vec();

 char kill;
 char ch='a';

 kill = 't';

 mode();
 menu3();

 while( kill != 'q') {

   switch(ch) {
     case'1':
        printf("  \n");
        printf("  \n");
        mode();
        system("tvec");              /* calculating b params */
        mode();
        menu3();
        break;
     case'2':
        printf("  \n");
        printf("  \n");
        mode();
        system("tplt");              /* calculating b params */
        mode();
        menu3();
        break;
     case'0':
        kill = 'q';         /* to quit program */
        mode();
        continue;
        default:
        ;
     }
  gotto(21,37);
  ch=bdos(7,0,0);    /*  getchar();              reads keyboard selection */
  }
}
/******* END OF MAIN ******************************/


/******* MENU *************************************/

void menu3()
{
```

```
printf("  \n");
printf("  \n");
printf("  \n");
printf("                                                                  \n");
printf("  ╔═══════════════════════════════════════════════════════════╗   \n");
printf("  ║                                                           ║   \n");
printf("  ║     GRAPHICS :                                            ║   \n");
printf("  ║                                                           ║   \n");
printf("  ╠═══════════════════════════════════════════════════════════╣   \n");
printf("  ║                                                           ║   \n");
printf("  ║                                                           ║   \n");
printf("  ║                                                           ║   \n");
printf("  ║  1.  Stereo vectors on stereo video frame                 ║   \n");
printf("  ║                                                           ║   \n");
printf("  ║  2.  Displacement & 3D vectors - soft & hardcopy          ║   \n");
printf("  ║                                                           ║   \n");
printf("  ║  0.  To return to MAIN MENU                               ║   \n");
printf("  ║                                                           ║   \n");
printf("  ║                                                           ║   \n");
printf("  ║                                                           ║   \n");
printf("  ╠═══════════════════════════════════════════════════════════╣   \n");
printf("  ║                                                           ║   \n");
printf("  ║     Enter number of your choice :                         ║   \n");
printf("  ║                                                           ║   \n");
printf("  ╚═══════════════════════════════════════════════════════════╝   \n");

}
```

APPENDIX 8 TOD.C - TAG PROGRAM TO TRANSFER VIDEO IMAGES TO DISC

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>

void tod()
{

void mode();
void gotto(int,int);

char buf[260];
char dff[1100];
char str[15];
char nam[20];
char namm[20];
char name[20];
char s[30];
char str1[20];
char str2[20];
char ch;
char cht;
int no;
int wait = 1;
int index = 255;



printf("  \n");
printf(" CAPTURING IMAGES FROM VIDEO TAPE  \n");
printf(" ================================  \n");
printf("  \n");
printf("  \n");
printf(" Ensure that the video lead marked VIDEO OUT is connected \n");
printf(" to the green cable, which in turn is plugged into chan 0  \n");
printf(" of the PIP board 0 in the back of the computer. \n");
printf("  \n");
printf(" Board 0 is the lower PIP board and chan 0 is on the right \n");
printf(" when looking at the back of the computer. \n");
printf("  \n");
printf(" Board 0 is connected to the left hand Phillips monitor \n");
printf(" when looking at the front of the computer. \n");
printf("  \n");
printf(" Press SPACE BAR to continue : ");
ch = bdos(7,0,0);
while( ch != 32 ) {
    ch = bdos(7,0,0);
    }
printf("  \n");
printf("  \n");
printf("  PIP BOARD BEING ACTIVATED \n");


/********** INITIALISING THE PIP BOARD ********************************/

fg_inifat(620,1,0,0,1,0);    /* initialises board at base address */
```

```
fg_setind(index);              /* set drawing index */
fg_clear(0,7);                /* clear frame buffer to current index */
fg_sboard(0);
fg_sbuf(0);
fg_chan(0);
fg_sync(1);

/*************************************************************************/

mode();

do {
  printf("  \n");
  printf(" CAPTURING IMAGES FROM VIDEO TAPE  \n");
  printf(" ================================= \n");
  printf("  \n");

  printf(" PATIENT'S NAME ( 4 or less keystrokes ) : ");
  scanf("%s",name);                          /*    name = deb          */
  printf("  \n");
  mode();

  strcpy(nam,name);                          /*    nam = deb           */
  strcpy(str,"c");                           /*    str = c             */
  strcat (nam,str);                          /*    nam = debc          */
  strcpy(namm,name);                         /*    namm = deb          */
  strcpy(str,"o");                           /*    str = c             */
  strcat (namm,str);                         /*    namm = debo         */


  /******* CONTROL IMAGE TO DISC*******************************/

  printf(" STEREO CONTROL IMAGE TO DISC  \n");
  printf(" ============================= \n");
  printf("   \n");
  printf("   \n");
  printf(" Do you wish to store a control image ? y/n :");
  ch = bdos(7,0,0);
  while( ch !='y' && ch !='n'&& ch !='Y' && ch !='N') {
      ch = bdos(7,0,0);
      }
  mode();

  if(ch == 89 || ch == 121) {
    printf(" STEREO CONTROL IMAGE TO DISC  \n");
    printf(" ============================= \n");
    printf("   \n");
    printf(" Press the OPERATE button to turn the video machine on. \n");
    printf(" Insert the video tape and set the channel to AV.  \n");
    printf("  \n");
    printf(" Now press the PLAY button for normal playback. \n");
    printf(" To reduce the playback speed press SLOW - for the  \n");
    printf(" slowest slow motion speed. \n");
    printf("  \n");
    printf(" When correct image frozen on screen ( by pressing PAUSE button ) \n");
    printf(" Press SPACE BAR to start image transfer : ");
    ch = bdos(7,0,0);
    while( ch != 32) {
       ch = bdos(7,0,0);
```

```
    }
  printf("  \n");
  printf("  \n");
  printf(" IMAGE BEING TRANSFERRED  \n");

  fg_snap(wait);
  strcpy(s,"\\pip\\i\\");                    /*    s =  \pip\i\          */
  strcat (s,nam);                            /*    s = \pip\i\debc       */
  fg_todisk(1024,0,s,dff,-1);     /* call up image */
  }

mode();


/******* OBJECT IMAGES TO DISC ********************************/

strcpy(s,"\\pip\\i\\");                    /*    s =  \pip\i\          */
strcat(s,name);                            /*    s = \pip\i\debo       */
strcpy(name,s);                            /*    name = \pip\i\debo    */


printf(" STEREO OBJECT IMAGES TO DISC  \n");
printf(" ============================ \n");
printf("   \n");
printf("   \n");
printf(" Do you wish to store object images ?  y/n : ");
ch = bdos(7,0,0);
while( ch !='y' && ch !='n'&& ch !='Y' && ch !='N') {
    ch = bdos(7,0,0);
    }
mode();

while(ch == 89 || ch == 121) {
    printf(" STEREO OBJECT IMAGES TO DISC  \n");
    printf(" ============================ \n");
    printf("   \n");
    printf(" Enter the breath number  and  in or out  i.e. 1i : ");
    scanf("%s",str);
    printf("   \n");
    printf(" Press the play button to continue video playback \n");
    printf("   \n");

    strcpy(s,name);                        /*    s = \pip\i\debo     */
    strcat(s,str);                         /*    s = \pip\i\debo1i   */
    printf(" Hit SLOW - for frame by frame movement \n");
    printf(" When correct image frozen on screen ( by pressing PAUSE button ) \n");
    printf(" Press SPACE BAR to start image transfer : ");
    ch = bdos(7,0,0);
    while(ch !=32) {
      ch = bdos(7,0,0);
      }
    printf("  \n");
    printf("  \n");
    printf(" IMAGE BEING TRANSFERRED  \n");
```

```
      fg_snap(wait);
      fg_todisk(1024,0,s,dff,-1);     /* call up image */

      mode();
      printf(" STEREO OBJECT IMAGES TO DISC  \n");
      printf(" ========================== \n");
      printf("   \n");
      printf(" Do you wish to store another image ?  y/n : ");
      ch = bdos(7,0,0);
      while( ch !='y' && ch !='n'&& ch !='Y' && ch !='N') {
         ch = bdos(7,0,0);
         }
      mode();
      }

   printf(" CAPTURING IMAGES FROM VIDEO TAPE  \n");
   printf(" ================================= \n");
   printf("  \n");
   printf(" Do you wish to do another patient ?  y/n : ");
   cht = bdos(7,0,0);
   while( cht !='y' && cht !='n'&& cht !='Y' && cht !='N') {
       cht = bdos(7,0,0);
       }

}while(cht == 89 || cht == 121);

mode();

printf(" STOPPING THE VIDEO SESSION  \n");
printf(" ========================== \n");
printf("  \n");
printf(" Stop the video playback by pressing STOP \n");
printf("  \n");
printf(" Press OPERATE to turn the video machine off \n");
printf("  \n");
printf(" Press SPACE BAR to continue : ");
ch = bdos(7,0,0);
while( ch != 32 ) {
   ch = bdos(7,0,0);
   }
mode();


fg_exit();
}
/******* END OF MAIN ******************************/
```

APPENDIX 9 CONL.C - TAG PROGRAM TO DIGITSE LEFT CONTROL IMAGE

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>


char buf[260];
char dff[1100];
int numi[30];
int count;

conl()
{

 char str[15];
 char nam[20];
 char name[20];
 char name[20];
 char s[30];
 char str1[20];
 char str2[20];
 char cont,ch;
 int num[30];
 int no;
 int sumcon = 24;
 int count = 24;
 int noa = 1;
 int trash = 1;
 int index = 255;
 int sum = 30;

 printf("  \n");
 printf(" STEREO CONTROL IMAGE DIGITISING - LEFT \n");
 printf(" ================================== \n");
 printf("  \n");
 printf(" PATIENT'S NAME : ");
 scanf("%s",name);                        /*    name = deb            */
 printf("  \n");

 /********** INITIALISING THE PIP BOARD ********************************/

 fg_inifat(620,1,0,0,1,0);    /* initialises board at base address */
 fg_setind(60);              /* set drawing index */
 fg_clear(0,7);               /* clear frame buffer to current index */
 fg_sboard(0);
 fg_sbuf(1);
 fg_chan(1);
 fg_sync(0);

 /*************************************************************************/


 strcpy(nam,name);                        /*    nam = deb             */
 strcpy(name,name);                       /*    name = deb            */
 printf("  \n");
 strcpy(s,"\\pip\\i\\");                  /*    s =  \pip\i\          */
```

```
strcpy(str,"c");                        /*    str = c                 */
strcat (name,str);                      /*    name = debc             */
strcat (s,name);
fg_frdisk(1024,0,s,dff,-1);     /* call up image */
fg_rectf(275,0,512,512);          /* rectfills the right image */
fg_setind(index);               /* set drawing index */
fg_moveto(290,160);
fg_text("Digitise",3);
fg_moveto(290,230);
fg_text(" Left",3);
fg_moveto(290,300);
fg_text(" Image",3);
printf(" Digitising Large or Small control l/s: ");
cont=getch();                           /*    cont = l or s           */
while(cont != 'l' && cont != 's' && cont != 'L' && cont != 'S') {
    cont=getch();
    }
printf(" \n\n");
printf(" Turn on the right hand monitor. \n");
printf("  \n");
printf(" Spress SPACE BAR to continue : ");
ch=getch();
while( ch!=32) {
    ch=getch();
    }
fg_sboard(1);
if( cont==115 || cont==83) {
   strcpy(s,"babcon1");
   }
if( cont==108 || cont==76) {
   strcpy(s,"babcon2");
   }
fg_frdisk(1024,0,s,dff,-1);     /* call up image */
fg_setind(60);                /* set drawing index */
fg_rectf(265,0,512,512);          /* rectfills the right image */
fg_setind(index);               /* set drawing index */
fg_moveto(290,160);
fg_text("Digitise",3);
fg_moveto(290,230);
fg_text(" Left",3);
fg_moveto(290,300);
fg_text(" Image",3);
fg_sboard(0);
fg_setind(index);               /* set drawing index */

/******* LEFT IMAGE CONTROL ********************************/

mode();
printf("  \n");
printf(" ONLY DIGITISING LEFT IMAGE OF STEREO PAIR \n");
printf(" ========================================== \n");
printf("  \n");
printf(" Enter point nos. not visible (in ascending order), then enter 0 ! \n");

for(no=1; no <= sumcon; no++)  {
    num[no] = no;
    }
do {
   count = count - 1;
```

```
    printf(" No = ");
    scanf("%d",&trash);
    num[trash] = 0;
    } while(trash != 0);
 count = count + 1;
 for(no=1; no <= count; no++)  {
    if(num[noa] != 0) {
         num1[no] = noa;
         }
    else {
         no = no - 1;
         }
    noa = noa + 1;
    }
 count = count + 1;

 for(no=count; no <= sum; no++)  {
   num1[no] = 0;
   }

 strcpy(s,"c1.txt");
 strcat (naa,s);
 strcpy(str1," LEFT IMAGE ");
 strcpy(str2," ========== ");
 mode();
 cursor(naa,buf,dff,num1,count,str1,str2);
 mode();
 fg_sboard(1);
 fg_setind(60);              /* set drawing index */
 fg_rectf(0,0,512,512);           /* rectfills the right image */
 fg_sboard(0);

 fg_exit();
}
/******* END OF MAIN ******************************/
```

APPENDIX 10 CONR.C - TAG PROGRAM TO DIGITISE RIGHT CONTROL IMAGE

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>


char buf[260];
char dff[1100];
int num1[30];
int count;

conr()
{

 char str[15];
 char naa[20];
 char name[20];
 char nam[20];
 char s[30];
 char str1[20];
 char str2[20];
 char cont,ch;
 int num[30];
 int no;
 int suncon = 24;
 int count = 24;
 int noa = 1;
 int trash = 1;
 int index = 255;
 int sum = 30;

 printf("  \n");
 printf(" STEREO CONTROL IMAGE DIGITISING - RIGHT \n");
 printf(" ================================== \n");
 printf("  \n");
 printf(" PATIENT'S NAME : ");
 scanf("%s",name);                          /*    name = deb            */
 printf("  \n");

 /********** INITIALISING THE PIP BOARD ********************************/

 fg_inifmt(620,1,0,0,1,0);    /* initialises board at base address */
 fg_setind(60);              /* set drawing index */
 fg_clear(0,7);               /* clear frame buffer to current index */
 fg_sboard(0);
 fg_sbuf(1);
 fg_chan(1);
 fg_sync(0);

 /*************************************************************************/


 strcpy(naa,name);                          /*    naa = deb             */
 strcpy(nama,name);                         /*    nam = deb             */
 printf("  \n");
 strcpy(s,"\\pip\\i\\");                    /*    s =  \pip\i\          */
```

```
strcpy(str,"c");                        /*    str = c                 */
strcat (name,str);                      /*    name = debc             */
strcat (s,name);
fg_frdisk(1024,0,s,dff,-1);     /* call up image */
fg_rectf(0,0,250,512);           /* rectfills the right image */
fg_setind(index);               /* set drawing index */
fg_moveto(30,160);
fg_text("Digitise",3);
fg_moveto(30,230);
fg_text(" Right",3);
fg_moveto(30,300);
fg_text(" Image",3);
printf(" Digitising Large or Small control l/s: ");
cont=getch();                           /*    cont = l or s           */
while(cont != 'l' && cont != 's' && cont != 'L' && cont != 'S') {
    cont=getch();
    }
printf(" \n\n");
printf(" Turn on the right hand monitor. \n");
printf("  \n");
printf(" Press SPACE BAR to continue : ");
ch=getch();
while(ch!=32) {
   ch=getch();
   }
fg_sboard(1);
if( cont==115 || cont==83) {
   strcpy(s,"babcon1");
   }
if( cont==108 || cont==76) {
   strcpy(s,"babcon2");
   }
fg_frdisk(1024,0,s,dff,-1);     /* call up image */
fg_setind(60);              /* set drawing index */
fg_rectf(0,0,250,512);           /* rectfills the right image */
fg_setind(index);               /* set drawing index */
fg_moveto(30,160);
fg_text("Digitise",3);
fg_moveto(30,230);
fg_text(" Right",3);
fg_moveto(30,300);
fg_text(" Image",3);
fg_sboard(0);
fg_setind(index);               /* set drawing index */


/******* RIGHT IMAGE CONTROL ********************************/

mode();
printf("  \n");
printf(" ONLY DIGITISING RIGHT IMAGE OF STERED PAIR \n");
printf(" ================================== \n");
printf("  \n");
printf(" Enter point nos. not visible (in ascending order), then enter 0 ! \n");

for(no=1; no <= sumcon; no++)  {
    num[no] = no;
    }
do {
```

```
    count = count - 1;
    printf(" No = ");
    scanf("%d",&trash);
    num[trash] = 0;
    } while(trash != 0);
  count = count + 1;
  for(no=1; no <= count; no++)  {
    if(num[noa] != 0) {
         num1[no] = noa;
         }
    else {
         no = no - 1;
         }
    noa = noa + 1;
    }
  count = count + 1;

  for(no=count; no <= sum; no++)  {
    num1[no] = 0;
    }

  strcpy(s,"cr.txt");
  strcat (naa,s);
  strcpy(str1," RIGHT IMAGE ");
  strcpy(str2," ========== ");
  mode();
  cursor(naa,buf,dff,num1,count,str1,str2);
  mode();
  fg_sboard(1);
  fg_setind(60);                    /* set drawing index */
  fg_rectf(0,0,512,512);            /* rectfills the right image */
  fg_setind(index);                 /* set drawing index */
  fg_sboard(0);

  fg_exit();
}
/******* END OF MAIN ******************************/
```

APPENDIX 11 OBJL.C - TAG PROGRAM TO DIGITISE LEFT OBJECT IMAGE

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>


char buf[260];
char dff[1100];
int num1[30];
int count;

void obj1()
{
void mode();
void cursor(char [20],char [260],char [1100],int [30],int,char [20],char [20]);

char str[15];
char nam[20];
char name[20];
char namne[20];
char s[30];
char str1[20];
char str2[20];
char ch,cht;
int num[30];
int no;
int count = 18;
int noe = 1;
int index = 255;
int sum = 30;

printf(" \n");
printf(" STEREO OBJECT IMAGE DIGITISING - LEFT \n");
printf(" ================================== \n");
printf(" \n");
printf (" PATIENT'S NAME : ");
scanf("%s",nam);                          /* nam = deb          */
strcpy(namne,nam);                        /* namne = deb        */
strcpy(s,"\\pip\\i\\");
strcpy(str,"o");
strcat(nam,str);                          /* nam = debo         */
printf(" \n");
printf(" Enter breath number  and  in or out  i.e.  1i : ");
scanf("%s",str);                          /*  str = 1i          */
strcat(nam,str);                          /*  nam = deboli      */
strcpy(name,nam);                         /*  name = deboli     */
printf(" \n");


/********** INITIALISING THE PIP BOARD ********************************/

fg_inifst(620,1,0,0,1,0);    /* initialises board at base address */
fg_setind(index);              /* set drawing index */
fg_clear(0,7);               /* clear frame buffer to current index */
fg_sboard(0);
fg_sbuf(1);
```

```
fg_chan(1);
fg_sync(0);

/*************************************************************************/

strcat(s,nam);
fg_frdisk(1024,0,s,dff,-1);     /* call up image */

/*************************************************************************/

mode();
printf(" \n");
printf(" STEREO OBJECT IMAGE DIGITISING - LEFT \n");
printf(" ================================== \n");
printf(" \n");
printf (" ONLY DIGITISE LEFT IMAGE ! \n");
printf (" IS THIS THE FIRST BREATHING FRAME OF THE PATIENT   y/n : ");
ch=getch();
while(ch != 'y' && ch !='n' && ch !='Y' && ch !='N') {
    ch=getch();
    }
printf(" \n");
mode();
if( ch==121 || ch==89) {      /* yes */
   printf(" \n");
   printf(" STEREO OBJECT IMAGE DIGITISING - LEFT \n");
   printf(" ================================== \n");
   printf(" \n");
   printf (" Only digitise the four fiducial marks ! \n");
   printf(" \n");
   printf (" Press SPACE BAR to continue : ");
   cht=getch();
   while( cht !=32 ) {
      cht=getch();
      }
   printf(" \n");
   strcpy(s,"4L.TXT");
   strcat(name,s);
   strcpy(str1," 4 FIDUCIAL ");
   strcpy(str2," ========= ");
   for(no=1; no <= 4; no++)  {
      num1[no] = no;
      }
   num1[5]=0;
   cursor(name,buf,dff,num1,count,str1,str2);
   mode();
   strcpy(s,"\\pip\\i\\");
   strcat(s,nam);
   fg_frdisk(1024,0,s,dff,-1);     /* call up image */
   }
mode();
printf(" \n");
printf(" STEREO OBJECT IMAGE DIGITISING - LEFT \n");
printf(" ================================== \n");
printf(" \n");
printf (" First digitise the four fiducial marks ! \n");
printf (" Then digitse the left object points ! \n");
printf(" \n");
```

```
printf (" Press SPACE BAR to continue : ");
cht=getch();
while( cht !=32 ) {
    cht=getch();
    }
printf(" \n");

 /******** MEASURING LEFT OBJECT POINTS ***************************/

for(no=1; no <= sum; no++)  {
    num1[no] = no;
    }

strcpy(s,"1.txt");
strcat (name,s);
strcpy(str1," LEFT IMAGE ");
strcpy(str2," ========== ");

cursor(name,buf,dff,num1,count,str1,str2);
mode();
fg_exit();
}
```

APPENDIX 12 OBJR.C - TAG PROGRAM TO DIGITISE RIGHT OBJECT IMAGE

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>


char buf[260];
char dff[1100];
int num1[30];
int count;

void objr()
{
void mode();
void cursor(char [20],char [260],char [1100],int [30],int,char [20],char [20]);

char str[15];
char naa[20];
char name[20];
char nanne[20];
char s[30];
char str1[20];
char str2[20];
char ch;
int num[30];
int no;
int count = 18;
int noa = 1;
int index = 255;
int sum = 30;

printf(" \n");
printf(" STEREO OBJECT IMAGE DIGITISING - RIGHT \n");
printf(" ========================================= \n");
printf(" \n");
printf (" PATIENT'S NAME : ");
scanf("%s",naa);                          /* naa = deb          */
strcpy(nanne,naa);                        /* nanne = deb        */
strcpy(s,"\\pip\\i\\");
strcpy(str,"o");
strcat(naa,str);                          /* naa = debo         */
printf(" \n");
printf(" Enter breath number  and   in or out   i.e.   1i : ");
scanf("%s",str);                          /* str = 1i           */
strcat(naa,str);                          /* naa = deboli       */
strcpy(name,naa);                        /* name = deboli      */


/********** INITIALISING THE PIP BOARD ********************************/

fg_inifmt(620,1,0,0,1,0);    /* initialises board at base address */
fg_setind(index);             /* set drawing index */
```

```
fg_clear(0,7);              /* clear frame buffer to current index */
fg_sboard(0);
fg_sbuf(1);
fg_chan(1);
fg_sync(0);

/*************************************************************************/

strcat(s,nam);
fg_frdisk(1024,0,s,dff,-1);     /* call up image */

/*************************************************************************/

mode();
printf(" \n");
printf(" STEREO OBJECT IMAGE DIGITISING - RIGHT \n");
printf(" ======================================= \n");
printf(" \n");
printf(" ONLY DIGITISE RIGHT IMAGE ! \n");
printf("\n IS THIS THE FIRST BREATHING FRAME OF THE PATIENT  y/n : ");
ch=bdos(7,0,0);
while( ch !='y' && ch !='n' && ch !='Y' && ch !='N') {
   ch=bdos(7,0,0);
   }
printf(" \n");
mode();
if(ch==121 || ch==89) {
   printf(" \n");
   printf(" STEREO OBJECT IMAGE DIGITISING - RIGHT \n");
   printf(" ======================================= \n");
   printf(" \n");
   printf (" Only digitise the four fiducial marks ! \n");
   printf(" \n");
   printf(" Press SPACE BAR to continue : ");
   ch=bdos(7,0,0);
   while( ch !=32) {
      ch=bdos(7,0,0);
      }
   printf("\n");
   strcpy(s,"4R.TXT");
   strcat(namme,s);
   strcpy(str1," 4 FIDUCIAL ");
   strcpy(str2," ========== ");
   for(no=1;no<=4;no++) {
    numi[no] = no;
    }
   numi[5]=0;
   cursor(namme,buf,dff,numi,count,str1,str2);
   mode();
   strcpy(s,"\\pip\\i\\");
   strcat(s,nam);
   fg_frdisk(1024,0,s,dff,-1);     /* call up image */
   }
mode();
printf(" \n");
printf(" STEREO OBJECT IMAGE DIGITISING - RIGHT \n");
printf(" ======================================= \n");
printf(" \n");
```

```
printf("  First digitise the four fiducial marks ! \n");
printf("  Then digitise right object points \n");
printf(" \n");
printf("\n Press SPACE BAR to continue : ");
ch=bdos(7,0,0);
while( ch !=32) {
    ch=bdos(7,0,0);
    }
printf(" \n");

/******** MEASURING RIGHT OBJECT POINTS ***************************/

for(no=1;no<=sum;no++) {
    numi[no] = no;
    }

strcpy(s,"\\pip\\i\\");
strcat (s,name);
strcpy(s,"r.txt");
strcat (name,s);
strcpy(str1," RIGHT IMAGE ");
strcpy(str2," ========== ");

cursor(name,buf,dff,numi,count,str1,str2);
mode();


/*************************************************************/


fg_exit();
}
/******** END OF MAIN ****************************/
```

APPENDIX 13 SINGLE.C - TAG PROGRAM TO DIGITISE SINGLE IMAGE

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>


char buf[260];
char dff[1100];
int num1[30];
int count;

void single()
{

void cursor(char [20],char [260],char [1100],int [30],int,char [20],char [20]);

char str[15];
char nam[20];
char name[20];
char name[20];
char namee[20];
char s[30];
char str1[20];
char str2[20];
int num[30];
int trash = 1;
int no;
int suncon = 18;
int count = 18;
int non = 1;
int index = 255;
int sum = 30;

printf ("  \n");
printf (" SINGLE IMAGE DIGITISING  \n");
printf (" ====================== \n");
printf ("  \n");
printf (" ENTER IMAGE NAME: ");
scanf("%s",str);
printf ("  \n");
printf (" CO-ORDINATE FILE NAME  and EXTENSION : ");
scanf("%s",name);
printf ("  \n");


/********** INITIALISING THE PIP BOARD ********************************/

fg_inifmt(620,1,0,0,1,0);    /* initialises board at base address */
fg_setind(index);              /* set drawing index */
fg_clear(0,7);               /* clear frame buffer to current index */
fg_sboard(0);
fg_sbuf(1);
fg_chan(1);
fg_sync(0);

/*************************************************************************/
```

```
strcpy(s,"\\pip\\i\\");

strcat (s,str);
fg_frdisk(1024,0,s,dff,-1);     /* call up image */


for(no=1; no <= sum; no++)  {
   numl[no] = no;
   }

strcpy(str1," SINGLE IMAGE ");
strcpy(str2," ============ ");

cursor(name,buf,dff,numl,count,str1,str2);

fg_exit();
}
/******* END OF MAIN ******************************/
```

APPENDIX 14 OUTLINE.C - TAG PROGRAM TO DIGITSE PATIENT'S OUTLINE

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>

char buf[260];
char dff[1100];

int pixc[20];
int pixr[20];
float outx[500];
float outy[500];
int x, y;
int a, b, c, d, e, f, i, p, q;
int no, num, numm, grey;
char chnum[5];
int ended;

void outline()
{

void mode();
void gotto(int,int);
void pera(int,int,int,int,int[20],int[20],float[60],float[60]);
void pixread(int,int,int[20],int[20]);
void pixwrit(int,int,int[20],int[20]);
void cross(int,int);


char buf[260];
char dff[1100];
FILE *fp;                    /* file description for outputting data */
char ch = 't';
char cht;
char chnum[5];
char s[20];
char str1[20];
char str2[20];
char nam[20];
char name[20];
char str[15];
char kill;
char outl='e';
int pixr[20];
int pixc[20];
float outx[500];
float outy[500];
float xx[50],yy[50],zz[50];
float Xc[500],Yc[500],A[4][1],X[4][4],B[4][1];
float qq;
int  n[50];
int num1[500];
int p,q;
int x = 0;
int y = 0;
int a = 0;
int b = 0;
```

```
int c = 0;
int d = 0;
int e = 0;
int f = 0;

int i = 0;
int no,mm,nn,j,k;
int num = 1;
int numm = 0;
int ended = 0;
int index = 255;
int grey;
int nu1,nu2;                          /* points on shoulder & hip for trans */

printf(" DIGITISNG BABIES OUTLINE \n");
printf(" ======================== \n");
printf(" \n");
printf(" PATIENT'S NAME : ");
scanf("%s",nam);                          /* nam = deb         */
strcpy(str,"outl.txt");
strcpy(name,nam);                         /* name = deb      */
strcat(name,str);                         /* name = deboutl.txt     */
strcpy(s,"\\pip\\i\\");
strcpy(str,"o");
strcat(nam,str);                          /* nam = debo        */
printf(" \n");
printf(" Enter breath number  and  in or out  i.e. 1i : ");
scanf("%s",str);                          /* str = 1i          */
strcat(nam,str);                          /* nam = deboli      */


/************** INITIALISING PIP BOARD ***************************************/

fg_inifat(620,1,0,0,1,0);        /* initialises pip board at base address */
fg_setind(index);                /* set drawing index   */
fg_clear(0,7);
fg_sboard(0);
fg_sbuf(1);
fg_chan(1);
fg_sync(0);
strcat(s,nam);                            /* s = \pip\i\deboli  */
fg_frdisk(1024,0,s,dff,-1);      /* call up image */

/*****************************************************************************/

/*** Checking for 3D co-ords for object points entered ********************/
strcpy(str,"3d.txt");
strcat(nam,str);

if ((fp = fopen(nam,"rt")) == NULL)  {
   printf(" 3D object point co-ordinates are required for the outline calculation. \n");
   printf(" These still have to be calculated \n");
   printf(" Do you wish to caculate the 3D co-ordinates ? \n");
   printf(" And continue with the outline y,n :");
   cht=getch();
   while(cht !='y' && cht !='n' && cht !='Y' && cht !='N') {
        cht=getch();
        }
```

```
   if( cht == 'y' || cht == 'Y') {
     pmenu();
     }
   }
else {
  cht='y';
  }
fclose(fp);

/*****************************************************************************/
if(cht=='y' || cht=='Y') {
   mode();
   printf(" DIGITISNG BABIES OUTLINE \n");
   printf(" ======================== \n");
   printf(" \n");
   printf(" Only one outline has to be digitised \n");
   printf(" i.e. the outline on the left or right image only  \n");
   printf(" \n");
   printf(" Digitise two diagonally opposite control points \n");
   printf(" i.e. in shoulder region and opposite hip region on the image \n");
   printf(" \n");
   printf(" Enter point no. in shoulder region : ");
   scanf("%d",&nu1);
   printf(" Enter point no. in opposite hip region : ");
   scanf("%d",&nu2);
   printf(" \n");
   printf(" Then digitise baby's outline \n");
   printf("  \n");
   printf(" Outline digitising runs on point mode or stream mode \n");
   printf(" Point mode  - cross point co-ordinates are only stored when <ENTERed> \n");
   printf(" Stream mode - all cross point co-ordinates are stored\n");
   printf("  \n");
   printf(" To run different modes enter \n");
   printf(" --------------------------- \n");
   printf(" Point mode  = x \n");
   printf(" Stream mode = z \n");
   printf("  \n");
   printf("  \n");
   printf(" Press SPACE BAR to continue : ");
   ch = bdos(7,0,0);
   while( ch!= 32 ) {
     ch = bdos(7,0,0);
     }
   printf(" \n");
   strcpy(str1," DO 2 CONTROL");          /* partial heading for graphic display */
   strcpy(str2," ===========");           /* on cursor movements                 */

   num1[1] = 1;
   num1[2] = 2;
   num1[3] = 3;

   mice(str1,str2);                      /* graphic display on cursor */

   x = 256;
   y = 256;
   pixread(x,y,pixr,pixc);              /* reading 1st cross's pixel values */
   cross(x,y);
```

```
p = 24;                                 /* number of first pt. displayed */
q = 5;                                  /* in graphic display */
gotto(p,q);
printf("%2d \r",num1[1]);


while( kill != 'q' ) {                  /* movement options from keyboard */

  switch(ch){
    case 'S':                           /* Delete key - move 25 left */
    case 46:
       x=x+5;
       pixread(x,y,pixr,pixc);
       cross(x,y);
       if ( out1 == 'z') {
          num1[num] = num;
          outx[num] = x;
          outy[num] = y;
          num++;
       }
       break;
    case 82:                        /* 'R'Insert key - move 25 right */
    case 48:
       x=x-5;
       pixread(x,y,pixr,pixc);
       cross(x,y);
       if ( out1 == 'z') {
          num1[num] = num;
          outx[num] = x;
          outy[num] = y;
          num++;
       }
       break;
    case 'I':                         /* Page Up - move 25 up */
    case 57:
       y=y-5;
       pixread(x,y,pixr,pixc);
       cross(x,y);
       if ( out1 == 'z') {
          num1[num] = num;
          outx[num] = x;
          outy[num] = y;
          num++;
       }
       break;
    case 'Q':                         /* Page Down - move 25 down */
    case 51:
       y=y+5;
       pixread(x,y,pixr,pixc);
       cross(x,y);
       if ( out1 == 'z') {
          num1[num] = num;
          outx[num] = x;
          outy[num] = y;
          num++;
       }
       break;
    case 77:                        /*'M' Left arrow key - move 1 left */
    case 54:
```

```
      x=x+1;
      pixread(x,y,pixr,pixc);
      cross(x,y);
      if ( outl == 'z') {
         numl[num] = num;
         outx[num] = x;
         outy[num] = y;
         num++;
      }
      break;
   case 75:                    /* 'K'Right arrow key - move 1 right */
   case 52:
      x=x-1;
      pixread(x,y,pixr,pixc);
      cross(x,y);
      if ( outl == 'z') {
         numl[num] = num;
         outx[num] = x;
         outy[num] = y;
         num++;
      }
      break;
   case 72:                    /* 'H'Up arrow key - move 1 up */
   case 56:
      y=y-1;
      pixread(x,y,pixr,pixc);
      cross(x,y);
      if ( outl == 'z') {
         numl[num] = num;
         outx[num] = x;
         outy[num] = y;
         num++;
      }
      break;
   case 80:                    /* 'P'Down arrow key - move 1 down */
   case 50:
      y=y+1;
      pixread(x,y,pixr,pixc);
      cross(x,y);
      if ( outl == 'z') {
         numl[num] = num;
         outx[num] = x;
         outy[num] = y;
         num++;
      }
      break;
   case '6':                    /* Home key - move to top of screen */
   case 55:
      x=20;
      y=20;
      pixread(x,y,pixr,pixc);
      cross(x,y);
      break;
   case '0':                    /* End key - move to bottom of screen */
   case 49:
      x=490;
      y=490;
      pixread(x,y,pixr,pixc);
      cross(x,y);
```

```
    break;
case '5':                       /* '5' numeric pad key - move to screen centre */
    x=256;
    y=256;
    pixread(x,y,pixr,pixc);
    cross(x,y);
    break;
case 'z':                     /* 'z' key - point to stream mode */
case 'Z':
    pixread(x,y,pixr,pixc);
    cross(x,y);
    outl = 'z';
    numl[num] = num;
    outx[num] = x;
    outy[num] = y;
    num++;
    p = 24;                                 /* number of first pt. displayed */
    q = 5;                                  /* in graphic display */
    gotto(p,q);
    printf("mode Z \r");
    break;
case 'x':                       /* 'x' key - stream to point mode */
case 'X':
    pixread(x,y,pixr,pixc);
    cross(x,y);
    outl = 'a';
    p = 24;                                 /* number of first pt. displayed */
    q = 5;                                  /* in graphic display */
    gotto(p,q);
    printf("mode X \r");
    break;
case 13:                        /* '13' numeric pad key - ENTER */
    numm = numl[num];
    if(numl[num] <  3) {
        pera(x,y,num,numm,pixr,pixc,outx,outy);
        }
    cross(x,y);
    p = 24;
    q = 5;
    gotto(p,q);
    if(numl[num] < 3) printf("%2d \r ",numl[num+1]);
    num++;
    if(numl[num] == 3) {
      mode();
      strcpy(str1," z - STREAM MODE");          /* partial heading for graphic display */
      strcpy(str2," x - POINT MODE");           /* on cursor movements */
      mice(str1,str2);                        /* graphic display on cursor */
      gotto(p,q);
      printf("mode X \r");
   }
    break;
case 'q':                       /* 'q' key - to quit */
    kill = 'q';
    continue;
    default:
      ;
}
ch = bdos(7,0,0);
```

```
      pixwrit(x,y,pixr,pixc);
      }

   ended = num;                        /* set counter one back */
   nun = nun - 1;
   num1[1] = nu1;                   /* allocating shoulder and hip point numbers */
   num1[2] = nu2;


/*****************************************************************************/
/*** CALCULATING THE OUTLINE FOR THE BABIES ********************************/

   if ((fp = fopen(nam,"rt")) == NULL)  {
      puts("cannot open file \n");
      exit();
      }

   fscanf( fp," %03d ", &nn );
   for (nu=0; (nu<nn) && (fscanf( fp," \n %03d %f %f %f \n", &n[nu], &xx[nu], &yy[nu], &zz[nu])) ; nu++);

   B[0][0]=xx[nu1-1];
   B[1][0]=yy[nu1-1];
   B[2][0]=xx[nu2-1];
   B[3][0]=yy[nu2-1];


   /* Systems are different. 3D system     - y increases upwards */
   /*                        Screen system - y decreases upwards */
   /* change sign on outy[] to change system                     */

   X[0][0]=outx[1];
   X[0][1]=(-outy[1]);
   X[0][2]=1;
   X[0][3]=0;

   X[1][0]=(-outy[1]);
   X[1][1]=-outx[1];
   X[1][2]=0;
   X[1][3]=1;

   X[2][0]=outx[2];
   X[2][1]=(-outy[2]);
   X[2][2]=1;
   X[2][3]=0;

   X[3][0]=(-outy[2]);
   X[3][1]=-outx[2];
   X[3][2]=0;
   X[3][3]=1;


   /* mat Y = INV(X)*******/
   for(nu=0;nu<4;nu++) {
       qq=1/X[nu][nu];
       X[nu][nu]=qq;
       for(j=0;j<4;j++) {
         if(j != nu) {
            X[nu][j]=X[nu][j]*qq;
            }
```

```
        }
      for(j=0;j<4;j++) {
         if(j != nu) {
            qq=X[j][nu];
            X[j][nu]=0;
            for(k=0;k<4;k++) {
               X[j][k]=X[j][k]-X[nu][k]*qq;
               }
            }
          }
       }


   /* mat A = Y * B **    Y = X *******/
      for(nu=0;nu<4;nu++) {
         for(j=0;j<1;j++)  {
              A[nu][j]=0;
              for(k=0;k<4;k++) {
                  A[nu][j]=A[nu][j]+X[nu][k]*B[k][j];
                  }
              }
         }

   /************************************************************/


   for(nu=3;nu<ended;nu++) {
       Xc[nu-3]=A[0][0]*outx[nu]+A[1][0]*(-outy[nu])+A[2][0];
       Yc[nu-3]=A[0][0]*(-outy[nu])-A[1][0]*outx[nu]+A[3][0];
       }

   /**** outputting pixel positions ****************************************/
   /************************************************************************/

   if ((fp = fopen(name,"w+")) == NULL)  {
      puts("cannot open file \n");
      exit();
      }

   fprintf( fp," %03d ", num-3 );
   for (num=1;(num<ended-3) && (fprintf( fp," \n %03d %6.1f %6.1f                    ", num, Xc[num], Yc[num])) ; num++);

   fclose(fp);
   kill = ' ';              /* to be able to rerun program */
   fg_exit();
 }                          /* end of very big if loop, line 111 to line 357 */

}
/******* END OF MAIN ******************************/
```

APPENDIX 15 CURSOR - TAG PROGRAM TO OPERATE DIGITISER

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>

int pixc[20];
int pixr[20];
float outx[60];
float outy[60];
float mata[13][13];
int x, y;
int a, b, c, d, e, f, i, p, q, h, j;
int no, num, numa, grey;
char chnum[5];
int ended;

void cursor(name,buf,dff,num1,count,str1,str2)
int num1[30];
{

void gotto(int,int);

FILE *fp;                    /* file description for outputting data */
char ch = 't';
char chnum[5];
char s[20];
char kill;
int pixr[20];
int pixc[20];
float outx[60];
float outy[60];
int mata[13][13];
int p,q;
int x = 0;
int y = 0;
int a = 0;
int b = 0;
int c = 0;
int d = 0;
int e = 0;
int f = 0;

int i = 0;
int num = 1;
int numa = 0;
int ended = 0;

keys(str1,str2);

x = 256;
y = 256;
put_pixread(x,y,pixr,pixc);              /* reading 1st cross's pixel values */
put_cross(x,y);

p = 24;
q = 5;
gotto(p,q);
```

```
put_pixread(x,y,pixr,pixc);              /* reading 1st cross's pixel values */
put_cross(x,y);

p = 24;
q = 5;
gotto(p,q);
printf("%2d \r",num1[1]);

while( kill != 'q' ) {                          /* movement options from keyboard */

    switch(ch){
      case 'S':                              /* Delete key - move 25 left */
      case 46:
         x=x+25;
         put_pixread(x,y,pixr,pixc);
         put_cross(x,y);
         break;
      case 82:                          /* 'R'Insert key - move 25 right */
      case 48:
         x=x-25;
         put_pixread(x,y,pixr,pixc);
         put_cross(x,y);
         break;
      case 'I':                          /* Page Up - move 25 up */
      case 57:
         y=y-25;
         put_pixread(x,y,pixr,pixc);
         put_cross(x,y);
         break;
      case 'Q':                          /* Page Down - move 25 down */
      case 51:
         y=y+25;
         put_pixread(x,y,pixr,pixc);
         put_cross(x,y);
         break;
      case 77:                          /*'M' Left arrow key - move 1 left */
      case 54:
         x=x+1;
         put_pixread(x,y,pixr,pixc);
         put_cross(x,y);
         break;
      case 75:                          /* 'K'Right arrow key - move 1 right */
      case 52:
         x=x-1;
         put_pixread(x,y,pixr,pixc);
         put_cross(x,y);
         break;
      case 72:                          /* 'H'Up arrow key - move 1 up */
      case 56:
         y=y-1;
         put_pixread(x,y,pixr,pixc);
         put_cross(x,y);
         break;
      case 80:                          /* 'P'Down arrow key - move 1 down */
      case 50:
         y=y+1;
         put_pixread(x,y,pixr,pixc);
         put_cross(x,y);
         break;
```

```
      case '6':                        /* Home key - move to top of screen */
      case 55:
         x=20;
         y=20;
         put_pixread(x,y,pixr,pixc);
         put_cross(x,y);
         break;
      case '0':                        /* End key - move to bottom of screen */
      case 49:
         x=490;
         y=490;
         put_pixread(x,y,pixr,pixc);
         put_cross(x,y);
         break;
      case '5':                        /* '5' numeric pad key - move to screen centre */
         x=256;
         y=256;
         put_pixread(x,y,pixr,pixc);
         put_cross(x,y);
         break;
      case 13:                        /* '13' numeric pad key - ENTER */
         numm = numl[num];
         put_pixwrit(x,y,pixr,pixc);
         put_pera(x,y,nua,numm,pixr,pixc,outx,outy);
         put_cross(x,y);
         num++;
         p = 24;
         q = 5;
         gotto(p,q);
         if(numl[num] != 0) printf("%2d \r ",numl[num]);
         if(numl[num] == 0) {
             ended = nua;
             nua = nua - 1;
             kill = 'q';
             continue;
             }
         break;
      case 'q':                        /* 'q' key - to quit */
         ended = nua;
         nua = nua - 1;
         kill = 'q';
         continue;
         default:
           ;
      }
   ch = bdos(7,0,0);

   put_pixwrit(x,y,pixr,pixc);
}

/**** outputting pixel positions ******/


if ((fp = fopen(name,"w+")) == NULL)  {
   puts("cannot open file \n");
   exit();
   }

fprintf( fp," %03d ", nua );
```

```
for (num=1; (num<ended) && (fprintf( fp," \n %03d %6.2f %6.2f                    ", num1[num], outx[num], outy[num])) ; num++);

fclose(fp);
kill = ' ';             /* to be able to rerun program */
fg_exit();
}
/******* END OF MAIN ******************************/


/******* PIXEL WRITE ******************************/


put_pixwrit(x,y,pixr,pixc)                  /* replacing pixels for each cross */
int pixr[20];
int pixc[20];                               /* movement case */
{
  i = 0;
  a = x - 7;                    /* calculating cross position */
  b = y - 7;
  c = x + 8;
  d = y + 8;
  e = a;
  f = b;

 for (no=a; no<c; no=no+1) {
     fg_pixw(e,y,pixr[i]);      /* replacing pixels of previous cross */
     fg_pixw(x,f,pixc[i]);

     i++ ;
     e++;
     f++;
     }
}


/******* PIXEL READ ******************************/


put_pixread(x,y,pixr,pixc)                  /* reading cross for each key */
int pixr[20];
int pixc[20];                               /* movement case */
{
  i = 0;
  a = x - 7;                    /* calculating cross position */
  b = y - 7;
  c = x + 8;
  d = y + 8;
  e = a;
  f = b;


  for (no=a; no<c; no=no+1) {   /* reading next cross pixels */
     pixr[i]=fg_pixr(e,y);
     pixc[i]=fg_pixr(x,f);
     i++;
     e++;
     f++;
      }
```

```
  i = 0;

}

/******** PUT CURSOR  ********************/

put_cross(x,y)
{
  a = x - 7;
  b = y - 7;
  c = x + 7;
  d = y + 7;

  grey = fg_pixr(x,y);
  p = 24;
  q = 29;
  gotto(p,q);
  printf("%3d   %3d   %3d \r ", x, y, grey);

  if(grey<80) {
     grey = 255;
     }
  else {
     grey = 0;
     }
  fg_setind(grey);
  fg_pixw(x,y,grey);
  fg_pixw(x-1,y,grey);
  fg_pixw(x+1,y,grey);
  fg_pixw(x,y-1,grey);
  fg_pixw(x,y+1,grey);
  fg_moveto(a,y);              /* drawing cross */
  fg_lineto(x-4,y);
  fg_moveto(x+4,y);
  fg_lineto(c,y);
  fg_moveto(x,d);              /* drawing cross */
  fg_lineto(x,y+4);
  fg_moveto(x,y-4);
  fg_lineto(x,b);
}

/****** CURSOR SAVE **************/

put_para(x,y,num,numm,pixr,pixc,outx,outy)
int pixr[20];
int pixc[20];
float outx[60];
float outy[60];
{
   FILE *fp;
   float mata[20][20];
   float M,N,G,x1,y11;
   int k,l;
   int gy,temp;
```

```
 i = 0;
 a = x - 7;                    /* calculating cross position */
 b = y - 7;
 c = x + 8;
 d = y + 8;
 e = a;
 f = b;
 N = 0;
 M = 0;
 G = 0;
 h = 0;
 j = 0;

 grey = fg_pixr(x,y);

/* Calculating centre of gravity of target ************************/


 gy = fg_pixr(x-4,y-4);
 h=0;
 for (k=y-4; k< (y+5); k++) {          /* finding highest grey value */
   j = 0;
   for (l=x-4; l< (x+5); l++) {
        temp = fg_pixr(l,k);
        if( temp > gy )
            gy=temp;
        j++;
      }
    h++;
    }

 h=0;
 for (k=y-4; k<(y+5);k++) {
   j = 0;
   for (l=x-4; l<(x+5);l++) {
        mata[h][j] = fg_pixr(l,k);
        if( mata[h][j] <= (gy-100) )
            mata[h][j]=0;
        if( j < 2 && j >6 && mata[h][j] >(gy-100) )
            mata[h][j] = 0;
        if( h < 2 && h >6 && mata[h][j] >(gy-100) )
            mata[h][j] = 0;
        j++;
      }
    h++;
    }

 for (h=0; h<9; h++) {
    for (j=0; j<9; j++) {
      N = N + mata[h][j] * (h+1);
      M = M + mata[h][j] * (j+1);
      G = G + mata[h][j];
      }
    }

 x1 = N / G;
 y11 = N / G;
 x1 = (x-4) + x1 - 0.5;
 y11 = (y-4) + y11 - 0.5;
```

```
    outx[num] = x1;
    outy[num] = y11;

   if(grey<80) {
      grey = 255;
      }
   else {
      grey = 0;
      }
   fg_setind(grey);


   for (no=a; no<c; no=no+1) {   /* reading next cross pixels */
     pixr[i]= grey;               /*  was 255 */
     pixc[i]= grey;               /*  was 255 */
     i++;
     e++;
     f++;
      }


  itoa(numm, chnum,10);          /* converts int to char to print as text */

   grey = fg_pixr(x-7,y-6);
   if(grey<80) {
      grey = 100;
      }
   else {
      grey = 0;
      }
  fg_setind(grey);
  fg_moveto(x-7,y-3);
  fg_text(chnum,1);

  i = 0;

  p = 24;
  q = 52;
  gotto(p,q);
  printf("%3d    %3.1f    %3.1f \r",numm,x1,y11);

}
```

APPENDIX 16 BPARAMS.C - TAG PROGRAM TO CALCULATE B PARAMETERS

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>
#include <malloc.h>
#include <math.h>

/* this program undertakes a simple projective transformation for a */
/* two camera case in which it is assumed that the same control points */
/* have been observed in the same order in each image */


void bparams()
{
  void mode();

  float x[50],y[50],z[50],x1[50],y1[50],z1[50];
  float p[50],q[50],p2[50],q2[50],m1[11][1],m2[11][1];
  float m3[4][3],l3[4][1],m4[3][3],m5[3][1];
  float Q2[50],m[11][1],s1[50],t[50],v[50];
  float x7[50],y7[50],z7[50],a3[3][4],a4[3][3],a1[3][3],b2[3][4];
  float f[100];
  int noc[50];
  int no1[50],no2[50];
  int no,n2,n3,n;
  int num,ab,ac,ex,i,ii,abcd;
  float x9,y9,z9;
  float tt,tac;
  int ia,ja,ib,jb,ic,jc;
  int zt;
  int jl,k;
  float ql;
  float x4[4],y4[4],no4[4],xtl,ytl;
  float x5[4],y5[4],no5[4],xtr,ytr;
  float oa[50],ob[50],cxl[4],cyl[4],cxr[4],cyr[4];
  float xx,yy,yx,xy,sum,aa,bb,xt,yt;
  int ax;
  char nam[20];
  char nam1[20], nam2[20];
  char na[20];
  char s[20];
  char bth[20];
  char ch,cls;
  char fl[15],fr[15],cl[15],cr[15],lhb[15],rhb[15];
  char lobj[15],robj[15],d3[15];
  char c7 = (205);


  FILE *fp;
  FILE *ft;

  mode();

  printf("                                                                \n");
  printf("                                                                \n");
  printf("                                                                \n");
```

```
printf("  ┌──────────────────────────────────────────────────┐  \n");
printf("  │                                                  │  \n");
printf("  │                                                  │  \n");
printf("  │            CHECKING ACCURACY OF CONTROL          │  \n");
printf("  │                                                  │  \n");
printf("  │                                                  │  \n");
printf("  │            CALCULATING CAMERA PARAMETERS         │  \n");
printf("  │                                                  │  \n");
printf("  │                                                  │  \n");
printf("  └──────────────────────────────────────────────────┘  \n");
printf("  \n");
printf("  \n");
printf("   PATIENT'S NAME : ");
scanf("%s",naa);                        /* naa = deb */
printf("  \n");
printf("   CO-ORDINATING SMALL OR LARGE CONTROL s/l : ");
cls = getch();
while( cls !='s' && cls !='l' && cls !='S' && cls !='L' ) {
   cls = getch();
   }
printf("  \n\n");
printf("   SOFT AND HARDCOPY OUTPUT OF ACCURACIES ? y,n : ");
abcd = getch();
while( abcd !='y' && abcd !='Y' && abcd !='n' && abcd !='N' ) {
   abcd = getch();
   }
printf("  \n");
printf("  \n");
printf("  \n");
if( abcd == 89 || abcd == 121 ) {
   printf("   ENSURE THAT PRINTER IS CONNECTED AND ON LINE \n");
   printf("   Press SPACE BAR to continue : ");
   ch = bdos(7,0,0);
   while( ch !=32 ) {
       ch = bdos(7,0,0);
       }
   printf(" \n");
   fprintf(stdprn,"  ┌──────────────────────────────────────────────┐  \n");
   fprintf(stdprn,"  │                                              │  \n");
   fprintf(stdprn,"  │                                              │  \n");
   fprintf(stdprn,"  │      BABY'S NAME : %s                        │  \n",naa);
   fprintf(stdprn,"  │                                              │  \n");
   fprintf(stdprn,"  └──────────────────────────────────────────────┘  \n");
   fprintf(stdprn,"  \n");
   fprintf(stdprn,"  \n");
   }
printf("   BUSY !!!!! \n");


/*** READING BABY CONTROL FILE ***********************************************/

if(cls=='s' || cls=='S') {
   strcpy(na,"BABY.BYT");     /* file of 3D co-ordinates of control */
   if((fp=fopen(na,"rb")) == NULL) {       /* r-read , b-byte file */
      puts("cannot open file \n");
      exit();
      }
   fread(f,sizeof(float),55,fp);
   }
else if(cls=='l' || cls=='L') {
```

```
    strcpy(na,"BABYLAR6.BYT");     /* file of 3D co-ordinates of control */
    if((fp=fopen(na,"rb")) == NULL) {       /* r-read , b-byte file */
      puts("cannot open file \n");
      exit();
      }
    fread(f,sizeof(float),73,fp);
    }
  n2=f[0];
  for(no=0;no<n2;no++)  {
    x1[no]=f[1+3*no];
    y1[no]=f[2+3*no];
    z1[no]=f[3+3*no];
    }
  fclose(fp);
/*** CREATING FILE EXTENSIONS ***************************/

 strcpy(nam1,nam);          /* nam1 = deb */
 strcpy(s,"CL.TXT");        /* s = cl.txt */
 strcat(nam1,s);            /* nam1 = debcl.txt */
 strcpy(cl,nam1);           /* cl = debcl.txt */

 strcpy(nam1,nam);          /* nam1 = deb */
 strcpy(s,"CR.TXT");        /* s = cr.txt */
 strcat(nam1,s);            /* nam1 = debcr.txt */
 strcpy(cr,nam1);           /* cr = debcr.txt */

 strcpy(nam1,nam);          /* nam1 = deb */
 strcpy(s,"LHB.TXT");       /* s = lhb.txt */
 strcat(nam1,s);            /* nam1 = deblhb.txt */
 strcpy(lhb,nam1);          /* lhb = deblhb.txt */

 strcpy(nam1,nam);          /* nam1 = deb */
 strcpy(s,"RHB.TXT");       /* s = rhb.txt */
 strcat(nam1,s);            /* nam1 = debrhb.txt */
 strcpy(rhb,nam1);          /* rhb = debrhb.txt */


/*************************************************************************/
/* now reading 2d coordinates for left and right camera, control */
/* cl.txt files -- control left text files*****************************/

  if((fp=fopen(cl,"rt")) == NULL) {     /* rt r-read , t-text file */
    puts("cannot open file \n");
    exit();
    }

  fscanf(fp," %03d ", &n2);

  for(no=0;no<n2;no++) {
      fscanf(fp," %03d %f %f \n", &no1[no],&p[no],&q[no]);
      p[no]= p[no] * 0.015;   /*  1.059   for baby image,   0.015  for proton */
      q[no]=-q[no] * 0.011;     /*  0.749                     0.011            */
      no1[no] = no1[no]-1;     /* Due to matrix numbering from 0 */
      x[no] = x1[no1[no]];
```

```
      y[no] = y1[no1[no]];
      z[no] = z1[no1[no]];
      }
  fclose(fp);

 sub(x,y,z,p,q,n2,a1);

/*****  Now save b parameters on separate files  ********************/
if((fp=fopen(lhb,"w+")) == NULL) {     /* w+ -write , t-text file */
     puts("cannot open file \n");
     exit();
     }
for(i=0;i<11;i++) {
    fprintf(fp,"%10.6f \n",a1[i][0]);
    }
fclose(fp);

  for(no=0;no<n2;no++) {
     p2[no]=p[no];
     q2[no]=q[no];
     }


/*************************************************************************/
/* now reading 2d coordinates for left and right camera, control */
/* cr.txt files --  control right text files*****************************/

  if((fp=fopen(cr,"rt")) == NULL) {     /* rt r-read , t-text file */
     puts("cannot open file \n");
     exit();
     }

  fscanf(fp," %03d \n", &n3);

  for(no=0;no<n3;no++) {
      fscanf(fp," %03d %f %f \n", &no2[no],&p[no],&q[no]);
      p[no]= p[no] * 0.015;   /*  1.059   for baby image,   0.015  for proton */
      q[no]=-q[no] * 0.011;     /*  0.749                     0.011            */
      no2[no] = no2[no]-1;     /* Due to matrix numbering from 0 */
      x[no] = x1[no2[no]];
      y[no] = y1[no2[no]];
      z[no] = z1[no2[no]];
      }
  fclose(fp);

  sub(x,y,z,p,q,n3,a);

/*****  Now save b parameters on separate files  ********************/
if((fp=fopen(rhb,"w+")) == NULL) {     /* w+ -write , t-text file */
     puts("cannot open file \n");
     exit();
     }
for(i=0;i<11;i++) {
    fprintf(fp,"%10.6f \n",a[i][0]);
    }
fclose(fp);
```

```
        /*************************************************************************/
/**   determining common control points   ********************************/

  ab = 0;
  ac = 0;
  for(no=0;no<n2;no++) {
    if( no1[no] > no2[ab]) {
       ab = ab + 1;
       }
    if(no1[no] == no2[ab]) {
         noc[ac]=no1[no];
         p2[ac] = p2[no];
         q2[ac] = q2[no];
         x[ac] = x1[no1[no]];
         y[ac] = y1[no1[no]];
         z[ac] = z1[no1[no]];
         p[ac] = p[ab];
         q[ac] = q[ab];
         ab = ab + 1;
         ac = ac + 1;
         }
     }


  for(no=0;no<ac;no++) {
      no1[no] = noc[no];
      no2[no] = noc[no];
      }


  /*************************************************************************/
  /*  FOR PRINTOUT OF ACCURACIES ON THE PRINTER  */
  for(ii=0;ii<ac;ii++) {
     a3[0][0] = p2[ii] * a1[8][0] - a1[0][0];
     a3[0][1] = p2[ii] * a1[9][0] - a1[1][0];
     a3[0][2] = p2[ii] * a1[10][0] - a1[2][0];
     l3[0][0] = a1[3][0] - p2[ii];
     a3[1][0] = q2[ii] * a1[8][0] - a1[4][0];
     a3[1][1] = q2[ii] * a1[9][0] - a1[5][0];
     a3[1][2] = q2[ii] * a1[10][0] - a1[6][0];
     l3[1][0] = a1[7][0] - q2[ii];
     a3[2][0] = p[ii] * a[8][0] - a[0][0];
     a3[2][1] = p[ii] * a[9][0] - a[1][0];
     a3[2][2] = p[ii] * a[10][0] - a[2][0];
     l3[2][0] = a[3][0] - p[ii];
     a3[3][0] = q[ii] * a[8][0] - a[4][0];
     a3[3][1] = q[ii] * a[9][0] - a[5][0];
     a3[3][2] = q[ii] * a[10][0] - a[6][0];
     l3[3][0] = a[7][0] - q[ii];


      /**** TRANSPOSE   MAT A3  = TRANS M3 ************************/

       for(i=0;i<4;i++) {
         for(jl=0;jl<3;jl++) {
           a3[jl][i] = a3[i][jl];
           }
         }
```

```
/$$$$ MULT  MAT A4 = A3 $ M3 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$/
   for(i=0;i<3;i++) {
     for(jl=0;jl<3;jl++) {
       a4[i][jl] = 0;
       for(k=0;k<4;k++) {
          a4[i][jl]=a4[i][jl]+a3[i][k]$m3[k][jl];
          }
       }
     }


/$$$$$$$$$ INVERS  MAT a4 = INV a4 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$/
   for(i=0;i<3;i++) {
     for(jl=0;jl<3;jl++) {
       al[i][jl] = a4[i][jl];
       }
     }

   for(i=0;i<3;i++) {
     ql = 1/al[i][i];
     al[i][i] = ql;
     for(jl=0;jl<3;jl++) {
       if(jl != i) {
          al[i][jl] = al[i][jl] $ ql;
          }
       }
       for(jl=0;jl<3;jl++) {
         if(jl != i) {
           ql = al[jl][i];
           al[jl][i] = 0;
           for(k=0;k<3;k++) {
              al[jl][k] = al[jl][k] - al[i][k] $ ql;
              }
           }
         }
       }

   for(i=0;i<3;i++) {
     for(jl=0;jl<3;jl++) {
       a4[i][jl] = al[i][jl];
       }
     }



/$$$$ MULT  MAT B2 = M4 $ A3 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$/
    for(i=0;i<3;i++) {
      for(jl=0;jl<4;jl++) {
        b2[i][jl] = 0;
        for(k=0;k<3;k++) {
          b2[i][jl]=b2[i][jl]+m4[i][k]$a3[k][jl];
          }
        }
      }


/$$$$ MULT  MAT M5 = B2 $ L3 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$/
```

```
      for(i=0;i<3;i++) {
        for(jl=0;jl<1;jl++) {
          a5[i][jl] = 0;
          for(k=0;k<4;k++) {
             a5[i][jl]=a5[i][jl]+b2[i][k]*l3[k][jl];
             }
          }
        }

    sl[ii] = a5[0][0];
    t[ii] = a5[1][0];
    v[ii] = a5[2][0];
}


 node();
 printf(" \n");
 printf("  CHECK ON ACCURACY USING COMMON CONTROL POINTS \n");
 printf("  ============================================ \n");
 printf(" \n");
 printf("        NO.     DX      DY      DZ     (mm)      \n");
 printf("  ============================================ \n");


 if(abcd == 89 || abcd == 121) {
  fprintf(stdprn," \n");
  fprintf(stdprn,"  CHECK ON ACCURACY USING COMMON CONTROL POINTS \n");
  fprintf(stdprn,"  -------------------------------------------- \n");
  fprintf(stdprn," \n");
  fprintf(stdprn,"        NO.     DX      DY      DZ     (mm) \n");
  fprintf(stdprn,"  -------------------------------------------- \n");
  }

 tt = 19;
 for(i=0;i<ac;i++) {
   x9 = sl[i]-x[i];
   y9 = t[i]-y[i];
   z9 = v[i]-z[i];

   ii = i+1;
   tac = ii;
   if( ii > 18 && fmod(tac,tt)==0 ) {
     printf(" \n");
     printf("  Press SPACE BAR to continue : ");
     ch = bdos(7,0,0);
     while( ch !=32 ) {
          ch = bdos(7,0,0);
          }
     printf(" \n");
     node();
     printf(" \n");
     printf("  CHECK ON ACCURACY USING COMMON CONTROL POINTS \n");
     printf("  ============================================ \n");
     printf(" \n");
     printf("        NO.     DX      DY      DZ     (mm)      \n");
     printf("  ============================================ \n");
     }
   printf("       %3d    %4.1f    %4.1f    %4.1f  \n",noc[i]+1,x9,y9,z9);
```

```
    if(abcd == 89 || abcd == 121) {
      fprintf(stdprn, "      %3d    %4.1f    %4.1f    %4.1f \n",noc[i]+1,x9,y9,z9);
      }
    }

  if(abcd==89 || abcd == 121) {
     fprintf(stdprn," \n");
     fprintf(stdprn," \n");
     fprintf(stdprn," \n");
     }


printf(" \n");
printf("  Press SPACE BAR to continue : ");
ch = bdos(7,0,0);
while( ch !=32 ) {
    ch = bdos(7,0,0);
    }
printf(" \n");
mode();
/* END OF MAIN */
}


sub(x,y,z,p,q,n,m)
float x[50];
float y[50];
float z[50];
float p[50];
float q[50];
float m[1][1];
{
   float g[11][50];
   float a[11][11];
   float at[11][11];
   float r[50][1];
   float l[11][1];
   float b[11][11];
   float d[50][11];
   int ia,ja,ib,jb,ic,jc;
   int i,jl,k;
   float qt;

   for(i=0;i<n;i++) {
        d[i][0]=x[i];
        d[i][1]=y[i];
        d[i][2]=z[i];
        d[i][3]=1;
        d[i][4]=0;
        d[i][5]=0;
        d[i][6]=0;
        d[i][7]=0;
        d[i][8]=-p[i]*x[i];
        d[i][9]=-p[i]*y[i];
        d[i][10]=-p[i]*z[i];
        d[n+i][0]=0;
        d[n+i][1]=0;
        d[n+i][2]=0;
```

```
        d[n+i][3]=0;
        d[n+i][4]=x[i];
        d[n+i][5]=y[i];
        d[n+i][6]=z[i];
        d[n+i][7]=1;
        d[n+i][8] =-q[i]*x[i];
        d[n+i][9]=-q[i]*y[i];
        d[n+i][10]=-q[i]*z[i];
        }

/**** TRANSPOSE   MAT G = TRANS D ************************/

       for(i=0;i<n*2;i++) {
         for(jl=0;jl<11;jl++) {
           g[jl][i] = d[i][jl];
           }
         }

/**** MULT  MAT A = G * D ********************************/
       for(i=0;i<11;i++) {
         for(jl=0;jl<11;jl++) {
           a[i][jl] = 0;
           for(k=0;k<n*2;k++) {
              a[i][jl]=a[i][jl]+g[i][k]*d[k][jl];
              }
           }
         }

/*****************************************************************/

      for(i=0;i<n;i++) {
         r[i][0]=p[i];
         r[n+i][0]=q[i];
         }

/**** MULT MAT L = G * R ************************************/

       for(i=0;i<11;i++) {
         for(jl=0;jl<1;jl++) {
           l[i][jl] = 0;
           for(k=0;k<n*2;k++) {
              l[i][jl]=l[i][jl]+g[i][k]*r[k][jl];
              }
           }
         }

/********** INVERS  MAT B = INV A *******************************/
       for(i=0;i<11;i++) {
         for(jl=0;jl<11;jl++) {
           at[i][jl] = a[i][jl];
           }
         }

       for(i=0;i<11;i++) {
         qt = 1/at[i][i];
         at[i][i] = qt;
         for(jl=0;jl<11;jl++) {
           if(jl != i) {
              at[i][jl] = at[i][jl] * qt;
```

```
            }
          }
          for(jl=0;jl<11;jl++) {
            if(jl != i) {
              qt = at[jl][i];
              at[jl][i] = 0;
              for(k=0;k<11;k++) {
                 at[jl][k] = at[jl][k] - at[i][k] * qt;
                 }
              }
            }
          }

        for(i=0;i<11;i++) {
          for(jl=0;jl<11;jl++) {
            b[i][jl] = at[i][jl];
            }
          }


/**** MULT MAT M = B * L ************************************/
        for(i=0;i<11;i++) {
          for(jl=0;jl<1;jl++) {
            a[i][jl] = 0;
            for(k=0;k<11;k++) {
               a[i][jl]=a[i][jl]+b[i][k]*l[k][jl];
               }
            }
          }
}
```

APPENDIX 17 D3.C - TAG PROGRAM TO CALCULATE 3D CO-ORDINATES

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>
#include <malloc.h>
#include <math.h>

/* this program undertakes a simple projective transformation for a */
/* two camera case in which it is assumed that the same control points */
/* have been observed in the same order in each image */


void d3()
{
  void mode();
  float x[50],y[50],z[50],x1[50],y1[50],z1[50];
  float p[50],q[50],p2[50],q2[50],m1[11][1],m2[11][1];
  float m3[4][3],l3[4][1],m4[3][3],m5[3][1];
  float Q2[50],m[11][1],s1[50],t[50],v[50];
  float x7[50],y7[50],z7[50],a3[3][4],a4[3][3],a1[3][3],b2[3][4];
  float f[100];
  int noc[50];
  int no1[50],no2[50];
  int no,n2,n3,n;
  int num,ab,ex,i,ii,abcd;
  float x9,y9,z9;
  float tt,tac;
  int ia,ja,ib,jb,ic,jc;
  int zt;
  int jl,k;
  float ql;
  float x4[4],y4[4],no4[4],xtl,ytl;
  float x5[4],y5[4],no5[4],xtr,ytr;
  float oa[50],ob[50],cxl[4],cyl[4],cxr[4],cyr[4];
  float xx,yy,yx,xy,sum,aa,bb,xt,yt;
  int ax;
  char naa[20];
  char naa1[20], naa2[20];
  char na[20];
  char s[20];
  char bth[20];
  char ch;
  char fl[15],fr[15],cl[15],cr[15],lhb[15],rhb[15];
  char lobj[15],robj[15],d3[15];
  char c7 = (205);


  FILE *fp;
  FILE *ft;

  mode();

  printf("                                                    \n");
  printf("                                                    \n");
  printf("                                                    \n");
  printf("          CHECKING ACCURACY OF CONTROL              \n");
```

```
printf("  ┌──────────────────────────────────────────────────┐  \n");
printf("  │                                                  │  \n");
printf("  │                                                  │  \n");
printf("  │          & 3D CO-ORDINATES OF OBJECT POINTS      │  \n");
printf("  │                                                  │  \n");
printf("  │                                                  │  \n");
printf("  └──────────────────────────────────────────────────┘  \n");


printf("  \n");
printf("  \n");
printf("   PATIENT'S NAME : ");
scanf("%s",naa);                        /* naa = deb */
printf("  \n");
printf("   ENTER BREATH NUMBER   AND   IN OR OUT  i.e. 1i ? ");
scanf("%s",bth);
printf("  \n");
printf("   SOFT AND HARDCOPY OUTPUT OF 3D CO-ORDINATES ? y,n : ");
abcd = getch();
while( abcd !='y' && abcd !='Y' && abcd !='n' && abcd !='N' ) {
    abcd = getch();
    }
printf("  \n");
printf("  \n");
printf("  \n");

/*** CREATING FILE EXTENSIONS ***************************/

strcpy(nam1,naa);           /* nam1 = deb */
strcpy(s,"4L.TXT");         /* s = 4l.txt */
strcat(nam1,s);             /* nam1 = deb4l.txt */
strcpy(fl,nam1);            /* cl = deb4l.txt */

strcpy(nam1,naa);           /* nam1 = deb */
strcpy(s,"4R.TXT");         /* s = 4r.txt */
strcat(nam1,s);             /* nam1 = deb4r.txt */
strcpy(fr,nam1);            /* cl = deb4r.txt */

strcpy(nam1,naa);           /* nam1 = deb */
strcpy(s,"CL.TXT");         /* s = cl.txt */
strcat(nam1,s);             /* nam1 = debcl.txt */
strcpy(cl,nam1);            /* cl = debcl.txt */

strcpy(nam1,naa);           /* nam1 = deb */
strcpy(s,"CR.TXT");         /* s = cr.txt */
strcat(nam1,s);             /* nam1 = debcr.txt */
strcpy(cr,nam1);            /* cr = debcr.txt */

strcpy(nam1,naa);           /* nam1 = deb */
strcpy(s,"LHB.TXT");        /* s = lhb.txt */
strcat(nam1,s);             /* nam1 = deblhb.txt */
strcpy(lhb,nam1);           /* lhb = deblhb.txt */

strcpy(nam1,naa);           /* nam1 = deb */
```

```
  strcpy(s,"RHB.TXT");        /* s = rhb.txt */
  strcat(nam1,s);             /* nam1 = debrhb.txt */
  strcpy(rhb,nam1);           /* rhb = debrhb.txt */

  strcpy(nam2,nam);           /* nam2 = deb */
  strcat(nam2,"o");           /* nam2 = debo */
  strcat(nam2,bth);           /* nam2 = deboli */

  strcpy(nam1,nam2);          /* nam1=nam2=deboli */
  strcpy(s,"L.TXT");          /* s = l.txt */
  strcat(nam1,s);             /* nam1 = debolil.txt */
  strcpy(lobj,nam1);          /* lobj = debolil.txt */

  strcpy(nam1,nam2);          /* nam1=nam2=deboli */
  strcpy(s,"R.TXT");          /* s = r.txt */
  strcat(nam1,s);             /* nam1 = debolir.txt */
  strcpy(robj,nam1);          /* robj = debolir.txt */

  strcpy(nam1,nam2);          /* nam1=nam2=deboli */
  strcpy(s,"3D.TXT");         /* s = 3d.txt  */
  strcat(nam1,s);             /* nam1 = deboli3d.txt */
  strcpy(d3,nam1);            /* d3 = deboli3d.txt */

/*****************************************************************/

  if((fp=fopen(lhb,"rt")) == NULL) {     /* checking for b paraas */
     printf(" \n\n\n");
     printf("   B PARAMETERS ARE NEEDED TO RUN THIS PROGAM  \n");
     printf("  \n");
     printf("   B PARAMETERS HAVE NOT YET BEEN CALCULATED \n");
     printf("  \n");
     printf("   RUN 9.1 BEFORE 9.2 TO CALCULATE B PARAMETERS \n");
     printf("  \n");
     printf("  \n");
     printf("   Press SPACE BAR to continue : ");
     ch = bdos(7,0,0);
     while( ch !=32 ) {
        ch = bdos(7,0,0);
        }
     printf(" \n");
     }
 else {
     if( abcd == 89 || abcd == 121 ) {
        printf("   ENSURE THAT PRINTER IS CONNECTED AND ON LINE \n");
        printf("   Press SPACE BAR to continue : ");
        ch = bdos(7,0,0);
        while( ch !=32 ) {
            ch = bdos(7,0,0);
            }
        printf(" \n");
        fprintf(stdprn,"_____________________________________________ \n");
        fprintf(stdprn,"                                             \n");
        fprintf(stdprn,"                                             \n");
        fprintf(stdprn,"      BABY'S NAME : %s                       \n",nam);
        fprintf(stdprn,"      BREATH NUMBER AND IN OR OUT : %s       \n",bth);
        fprintf(stdprn,"                                             \n");
        fprintf(stdprn,"                                             \n");
        fprintf(stdprn,"_____________________________________________ \n");
```

```
   fprintf(stdprn,"                                                    \n");
   fprintf(stdprn,"                                                    \n");
   }
printf("   BUSY !!!!! \n");


/*** READING left hand B PARAMETERS  ********************************************/
if((fp=fopen(lhb,"rt")) == NULL) {     /* rt r-read , t-text file */
   puts("cannot open file \n");
   exit();
   }
for(no=0;no<11;no++) {
   fscanf(fp," %f \n", &m1[no][0]);
   }
fclose(fp);


/*** READING right hand B PARAMETERS  *******************************************/
if((fp=fopen(rhb,"rt")) == NULL) {     /* rt r-read , t-text file */
   puts("cannot open file \n");
   exit();
   }
for(no=0;no<11;no++) {
   fscanf(fp," %f \n", &m[no][0]);
   }
fclose(fp);


/*************************************************************************/
/* now reading 2d coordinates for left and right camera, fiducial */
/* fl.txt files -- fiducial left.text files*****************************/

if((fp=fopen(fl,"rt")) == NULL) {     /* rt r-read , t-text file */
   puts("cannot open file \n");
   exit();
   }

fscanf(fp," %03d ", &n2);

for(no=0;no<n2;no++) {
   fscanf(fp," %03d %f %f \n", &no4[no],&x4[no],&y4[no]);
   no4[no] = no4[no]-1;     /* Due to matrix numbering from 0 */
   }
fclose(fp);

xtl=0;
ytl=0;
for(no=0;no<n2;no++) {
   xtl=x4[no]+xtl;
   ytl=y4[no]+ytl;
   }
xtl=xtl/4;
ytl=ytl/4;
for(no=0;no<n2;no++) {
   cxl[no]=x4[no]-xtl;
   cyl[no]=y4[no]-ytl;
   }
```

```
/*************************************************************************/
/* now reading 2d coordinates for left and right camera, fiducial */
/* fr.txt files -- fiducial right text files*****************************/

if((fp=fopen(fr,"rt")) == NULL) {     /* rt r-read , t-text file */
   puts("cannot open file \n");
   exit();
   }

fscanf(fp," %03d ", &n2);

for(no=0;no<n2;no++) {
   fscanf(fp," %03d %f %f \n", &no5[no],&x5[no],&y5[no]);
   no5[no] = no5[no]-1;     /* Due to matrix numbering from 0 */
   }
fclose(fp);

xtr=0;
ytr=0;
for(no=0;no<n2;no++) {
   xtr=x5[no]+xtr;
   ytr=y5[no]+ytr;
   }
xtr=xtr/4;
ytr=ytr/4;
for(no=0;no<n2;no++) {
   cxr[no]=x5[no]-xtr;
   cyr[no]=y5[no]-ytr;
   }


/*************************************************************************/
/* now reading 2d coordinates for left and right object files       */
/* lobj.txt files --  object left text files*****************************/

if((ft=fopen(lobj,"rt")) == NULL) {     /* rt r-read , t-text file */
   puts("cannot open file \n");
   exit();
   }

fscanf(ft," %03d \n", &ex);

for(no=0;no<ex;no++) {
    fscanf(ft," %03d %f %f \n", &no1[no], &p2[no], &q2[no]);
    }
fclose(ft);

for(no=0;no<4;no++) {
   x4[no]=p2[no];
   y4[no]=q2[no];
   }

xt=0;
yt=0;
for(no=0;no<4;no++) {
   xt=x4[no]+xt;
   yt=y4[no]+yt;
   }
xt=xt/4;
```

```
aa=     aa=(xx+yy)/sum;
bb=(yx-xy)/sum;

for(no=0;no<ex;no++) {
   oa[no]=p2[no]-xt;
   ob[no]=q2[no]-yt;
   }
for(no=0;no<ex;no++) {
   p2[no]=aa*oa[no]+bb*ob[no]+xtl;
   q2[no]=aa*ob[no]-bb*oa[no]+ytl;
   }

for(no=0;no<ex;no++) {
   p2[no]= p2[no] * 0.015;     /*  1.059   for baby image,   0.015  for proton */
   q2[no]=-q2[no] * 0.011;     /*  0.749                     0.011            */
   }

/*************************************************************************/
/* now reading 2d coordinates for left and right object files          */
/* robj.txt files --  object right text files*****************************/

if((fp=fopen(robj,"rt")) == NULL) {     /* rt r-read , t-text file */
   puts("cannot open file \n");
   exit();
   }

fscanf(fp," %03d \n", &ex);

for(no=0;no<ex;no++) {
   fscanf(fp," %03d %f %f \n", &no2[no],&p[no],&q[no]);
   }
fclose(fp);

for(no=0;no<4;no++) {
   x5[no]=p[no];
   y5[no]=q[no];
   }

xt=0;
yt=0;
for(no=0;no<4;no++) {
   xt=x5[no]+xt;
   yt=y5[no]+yt;
   }
xt=xt/4;
yt=yt/4;
for(no=0;no<4;no++) {
   oa[no]=x5[no]-xt;
   ob[no]=y5[no]-yt;
   }
xx=0;
yy=0;
yx=0;
xy=0;
sum=0;
for(no=0;no<4;no++) {
   xx=oa[no]*cxr[no]+xx;
   yy=ob[no]*cyr[no]+yy;
```

```
   yx=cyr[no]*oa[no]+yx;
   xy=cxr[no]*ob[no]+xy;
   sum=oa[no]*oa[no]+ob[no]*ob[no]+sum;
   }
aa=(xx+yy)/sum;
bb=(yx-xy)/sum;

for(no=0;no<ex;no++) {
   oa[no]=p[no]-xt;
   ob[no]=q[no]-yt;
   }
for(no=0;no<ex;no++) {
   p[no]=aa*oa[no]+bb*ob[no]+xtr;
   q[no]=aa*ob[no]-bb*oa[no]+ytr;
   }

for(no=0;no<ex;no++) {
   p[no]= p[no] * 0.015;   /* 1.059   for baby image,   0.015  for proton */
   q[no]=-q[no] * 0.011;    /* 0.749                     0.011            */
   }


/*****************************************************************************/
/*  FOR PRINTOUT OF ACCURACIES ON THE PRINTER  */
for(ii=0;ii<ex;ii++) {
   a3[0][0] = p2[ii] * a1[8][0] - a1[0][0];
   a3[0][1] = p2[ii] * a1[9][0] - a1[1][0];
   a3[0][2] = p2[ii] * a1[10][0] - a1[2][0];
   13[0][0] = a1[3][0] - p2[ii];
   a3[1][0] = q2[ii] * a1[8][0] - a1[4][0];
   a3[1][1] = q2[ii] * a1[9][0] - a1[5][0];
   a3[1][2] = q2[ii] * a1[10][0] - a1[6][0];
   13[1][0] = a1[7][0] - q2[ii];
   a3[2][0] = p[ii] * a[8][0] - a[0][0];
   a3[2][1] = p[ii] * a[9][0] - a[1][0];
   a3[2][2] = p[ii] * a[10][0] - a[2][0];
   13[2][0] = a[3][0] - p[ii];
   a3[3][0] = q[ii] * a[8][0] - a[4][0];
   a3[3][1] = q[ii] * a[9][0] - a[5][0];
   a3[3][2] = q[ii] * a[10][0] - a[6][0];
   13[3][0] = a[7][0] - q[ii];


 /**** TRANSPOSE   MAT A3  = TRANS M3 ***************************/
 for(i=0;i<4;i++) {
    for(j1=0;j1<3;j1++) {
      a3[j1][i] = a3[i][j1];
      }
    }

 /**** MULT  MAT A4 = A3 * M3 ***********************************/
  for(i=0;i<3;i++) {
    for(j1=0;j1<3;j1++) {
      a4[i][j1] = 0;
      for(k=0;k<4;k++) {
         a4[i][j1]=a4[i][j1]+a3[i][k]*a3[k][j1];
         }
      }
```

```
    }


/********** INVERS  MAT a4 = INV a4 ********************************/
  for(i=0;i<3;i++) {
    for(jl=0;jl<3;jl++) {
      al[i][jl] = a4[i][jl];
      }
    }

  for(i=0;i<3;i++) {
    ql = 1/al[i][i];
    al[i][i] = ql;
    for(jl=0;jl<3;jl++) {
      if(jl != i) {
         al[i][jl] = al[i][jl] * ql;
         }
      }
      for(jl=0;jl<3;jl++) {
        if(jl != i) {
          ql = al[jl][i];
          al[jl][i] = 0;
          for(k=0;k<3;k++) {
             al[jl][k] = al[jl][k] - al[i][k] * ql;
             }
          }
        }
      }

  for(i=0;i<3;i++) {
    for(jl=0;jl<3;jl++) {
      a4[i][jl] = al[i][jl];
      }
    }



/**** MULT  MAT B2 = M4 * A3 ***********************************/
  for(i=0;i<3;i++) {
    for(jl=0;jl<4;jl++) {
      b2[i][jl] = 0;
      for(k=0;k<3;k++) {
         b2[i][jl]=b2[i][jl]+a4[i][k]*a3[k][jl];
         }
      }
    }


/**** MULT  MAT M5 = B2 * L3 ***********************************/
  for(i=0;i<3;i++) {
    for(jl=0;jl<1;jl++) {
      a5[i][jl] = 0;
      for(k=0;k<4;k++) {
         a5[i][jl]=a5[i][jl]+b2[i][k]*l3[k][jl];
         }
      }
    }
```

```
s1[ii] = a5[0][0];
t[ii] = a5[1][0];
v[ii] = a5[2][0];
}


mode();
printf("   TRANSFORMED OBJECT POINTS              \n");
printf("  ═══════════════════════════════════ \n");
printf("     X        Y       Z       (mm)       \n");
printf("  ═══════════════════════════════════ \n");
if(abcd == 89 || abcd == 121)  {
   fprintf(stdprn," \n");
   fprintf(stdprn,"  TRANSFORMED OBJECT POINTS                 \n");
   fprintf(stdprn,"------------------------------------------- \n");
   fprintf(stdprn,"  No.     X       Y       Z       (mm)    \n");
   fprintf(stdprn,"------------------------------------------- \n");
   fprintf(stdprn," \n");
   }
printf(" \n");

if((fp=fopen(d3,"w+")) == NULL) {     /* w+ -write , t-text file */
   puts("cannot open file \n");
   exit();
   }

tt=19;
fprintf(fp,"%4d \n",ex-4);
for(i=4;i<ex;i++) {
   ii=i-3;
   tac = ii;
   if( ii > 18 && fmod(tac,tt)==0 ) {
      printf(" \n");
      printf("  Press SPACE BAR to continue : ");
      ch = bdos(7,0,0);
      while( ch !=32 ) {
          ch = bdos(7,0,0);
          }
      printf(" \n");
      mode();
      printf("   TRANSFORMED OBJECT POINTS              \n");
      printf("  ═══════════════════════════════════ \n");
      printf("     X        Y       Z       (mm)       \n");
      printf("  ═══════════════════════════════════ \n");
      }
   printf(" %3d  %6.1f  %6.1f  %6.1f \n",ii,s1[i],t[i],v[i]);
   if(abcd == 89 || abcd == 121) {
      fprintf(stdprn," %3d  %6.1f  %6.1f  %6.1f \n",ii,s1[i],t[i],v[i]);
      }
   fprintf(fp," %03d %6.2f %6.2f %6.2f \n",ii,s1[i],t[i],v[i]);
   }
fclose(fp);



if(abcd==89 || abcd == 121)  {
   fprintf(stdprn," \n");
   fprintf(stdprn," \n");
   fprintf(stdprn," \n");
```

```
        }


    printf(" \n");
    printf(" \n");
    printf("  Press SPACE BAR to continue : ");
    ch = bdos(7,0,0);
    while( ch !=32 ) {
        ch = bdos(7,0,0);
        }
    printf(" \n");
    }                       /* END OF ELSE */

/* END OF MAIN */
}
```

APPENDIX 18 MAIN/VEC - TAG PROGRAM TO PRODUCE VECTOGRAM

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>


char buf[260];
char dff[1100];
int num1[30];
int count;

main()
{
  float p[50],q[50],p2[50],q2[50];
  float pp[50],qq[50],pp2[50],qq2[50];
  int no1[50],no2[50];
  int no,n2;
  int ex;
  float x4[4],y4[4],no4[4],xtl,ytl;
  float x5[4],y5[4],no5[4],xtr,ytr;
  float oa[50],ob[50],cxl[4],cyl[4],cxr[4],cyr[4];
  float xx,yy,yx,xy,sum,aa,bb,xt,yt;
  char nam[20],name[20];
  char nam1[20], nam2[20];
  char na[20];
  char s[20];
  char bth1[20];
  char bth2[20];
  char ch,str;
  char kill='t';
  char fl[15],fr[15];
  char lobj1[15],robj1[15],vec[15];
  char lobj2[15],robj2[15];
  int aaa,bbb;
  float mag;

  FILE *fp;
  FILE *ft;

 mode();
 printf("  \n");
 printf(" DISPLACEMENT VECTORS \n");
 printf(" ==================== \n");
 printf("  \n");
 printf("  \n");
 printf(" PATIENT'S NAME : ");
 scanf("%s",nam);                        /* nam = deb */
 printf("  \n");
 printf(" ENTER BREATH NUMBER OUT  i.e. 1o : ");
 scanf("%s",bth1);
 printf(" ENTER BREATH NUMBER  IN  i.e. 1i : ");
```

```
scanf("%s",bth2);                       /*    bth2 = 1i              */
printf("  \n");
strcpy(name,nam);                       /*    name = deb             */
strcpy(s,"\\pip\\i\\");                 /*    s =  \pip\i\           */
strcpy(str,"o");                        /*    str = o                */
strcat (name,str);                      /*    name = debo            */
strcat (name,bth1);                     /*    name = debo1o          */

/********** INITIALISING THE PIP BOARD ********************************/

fg_inifmt(620,1,0,0,1,0);    /* initialises board at base address */
fg_setind(255);              /* set drawing index */
fg_clear(0,7);               /* clear frame buffer to current index */
fg_sboard(0);
fg_sbuf(1);
fg_chan(1);
fg_sync(0);

/*************************************************************************/

strcat (s,name);
fg_frdisk(1024,0,s,dff,-1);     /* call up image */


/*** CREATING FILE EXTENSIONS ***************************/

    strcpy(nam1,nam);           /* nam1 = deb */
    strcpy(s,"4L.TXT");         /* s = 4l.txt */
    strcat(nam1,s);             /* nam1 = deb4l.txt */
    strcpy(fl,nam1);            /* cl = deb4l.txt */

    strcpy(nam1,nam);           /* nam1 = deb */
    strcpy(s,"4R.TXT");         /* s = 4r.txt */
    strcat(nam1,s);             /* nam1 = deb4r.txt */
    strcpy(fr,nam1);            /* cl = deb4r.txt */

    strcpy(nam2,nam);           /* nam2 = deb */
    strcat(nam2,"o");           /* nam2 = debo */
    strcat(nam2,bth1);           /* nam2 = debo1o */

    strcpy(nam1,nam2);          /* nam1=nam2=debo1o */
    strcpy(s,"L.OBJ");          /* s = l.obj */
    strcat(nam1,s);             /* nam1 = debo1ol.obj */
    strcpy(lobj1,nam1);          /* lobj = debo1ol.obj */

    strcpy(nam1,nam2);          /* nam1=nam2=debo1o */
    strcpy(s,"R.OBJ");          /* s = r.obj */
    strcat(nam1,s);             /* nam1 = debo1or.obj */
    strcpy(robj1,nam1);          /* robj = debo1or.obj */

    strcpy(nam2,nam);           /* nam2 = deb */
    strcat(nam2,"o");           /* nam2 = debo */
    strcat(nam2,bth2);           /* nam2 = debo1i */

    strcpy(nam1,nam2);          /* nam1=nam2=debo1i */
    strcpy(s,"L.OBJ");          /* s = l.obj */
    strcat(nam1,s);             /* nam1 = debo1il.obj */
```

```
strcpy(lobj2,nam1);          /* lobj2 = debolil.obj */

strcpy(nam1,nam2);          /* nam1=nam2=deboli */
strcpy(s,"R.OBJ");          /* s = r.obj */
strcat(nam1,s);             /* nam1 = debolir.obj */
strcpy(robj2,nam1);          /* robj2 = debolir.obj */

strcpy(nam1,nam);           /* nam1=deb */
strcpy(s,"v");              /* s = v */
strcat(nam1,s);             /* nam1 = debv */
strcat(nam1,bth1);           /* nam1 = debvlo */
strcpy(vec,nam1);           /* vec = debvlo */

/*************************************************************************/
/* now reading 2d coordinates for left and right camera, fiducial */
/* fl.txt files -- fiducial left text files*****************************/

if((fp=fopen(fl,"rt")) == NULL) {     /* rt r-read , t-text file */
   puts("cannot open file \n");
   exit();
   }

fscanf(fp," %03d ", &n2);

for(no=0;no<n2;no++) {
   fscanf(fp," %03d %f %f \n", &no4[no],&x4[no],&y4[no]);
   no4[no] = no4[no]-1;     /* Due to matrix numbering from 0 */
   }
fclose(fp);

xtl=0;
ytl=0;
for(no=0;no<n2;no++) {
   xtl=x4[no]+xtl;
   ytl=y4[no]+ytl;
   }
xtl=xtl/4;
ytl=ytl/4;
for(no=0;no<n2;no++) {
   cxl[no]=x4[no]-xtl;
   cyl[no]=y4[no]-ytl;
   }


/*************************************************************************/
/* now reading 2d coordinates for left and right camera, fiducial */
/* fr.txt files -- fiducial right text files*****************************/

if((fp=fopen(fr,"rt")) == NULL) {     /* rt r-read , t-text file */
   puts("cannot open file \n");
   exit();
   }

fscanf(fp," %03d ", &n2);

for(no=0;no<n2;no++) {
   fscanf(fp," %03d %f %f \n", &no5[no],&x5[no],&y5[no]);
   no5[no] = no5[no]-1;     /* Due to matrix numbering from 0 */
   }
```

```
fclose(fp);

xtr=0;
ytr=0;
for(no=0;no<n2;no++) {
   xtr=x5[no]+xtr;
   ytr=y5[no]+ytr;
   }
xtr=xtr/4;
ytr=ytr/4;
for(no=0;no<n2;no++) {
   cxr[no]=x5[no]-xtr;
   cyr[no]=y5[no]-ytr;
   }


/**** FIRST FRAME - BREATH OUT ***********************************************/
/*****************************************************************************/
/* now reading 2d coordinates for left and right object files       */
/* lobj1.txt files --  object left text files*************************/

if((ft=fopen(lobj1,"rt")) == NULL) {     /* rt r-read , t-text file */
   puts("cannot open file \n");
   exit();
   }

fscanf(ft," %03d \n", &ex);

for(no=0;no<ex;no++) {
    fscanf(ft," %03d %f %f \n", &no1[no], &p2[no], &q2[no]);
    }
fclose(ft);

for(no=0;no<4;no++) {
   x4[no]=p2[no];
   y4[no]=q2[no];
   }

xt=0;
yt=0;
for(no=0;no<4;no++) {
   xt=x4[no]+xt;
   yt=y4[no]+yt;
   }
xt=xt/4;
yt=yt/4;
for(no=0;no<4;no++) {
   oa[no]=x4[no]-xt;
   ob[no]=y4[no]-yt;
   }
xx=0;
yy=0;
yx=0;
xy=0;
```

```
sum=0;
for(no=0;no<4;no++) {
   xx=oa[no]*cx1[no]+xx;
   yy=ob[no]*cy1[no]+yy;
   yx=cy1[no]*oa[no]+yx;
   xy=cx1[no]*ob[no]+xy;
   sum=oa[no]*oa[no]+ob[no]*ob[no]+sum;
   }
aa=(xx+yy)/sum;
bb=(yx-xy)/sum;

for(no=0;no<ex;no++) {
   oa[no]=p2[no]-xt;
   ob[no]=q2[no]-yt;
   }
for(no=0;no<ex;no++) {
   p2[no]=aa*oa[no]+bb*ob[no]+xt1;
   q2[no]=aa*ob[no]-bb*oa[no]+yt1;
   }


/*************************************************************************/
/* now reading 2d coordinates for left and right object files          */
/* robj1.txt files --  object right text files*************************/

if((fp=fopen(robj1,"rt")) == NULL) {     /* rt r-read , t-text file */
   puts("cannot open file \n");
   exit();
   }

fscanf(fp," %03d \n", &ex);

for(no=0;no<ex;no++) {
   fscanf(fp," %03d %f %f \n", &no2[no],&p[no],&q[no]);
   }
fclose(fp);

for(no=0;no<4;no++) {
   x5[no]=p[no];
   y5[no]=q[no];
   }

xt=0;
yt=0;
for(no=0;no<4;no++) {
   xt=x5[no]+xt;
   yt=y5[no]+yt;
   }
xt=xt/4;
yt=yt/4;
for(no=0;no<4;no++) {
   oa[no]=x5[no]-xt;
   ob[no]=y5[no]-yt;
   }
```

```
xx=0;
yy=0;
yx=0;
xy=0;
sum=0;
for(no=0;no<4;no++) {
   xx=oa[no]*cxr[no]+xx;
   yy=ob[no]*cyr[no]+yy;
   yx=cyr[no]*oa[no]+yx;
   xy=cxr[no]*ob[no]+xy;
   sum=oa[no]*oa[no]+ob[no]*ob[no]+sum;
   }
aa=(xx+yy)/sum;
bb=(yx-xy)/sum;

for(no=0;no<ex;no++) {
   oa[no]=p[no]-xt;
   ob[no]=q[no]-yt;
   }
for(no=0;no<ex;no++) {
   p[no]=aa*oa[no]+bb*ob[no]+xtr;
   q[no]=aa*ob[no]-bb*oa[no]+ytr;
   }



/**** SECOND FRAME - BREATH IN ********************************************/
/***************************************************************************/
/* now reading 2d coordinates for left and right object files       */
/* lobj2.txt files --  object left text files*************************/

if((ft=fopen(lobj2,"rt")) == NULL) {     /* rt r-read , t-text file */
   puts("cannot open file \n");
   exit();
   }

fscanf(ft," %03d \n", &ex);

for(no=0;no<ex;no++) {
    fscanf(ft," %03d %f %f \n", &no1[no], &pp2[no], &qq2[no]);
    }
fclose(ft);

for(no=0;no<4;no++) {
   x4[no]=pp2[no];
   y4[no]=qq2[no];
   }

xt=0;
yt=0;
for(no=0;no<4;no++) {
   xt=x4[no]+xt;
   yt=y4[no]+yt;
   }
xt=xt/4;
yt=yt/4;
for(no=0;no<4;no++) {
   oa[no]=x4[no]-xt;
```

```
   ob[no]=y4[no]-yt;
   }
xx=0;
yy=0;
yx=0;
xy=0;
sum=0;
for(no=0;no<4;no++) {
   xx=oa[no]*cx1[no]+xx;
   yy=ob[no]*cy1[no]+yy;
   yx=cy1[no]*oa[no]+yx;
   xy=cx1[no]*ob[no]+xy;
   sum=oa[no]*oa[no]+ob[no]*ob[no]+sum;
   }
aa=(xx+yy)/sum;
bb=(yx-xy)/sum;

for(no=0;no<ex;no++) {
   oa[no]=pp2[no]-xt;
   ob[no]=qq2[no]-yt;
   }
for(no=0;no<ex;no++) {
   pp2[no]=aa*oa[no]+bb*ob[no]+xt1;
   qq2[no]=aa*ob[no]-bb*oa[no]+yt1;
   }


/*************************************************************************/
/* now reading 2d coordinates for left and right object files          */
/* robj2.txt files --  object right text files*************************/

if((fp=fopen(robj2,"rt")) == NULL) {     /* rt r-read , t-text file */
   puts("cannot open file \n");
   exit();
   }

fscanf(fp," %03d \n", &ex);

for(no=0;no<ex;no++) {
   fscanf(fp," %03d %f %f \n", &no2[no],&pp[no],&qq[no]);
   }
fclose(fp);

for(no=0;no<4;no++) {
   x5[no]=pp[no];
   y5[no]=qq[no];
   }

xt=0;
yt=0;
for(no=0;no<4;no++) {
   xt=x5[no]+xt;
   yt=y5[no]+yt;
   }
xt=xt/4;
yt=yt/4;
for(no=0;no<4;no++) {
   oa[no]=x5[no]-xt;
```

```
        ob[no]=y3[no]-yt;
        }
     xx=0;
     yy=0;
     yx=0;
     xy=0;
     sua=0;
     for(no=0;no<4;no++) {
        xx=oa[no]*cxr[no]+xx;
        yy=ob[no]*cyr[no]+yy;
        yx=cyr[no]*oa[no]+yx;
        xy=cxr[no]*ob[no]+xy;
        sua=oa[no]*oa[no]+ob[no]*ob[no]+sua;
        }
     aa=(xx+yy)/sua;
     bb=(yx-xy)/sua;

     for(no=0;no<ex;no++) {
        oa[no]=pp[no]-xt;
        ob[no]=qq[no]-yt;
        }
     for(no=0;no<ex;no++) {
        pp[no]=aa*oa[no]+bb*ob[no]+xtr;
        qq[no]=aa*ob[no]-bb*oa[no]+ytr;
        }


  /* frame lo:  left-p2,q2    right-p,q */
  /* frame li:  left-pp2,qq2  right-pp,qq */
  /* vectors on left frame  - p2,q2 to pp2,qq2 */
  /* vectors on right frame - p,q to pp,qq */

     mag=1.0;
     for(no=4;no<ex;no++) {
        aaa=p2[no];
        bbb=q2[no];
        fg_moveto(aaa,bbb);
        aaa=pp2[no]+(pp2[no]-p2[no])*(mag-1);
        bbb=qq2[no]+(qq2[no]-q2[no])*(mag-1);
        fg_lineto(aaa,bbb);
        aaa=p[no];
        bbb=q[no];
        fg_moveto(aaa,bbb);
        aaa=pp[no]+(pp[no]-p[no])*(mag-1);
        bbb=qq[no]+(qq[no]-q[no])*(mag-1);
        fg_lineto(aaa,bbb);
        }


while(kill != 'q') {
     mode();
     printf("  \n");
     printf(" DISPLACEMENT VECTORS \n");
     printf(" ================== \n");
     printf("  \n");
     printf("  \n");
     printf(" Do you wish to see the vectors at a different magnification y/n ");
     ch=getch();
```

```
    while(ch != 'y' && ch != 'Y' && ch != 'n' && ch != 'N') {
        getch();
        }
    printf(" \n");
    if(ch == 89 || ch == 121) {
       strcpy(s,"\\pip\\i\\");              /* s = \pip\i\       */
       strcat (s,name);
       printf(" Refreshing video image !!! \n\n");
       fg_frdisk(1024,0,s,dff,-1);     /* call up image */
       printf(" New magnification : ");
       scanf("%f",&mag);
       printf(" \n");
       for(no=4;no<ex;no++) {
          aaa=p2[no];
          bbb=q2[no];
          fg_moveto(aaa,bbb);
          aaa=pp2[no]+(pp2[no]-p2[no])*(mag-1);
          bbb=qq2[no]+(qq2[no]-q2[no])*(mag-1);
          fg_lineto(aaa,bbb);
          aaa=p[no];
          bbb=q[no];
          fg_moveto(aaa,bbb);
          aaa=pp[no]+(pp[no]-p[no])*(mag-1);
          bbb=qq[no]+(qq[no]-q[no])*(mag-1);
          fg_lineto(aaa,bbb);
          }
     }
    else {
       kill='q';
       }
     printf(" \n");
     printf(" Do you wish to save this particular image y/n ");
     ch=getch();
     while(ch != 'y' && ch != 'Y' && ch != 'n' && ch != 'N') {
         getch();
         }
     printf(" \n");
     if(ch == 89 || ch == 121) {
        strcpy(s,"\\pip\\i\\");              /* s = \pip\i\       */
        strcat(s,vec);                       /* s = \pip\i\debvio */
        fg_todisk(1024,0,s,dff,-1);          /* call up image */
        }
     }

    printf(" \n");
    printf(" \n");
    printf(" Press SPACE BAR to continue : ");
    ch = getch();
    while( ch !=32 ) {
        ch = getch();
        }
    printf(" \n");

}
/******* END OF MAIN ******************************/
```

APPENDIX 19 PLT.TRU - TAG PROGRAM TO PRODUCE XY AND Z DISPLACEMENT VECTORS AND STEREOGRAMS

```
OPTION ANGLE degrees
LIBRARY "\barb\dummy"
LIBRARY "\reto\graphlib"
LIBRARY "\reto\shell"
DIM y(1)
DIM x11(1),y11(1),z11(1),zdiff(1)
DIM x22(1),y22(1),z22(1)
DIM x1(1),y1(1),z1(1)
DIM x2(1),y2(1),z2(1)
DIM xcc(1),ycc(1),dcc(1)
DIM x3(1),y3(1),z3(1),x4(1),y4(1)
DIM xx3(1),yy3(1),xx4(1),yy4(1)
DIM xc(1),yc(1),dc(1),nu(1)
DIM xl1(1),yl1(1),xr1(1),yr1(1),xl2(1),yl2(1),xr2(1),yr2(1)
DIM xcl(1),ycl(1),xcr(1),ycr(1)
DIM xl(1),yl(1),xr(1),yr(1)
LET c= -5.5                    ! constant for plotter
LET e = .95                    ! factor for x on the ibm screen
LET r = 1.33                   ! factor for y on the ibm screen
LET fg = .8                    ! factor for x on the hp plotter
LET g = 1.1                    ! factor for y on the hp plotter

CLEAR
PRINT
PRINT
PRINT
PRINT
PRINT
PRINT
PRINT *
PRINT *
PRINT *       THIS PROGRAM PLOTS THE DISPLACEMENT VECTORS IN THE X-Y PLANE
PRINT *       AND THE 3D VECTORS USING THE 'BREATH-IN AND BREATH-OUT DATA
PRINT *             BOTH ON THE SCREEN AND ON THE HP PLOTTER
PRINT *
PRINT *
PRINT
PRINT
PRINT
PRINT
PRINT
PRINT
PRINT *    ENTER ANY KEY TO CONTINUE : ";
GET KEY TRASH
CLEAR
PRINT
PRINT
PRINT
PRINT
PRINT
PRINT
PRINT
INPUT prompt "ENTER THE NAME OF THE PATIENT : ": name$
PRINT
PRINT
PRINT
INPUT prompt "ENTER BREATH NUMBER  ( e.g 1) : ": tb$
PRINT
PRINT
LET ok=0
```

```
PRINT
PRINT "DO YOU REQUIRE THE OUTLINE OF THE PATIENT ? ENTER Y/N : ";
DO
   GET KEY ans
   IF ans= 121 or ans=89 or ans=110 or ans=78 then
      LET ok=1
   END IF
LOOP until ok=1
PRINT
PRINT
PRINT
PRINT
PRINT
PRINT
PRINT
PRINT
PRINT

LET file_a$="c:\barb\tag\" & name$ & "o" & tb$ & "i" & "3d"
LET file_b$="c:\barb\tag\" & name$ & "o" & tb$ & "o" & "3d"

LET file_a$= file_a$ & ".txt"
LET file_b$= file_b$ & ".txt"

OPEN #1 :name  file_a$ , access input, organization text   !breath-in
SET #1 : POINTER begin
INPUT #1 : a$
LET n=val(a$[1:3])
MAT redim x22(n),y22(n),z22(n),nu(n)
FOR I = 1 to n
    INPUT #1:a$
    LET nu(i)=val(a$[1:3])
    LET x22(i)=val(a$[5:10])
    LET y22(i)=val(a$[12:17])
    LET z22(i)=val(a$[19:24])
NEXT I

OPEN #2 :name  file_b$ , access input, organization text   !breath-out
SET #2 : POINTER begin
INPUT #2 : a$
LET n=val(a$[1:3])
MAT redim x11(n),y11(n),z11(n),zdiff(n),nu(n)
FOR I = 1 to n
    INPUT #2:a$
    LET nu(i)=val(a$[1:3])
    LET x11(i)=val(a$[5:10])
    LET y11(i)=val(a$[12:17])
    LET z11(i)=val(a$[19:24])
NEXT I
CLOSE #1
CLOSE #2
CLEAR
IF ans= 121 or ans=89 then        ! ASCII CODES 121=y   89=Y
   LET file_c$= name$ & "out1.txt"
   OPEN #1 : name file_c$ , access input, organization text
   SET #1 : POINTER begin
   INPUT #1 : a$
   LET np=val(a$[1:3])
   MAT redim xcc(np),ycc(np),dcc(np),nu(np)
```

```
   FOR i=1 to np
      INPUT #1:a$
      LET nu(i)=val(a$[1:3])
      LET xcc(i)=val(a$[6:10])
      LET ycc(i)=val(a$[13:17])
      LET dcc(i)=1
   NEXT i
END IF
CLOSE #1
MAT redim x1(n),y1(n),z1(n),zdiff(n)
MAT redim x2(n),y2(n),z2(n),y(n)
MAT redim x3(n),y3(n),z3(n),x4(n),y4(n)
MAT redim xl(n),yl(n),xr(n),yr(n)
MAT redim xc(np),yc(np)
MAT redim xl1(n),yl1(n),xr1(n),yr1(n),xl2(n),yl2(n),xr2(n),yr2(n)
MAT redim xc(np),yc(np),dc(np)


!           ##############################################
!           this section does the calculating of the
!           x , y and z displacement vectors using the
!           the original breath in and out input data
!           ##############################################

LET mag=5
LET zmag=mag

FOR i=1 to n
   LET x2(i)= x22(i)
   LET y2(i)= y22(i)
   LET z2(i)= z22(i)
NEXT i

FOR i=1 to n
   LET x1(i)= x11(i)
   LET y1(i)= y11(i)
   LET z1(i)= z11(i)
NEXT i
FOR i=1 to n
   LET x2(i)= x1(i)+zmag*(x2(i)-x1(i))
   LET y2(i)= y1(i)+zmag*(y2(i)-y1(i))
   LET zdiff(i)=zmag*(z2(i)-z1(i))
NEXT i

FOR i=1 to n
   LET xl(i)=x1(i)
   LET yl(i)=y1(i)
   LET xr(i)=x1(i)
   LET yr(i)=y1(i)
NEXT i

FOR i=1 to np
   LET yc(i)= ycc(i)
   LET xc(i)= xcc(i)
NEXT i

LET xlmin=xl(1)
LET xlmax=xl(1)
LET ylmin=yl(1)
```

```
LET ylmax=yl(1)
LET xrmin=xr(1)
LET xrmax=xr(1)
LET yrmin=yr(1)
LET yrmax=yr(1)

CALL minmax

CLEAR
PRINT
PRINT
PRINT
PRINT
PRINT  "  AFTER EACH PLOT PRESS ANY KEY TO CONTINUE        "
PRINT  "        WITH THE FOLLOWING PICTURE                 "
PRINT
PRINT
PRINT
PRINT "  ENTER ANY KEY TO CONTINUE : ";
GET KEY TRASH
PRINT
PRINT
PRINT
CLEAR
PRINT
PRINT
LET ok=0
PRINT
PRINT
PRINT " DO YOU REQUIRE TO VIEW THE DISPLACEMENT VECTORS ? Y/N : ";
DO
   GET KEY xyv
   IF xyv= 121 or xyv=89 or xyv=110 or xyv=78 then
      LET ok=1
   END IF
LOOP until ok=1
PRINT
PRINT
PRINT
IF xyv=89 OR xyv=121 THEN
   LET dxl =(e$xlmax)-(e$xlmin)
   LET dyl =(r$ylmax)-(r$ylmin)
   LET wxlmin=e$xlmin-.8$dxl
   LET wxlmax=e$xlmax+.8$dxl
   LET wylmin=r$ylmin-.15$dyl
   LET wylmax=r$ylmax+.15$dyl

   SET MODE "hires"
   OPEN #1: screen 0, 1, 0, 1
   CLEAR
   SET WINDOW wxlmin,wxlmax, wylmin,wylmax
   CALL frame
   PLOT TEXT ,AT wxlmin+12,wylmin+5:"X Y DISPLACEMENT VECTORS"
   DRAW vecxy
```

```
   GET KEY trash
   CLEAR

   SET WINDOW wxlmin,wxlmax,wylmin,wylmax
   CALL frame
   PLOT TEXT ,AT wxlmin+12,wylmin+5:"Z DISPLACEMENT VECTORS"
   DRAW vecz
   GET KEY trash
   CLEAR
   CLOSE #1

   LET ok=0
   SET MODE "mono"
   PRINT "DO YOU WANT TO PLOT ??  ENTER Y/N : ";
   DO
      GET KEY pl1
      IF pl1= 121 or pl1=89 or pl1=110 or pl1=78 then
         LET ok=1
      END IF
   LOOP until ok=1

   IF pl1=121 OR pl1=89 then
      CALL dos ("redirect lpt1 705")   ! redirects the printer to the hp plotter)
      OPEN #2: printer
      CALL plotxy
      CLEAR
      PRINT
      PRINT
      PRINT
      PRINT "     CHANGE THE PLOTTER PAPER   !   !!!!!!     "
      SOUND 600,2
      PAUSE 10
      CALL plotz
   END IF
ELSE IF xyv=78 OR xyv=110 THEN
   LET pl1=78
END IF
CLOSE #2

MAT redim xc(np),yc(np),dc(np)
MAT redim x1(n),y1(n),z1(n),zdiff(n)
MAT redim x2(n),y2(n),z2(n)
MAT redim x3(n),y3(n),z3(n),x4(n),y4(n)
MAT redim xx3(n),yy3(n),xx4(n),yy4(n)

FOR i=1 to n
    LET x2(i)=x22(i)+100
    LET y2(i)=y22(i)
    LET z2(i)=z22(i)
NEXT i

FOR i= 1 to n
    LET x1(i)=x11(i)+100
    LET y1(i)=y11(i)
    LET z1(i)=z11(i)
    LET x2(i)=x1(i)+mag$(x2(i)-x1(i))
    LET y2(i)=y1(i)+mag$(y2(i)-y1(i))
    LET z2(i)=z1(i)+mag$(z2(i)-z1(i))
NEXT i
```

```
FOR i=1 to np
    LET xc(i)=xcc(i)+100
    LET yc(i)=ycc(i)
NEXT i
LET zc=z1(1)

! deriving pseudo stereo photography imagery
! adopt constants of B=200mm , F=50mm, H=1000mm

LET B=200
LET F=50
LET H=1000

MAT redim xl1(n),yl1(n),xr1(n),yr1(n),xl2(n),yl2(n),xr2(n),yr2(n)
FOR i=1 to n
    LET xl1(i)=x1(i)*F/(H-z1(i))
    LET yl1(i)=y1(i)*F/(H-z1(i))
    LET xr1(i)=xl1(i)-F*B/(H-z1(i))
    LET yr1(i)=yl1(i)
    LET xl2(i)=x2(i)*F/(H-z2(i))
    LET yl2(i)=y2(i)*F/(H-z2(i))
    LET xr2(i)=xl2(i)-F*B/(H-z2(i))
    LET yr2(i)=yl2(i)
NEXT i
MAT redim xcl(np),ycl(np),xcr(np),ycr(np)
FOR i=1 to np
    LET xcl(i)=xc(i)*F/(H-zc)
    LET ycl(i)=yc(i)*F/(H-zc)
    LET xcr(i)=xcl(i)-F*B/(H-zc)
    LET ycr(i)=ycl(i)
NEXT i

FOR i=1 to n
    LET xl(i)=xl1(i)
    LET yl(i)=yl1(i)
    LET xr(i)=xr1(i)
    LET yr(i)=yr1(i)
NEXT i

LET xlmin=xl(1)
LET xlmax=xl(1)
LET ylmin=yl(1)
LET ylmax=yl(1)
LET xrmin=xr(1)
LET xrmax=xr(1)
LET yrmin=yr(1)
LET yrmax=yr(1)

CALL minmax


!       choice of left or right picture
!       ***********************************
CLEAR
PRINT
PRINT
PRINT
PRINT
```

```
LET ok=0
PRINT
PRINT
PRINT "DO YOU REQUIRE TO VIEW THE STEREO VECTORS ? Y/N : ";
DO
   GET KEY vec
   IF vec= 121 or vec=89 or vec=110 or vec=78 then
      LET ok=1
   END IF
LOOP until ok=1
IF vec= 121 or vec=89 then
   PRINT
   PRINT
   PRINT
   LET ok=0
   PRINT " IS THIS A LEFT OR A RIGHT PICTURE ?  ENTER L/R : ";
   DO
      GET KEY pic
      IF pic=108 or pic=76 or pic=114 or pic=82 then
         LET ok=1
      END IF
   LOOP until ok=1

   IF  pic=76 OR pic=108 then
      CALL left
      CALL right
   ELSE IF pic=114 OR pic=82  then
      CALL right
   END IF
   PRINT
   CLOSE #1
   CLOSE #2
   CLEAR
   PRINT
   PRINT
   PRINT
   PRINT
   SET MODE "mono"
   LET ok=0
   PRINT
   PRINT
   PRINT " DO YOU WANT TO PLOT ??  ENTER Y/N : ";
   DO
      GET KEY pl2
      IF pl2=121 OR pl2=89 OR pl2=110 OR pl2=78 then
         LET ok=1
      END IF
   LOOP until ok=1

   IF pl2=78 or pl2=110 then
      PRINT
      PRINT
      PRINT "  ENTER ANY KEY TO CONTINUE : ";
      GET KEY TRASH
   ELSE IF  pl1=78 or pl1=110 AND pl2=89 or pl2=121 then
      LET pl1=89
      CALL dos ("redirect lpt1 705")
   END IF
```

```
   IF  pl1=89 or pl1=121 and pl2=89 or pl2=121 then
      OPEN #2: printer
      CALL plotl
      CLEAR
      PRINT
      PRINT
      PRINT
      PRINT *     CHANGE THE PLOTTER PAPER   !   !!!!!!     *
      SOUND 600,2
      CLOSE #2
      PAUSE 10
      OPEN #2: printer
      CALL plotr
      CLOSE #2
   END IF
END IF
CLEAR
SET MODE "mono"
PRINT
PRINT
LET ok=0
PRINT
PRINT
PRINT
PRINT * DO YOU REQUIRE AN ANAGLYPH  PICTURE ?  ENTER Y/N   *
PRINT
PRINT "( this is only for colour screens  !      !!) : ";
DO
   GET KEY ana
   IF ana=121 or ana=89 OR ana=110 or ana=78 then
      LET ok=1
   END IF
LOOP until ok=1

IF vec=110 or vec=78 and ana=89 or ana=121 then
   CALL left
ELSE IF ana=89 or ana=121 then
   DRAW anaglyph
END IF
CALL dos ("redirect lpt1")
SET MODE "mono"
CLEAR
PRINT
PRINT
PRINT
PRINT
PRINT *                                                    *
PRINT *                                                    *
PRINT *                      THE END !                     *
PRINT *                                                    *
PRINT *                                                    *

!************************* screen display routines ********************

!          ### draws x,y vector arrows ###
SUB arrxy
```

```
    LET d1=sqr((x2(i)-x1(i))^2+(y2(i)-y1(i))^2)
    LET d=d1/20
    LET alpha=atn((y2(i)-y1(i))/(x2(i)-x1(i)+0.0000001))
    LET dx=d*sin(alpha)
    LET dy=d*cos(alpha)
    LET xo=x1(i)+.8*(x2(i)-x1(i))
    LET yo=y1(i)+.8*(y2(i)-y1(i))
    LET x3(i)=xo-dx
    LET y3(i)=yo+dy
    LET x4(i)=xo+dx
    LET y4(i)=yo-dy
END SUB

PICTURE vecxy
    FOR i=1 to n
        PLOT e*x1(i),r*y1(i);
        PLOT e*x2(i),r*y2(i)
        CALL arrxy
        PLOT e*x2(i),r*y2(i);e*x3(i),r*y3(i)
        PLOT e*x2(i),r*y2(i);e*x4(i),r*y4(i)
    NEXT i

    !  * plots outline *
    IF ans= 121 or ans=89 then
       FOR i=2 to np
          IF dc(i-1)=1 then
             PLOT e*xc(i),r*yc(i);
          ELSE IF dc(i-1)=2 then
             PLOT e*xc(i),r*yc(i)
          END IF
       NEXT i
    END IF
END PICTURE

!                    **** draws arrows for z displacement ****
SUB arrz

    LET a=zdiff(i)/5
    LET dx=a*tan(30)
    LET x3(i)=x1(i)-dx
    LET y3(i)=y1(i)+zdiff(i)*3/5
    LET x4(i)=x1(i)+dx
    LET y4(i)=y1(i)+zdiff(i)*3/5
END SUB

PICTURE vecz
    FOR i=1 to n
        LET y(i)=y1(i)+zdiff(i)
        PLOT e*x1(i),r*y1(i);
        PLOT e*x1(i),r*y(i)
        CALL arrz
        PLOT e*x1(i),r*y(i);e*x3(i),r*y3(i)
        PLOT e*x1(i),r*y(i);e*x4(i),r*y4(i)
    NEXT i

    !  * plots outline *
    IF ans= 121 or ans=89 then
       FOR i=2 to np
```

```
        IF dc(i-1)=8 then
           PLOT e$xc(i),r$yc(i);
        ELSE IF dc(i-1)<>8 then
           PLOT e$xc(i),r$yc(i);
        END IF
     NEXT i
  END IF
END PICTURE


!              $$$$ pseudo stereos $$$$
SUB left

  LET dxl =(e$xlmax)-(e$xlmin)
  LET dyl =(r$ylmax)-(r$ylmin)
  LET wxlmin=e$xlmin-.8$dxl
  LET wxlmax=e$xlmax+.8$dxl
  LET wylmin=r$ylmin-.15$dyl
  LET wylmax=r$ylmax+.15$dyl

  IF ana=121 or ana=89 and vec=78 or vec=110 then
     CALL right
  ELSE IF  pic=76 OR pic=108 then
     DRAW lefthand
     GET KEY trash
  END IF
END SUB

SUB right

  LET dxr =(e$xrmax)-(e$xrmin)
  LET dyr =(r$yrmax)-(r$yrmin)
  LET wxrmin=e$xrmin-.8$dxr
  LET wxrmax=e$xrmax+.8$dxr
  LET wyrmin=r$yrmin-.15$dyr
  LET wyrmax=r$yrmax+.15$dyr

  IF ana=121 or ana=89 and vec=78 or vec=110 then
     DRAW anaglyph
  ELSE IF ana=0 then
     CLEAR
     DRAW righthand
     GET KEY trash
  ELSE IF ana=78 or ana=110 then
     CLEAR
     GET KEY trash
  END IF
  CLOSE #1
END SUB


!              $$$$ draws arrows hopefully $$$$
SUB arrowl

  LET d1=sqr((x12(i)-x11(i))^2+(y12(i)-y11(i))^2)
  LET d=d1/20
  LET alpha=atn((y12(i)-y11(i))/(x12(i)-x11(i)))
  LET dx=d$sin(alpha)
```

```
    LET dy=d*cos(alpha)
    LET xo=xl1(i)+.8*(xl2(i)-xl1(i))
    LET yo=yl1(i)+.8*(yl2(i)-yl1(i))
    LET x3(i)=xo-dx
    LET y3(i)=yo+dy
    LET x4(i)=xo+dx
    LET y4(i)=yo-dy
END SUB

SUB arrowr

    LET d1=sqr((xr2(i)-xr1(i))^2+(yr2(i)-yr1(i))^2)
    LET d=d1/20
    LET alpha=atn((yr2(i)-yr1(i))/(xr2(i)-xr1(i)))
    LET dx=d*sin(alpha)
    LET dy=d*cos(alpha)
    LET xo=xr1(i)+.8*(xr2(i)-xr1(i))
    LET yo=yr1(i)+.8*(yr2(i)-yr1(i))
    LET xx3(i)=xo-dx
    LET yy3(i)=yo+dy
    LET xx4(i)=xo+dx
    LET yy4(i)=yo-dy
END SUB

PICTURE lefthand
    SET MODE "hires"
    OPEN #1: screen 0, 1, 0, 1
    SET WINDOW  wxlmin, wxlmax, wylmin, wylmax
    BOX LINES wxlmin,wxlmax,wylmin,wylmax
    PLOT TEXT ,AT wxlmin+.6,wylmin+.5:"LEFT HAND PSEUDO VECTORS"
    FOR i=1 to n
        PLOT e*xl1(i),r*yl1(i);
        PLOT e*xl2(i),r*yl2(i)
        CALL arrowl
        PLOT e*xl2(i),r*yl2(i);e*x3(i),r*y3(i)
        PLOT e*xl2(i),r*yl2(i);e*x4(i),r*y4(i)
    NEXT i

    !   * plots outline *
    IF ans= 121 or ans=89 then
       FOR i=2 to np
           IF dc(i-1)=8 then
              PLOT e*xcl(i),r*ycl(i);
           ELSE IF dc(i-1)<>8 then
              PLOT e*xcl(i),r*ycl(i);
           END IF
       NEXT i
    END IF
    CLOSE #1
END PICTURE

PICTURE righthand
    SET MODE "hires"
    OPEN #1: screen 0, 1, 0, 1
    SET WINDOW  wxrmin, wxrmax, wyrmin, wyrmax
    PLOT TEXT ,AT wxrmin+.6,wyrmin+.5:"RIGHT HAND PSEUDO VECTORS"
    CALL frame
    FOR i=1 to n
        PLOT e*xr1(i),r*yr1(i);
```

```
        PLOT e$xr2(i),r$yr2(i)
        CALL arrowr
        PLOT e$xr2(i),r$yr2(i);e$xx3(i),r$yy3(i)
        PLOT e$xr2(i),r$yr2(i);e$xx4(i),r$yy4(i)
    NEXT i
    !   $ plots the outline $
    IF ans= 121 or ans=89 then
       FOR i=2 to np
          IF dc(i-1)=0 then
             PLOT e$xcr(i),r$ycr(i);
          ELSE IF dc(i-1)<>0 then
             PLOT  e$xcr(i),r$ycr(i);
          END IF
       NEXT i
       CLOSE #1
    END IF
END PICTURE

PICTURE anaglyph
    CLEAR
    OPEN #1: screen 0, 1, 0, 1
    SET MODE "graphics"
    SET WINDOW  wxlmin, wxlmax, wylmin, wylmax
    PLOT TEXT ,AT wxlmin+.6,wylmin+.7:"ANAGLYPH VECTORS"
    SET COLOR "red"
    FOR i=1 to n
        PLOT e$xl1(i),r$yl1(i);
        PLOT e$xl2(i),r$yl2(i)
        CALL arrowl
        PLOT e$xl2(i),r$yl2(i);e$x3(i),r$y3(i)
        PLOT e$xl2(i),r$yl2(i);e$x4(i),r$y4(i)
    NEXT i
    !   $ plots outline $
    IF ans= 121 or ans=89 then
       FOR i=2 to np
          IF dc(i-1)=0 then      ! flag of the cursor
             PLOT e$xcl(i),r$ycl(i);  ! this lifts the pen
          ELSE IF dc(i-1)<>0 then
             PLOT e$xcl(i),r$ycl(i);
          END IF
       NEXT i
    END IF
    SET COLOR "green"
    FOR i=1 to n
        SET WINDOW  wxrmin, wxrmax, wyrmin, wyrmax
        CALL frame
        PLOT e$xr1(i),r$yr1(i);
        PLOT e$xr2(i),r$yr2(i)
        CALL arrowr
        PLOT e$xr2(i),r$yr2(i);e$xx3(i),r$yy3(i)
        PLOT e$xr2(i),r$yr2(i);e$xx4(i),r$yy4(i)
    NEXT i
    !   $ plots the outline $
    IF ans= 121 or ans=89 then    ! ASCII CODES 121=y   89=Y
       FOR i=2 to np
          IF dc(i-1)=0 then
             PLOT e$xcr(i),r$ycr(i);
          ELSE IF dc(i-1)<>0 then
```

```
            PLOT  e$xcr(i),r$ycr(i);
         END IF
      NEXT i
   END IF
   GET KEY trash
   CLOSE #1
END PICTURE

!############################## plotter routines ##############################!

SUB plotxy                        !plot for x y z displacement vectors

   LET cx=-100
   LET cy=10
   LET dxl =(fg$xlmax)-(fg$xlmin)
   LET dyl =(g$ylmax)-(g$ylmin)
   LET wxlmin=fg$xlmin-1.5$dxl
   LET wxlmax=fg$xlmax+1.5$dxl
   LET wylmin=g$ylmin-.3$dyl
   LET wylmax=g$ylmax+.3$dyl

   PRINT #2:"ro0"
   PRINT #2:"ps4;"
   PRINT #2:"sp1"
   PRINT #2:"in;ip250,600,10900,7600";";"
   PRINT #2:"papu";250,600,"pd",250,7600;
   PRINT #2:"papd";250,7600,10900,7600;
   PRINT #2:"papd";10900,7600,10900,600;"pu";
   PRINT #2:"papd";10900,600;250,600;"pu";";"
   PRINT #2:"iw";650,800,10000,7200;";"
   PRINT #2:"sc";wxlmin;wxlmax;wylmin;wylmax;";"
   FOR i=1 to n
      PRINT #2:"pa";fg$x1(i)+cx;g$y1(i)+cy;";"
      PRINT #2:"pdpa";fg$x2(i)+cx;g$y2(i)+cy;"pu";";"
      CALL arrxy
      PRINT #2:"pa";fg$x2(i)+cx;g$y2(i)+cy;"pd";fg$x3(i)+cx;g$y3(i)+cy;"pu";";"
      PRINT #2:"pupa";fg$x2(i)+cx;g$y2(i)+cy;"pd";fg$x4(i)+cx;g$y4(i)+cy;"pu";";"
   NEXT i

   !    $ plots the outline $
   IF ans= 121 or ans=89 then
      PRINT #2:"papu";fg$xc(1)+cx;g$yc(1)+cy;";"
      PRINT #2:"sp1"
      FOR i=2 to np
         IF dc(i-1)=8 then
            PRINT #2:"papd";fg$xc(i)+cx;g$yc(i)+cy;";"
         ELSE IF dc(i-1)<>8 then
            PRINT #2:"papd";fg$xc(i)+cx;g$yc(i)+cy;";"
         END IF
      NEXT i
   END IF
   !      $ plots the details   $
   PRINT #2:"iw"
   PRINT #2:"sp1"
```

```
    PRINT #2:"si-.187,-.269";
    PRINT #2:"papu",(wxlmax+cx)/2,wylmax-100
    PRINT #2:"di-1,0;pd","lb X Y DISPLACEMENT VECTORS"
    PRINT #2:chr$(3)
    PRINT #2:"papu",(wxlmax+cx)/2,wylmax-190
    PRINT #2:"di-1,0;pd","lb Vectors magnified x";zmag
    PRINT #2:chr$(3)
    PRINT #2:"papu",(wxlmax+cx)/2,wylmax-210
    PRINT #2:"di-1,0;pd","lb The patient is : ";name$
    PRINT #2:chr$(3)
    PRINT #2:"pu;sp0";
END SUB

SUB plotz

    PRINT #2:"ps4;"
    PRINT #2:"sp1"
    PRINT #2:"df;ip250,600,10900,7600";";"
    PRINT #2:"papu";250,600,"pd",250,7600;
    PRINT #2:"papd";250,7600,10900,7600;
    PRINT #2:"papd";10900,7600,10900,600;"pu";
    PRINT #2:"papd";10900,600;250,600;"pu";";"
    PRINT #2:"iw";650,800,10000,7200;";"
    PRINT #2:"sc";wxlmin;wxlmax;wylmin;wylmax;";"
    FOR i=1 to n
        LET y(i)=y1(i)+zdiff(i)
        PRINT #2:"pa";fg$x1(i)+cx;g$y1(i)+cy;";"
        PRINT #2:"pdpa";fg$x1(i)+cx;g$y(i)+cy;"pu";";"
        CALL arrz
        PRINT #2:"pa";fg$x1(i)+cx;g$y(i)+cy;"pd";fg$x3(i)+cx;g$y3(i)+cy;"pu";";"
        PRINT #2:"pupa";fg$x1(i)+cx;g$y(i)+cy;"pd";fg$x4(i)+cx;g$y4(i)+cy;"pu";";"
    NEXT i
    !    $ plots the outline $
    IF ans= 121 or ans=89 then
       PRINT #2:"papu";fg$xc(1)+cx;g$yc(1)+cy;";"
       FOR i=2 to np
           IF dc(i-1)=8 then
              PRINT #2:"papd";fg$xc(i)+cx;g$yc(i)+cy;";"
           ELSE IF dc(i-1)<>8 then
              PRINT #2:"papd";fg$xc(i)+cx;g$yc(i)+cy;";"
           END IF
       NEXT i
    END IF
    !      $ plots the details $
    PRINT #2:"iw"
    PRINT #2:"si-.187,-.269";
    PRINT #2:"papu",(wxlmax+cx)/2,wylmax-100
    PRINT #2:"di-1,0;pd","lb Z DISPLACEMENT VECTORS"
    PRINT #2:chr$(3)
    PRINT #2:"papu",(wxlmax+cx)/2,wylmax-190
    PRINT #2:"di-1,0;pd","lb Vectors magnified x";zmag
    PRINT #2:chr$(3)
    PRINT #2:"papu",(wxlmax+cx)/2,wylmax-210
    PRINT #2:"di-1,0;pd","lb The patient is : ";name$
    PRINT #2:chr$(3)
    PRINT #2:"pu;sp0";
    PRINT #2:"ro0"
END SUB
```

```
SUB plotl                          ! plots the pseudo stereo vectors L&R

    LET dxl =(fg$xlmax)-(fg$xlmin)
    LET dyl =(g$ylmax)-(g$ylmin)
    LET wxlmin=fg$xlmin-dxl
    LET wxlmax=fg$xlmax+dxl
    LET wylmin=g$ylmin-.15$dyl
    LET wylmax=g$ylmax+.15$dyl

    PRINT #2:"ps4;"
    PRINT #2:"sp1"
    PRINT #2:"in;ip250,600,10900,7600";";"
    PRINT #2:"papu";250,600,"pd",250,7600;
    PRINT #2:"papd";250,7600,10900,7600;
    PRINT #2:"papd";10900,7600,10900,600;"pu";
    PRINT #2:"papd";10900,600;250,600;"pu";";"
    PRINT #2:"iw";650,800,10000,7200;";"
    PRINT #2:"sc";wxlmin;wxlmax;wylmin;wylmax;";"
    FOR i=1 to n
        PRINT #2:"pa";fg$xl1(i)+c;g$yl1(i);";"
        PRINT #2:"pdpa";fg$xl2(i)+c;g$yl2(i);"pu";";"
        CALL arrowl
        PRINT #2:"pa";fg$xl2(i)+c;g$yl2(i);"pd";fg$x3(i)+c;g$y3(i);"pu";";"
        PRINT #2:"pupa";fg$xl2(i)+c;g$yl2(i);"pd";fg$x4(i)+c;g$y4(i);"pu";";"
    NEXT i
    !    $ plots the outline $
    IF ans= 121 or ans=89 then
       PRINT #2:"papu"; fg$xcl(1)+c;g$ycl(1);";"
       FOR i=2 to np
           IF dc(i-1)=8 then
              PRINT #2:"papd";fg$xcl(i)+c;g$ycl(i);";"
           ELSE IF dc(i-1)<>8 then
              PRINT #2:"papd";fg$xcl(i)+c;g$ycl(i);";"
           END IF
       NEXT i
    END IF
    !      $ plots the details $
    PRINT #2:"iw"
    PRINT #2:"si-.187,-.269"
    PRINT #2:"papu",(wxlmax+c)-5,wylmax-5;
    PRINT #2:"di-1,0;pd","lb LEFT HAND PSEUDO PHOTO  "
    PRINT #2:chr$(3)
    PRINT #2:"papu",(wxlmax+c)-5,wylmax-8;
    PRINT #2:"di-1,0;pd","lb Space Vectors magnified x";mag
    PRINT #2:chr$(3)
    PRINT #2:"papu",(wxlmax+c)-5,wylmax-9;
    PRINT #2:"di-1,0;pd","lb The patient is : ";name$
    PRINT #2:chr$(3)
    PRINT #2:"pu;sp0"
    CLOSE #2
END SUB

SUB plotr

    LET dxr =(fg$xrmax)-(fg$xrmin)
    LET dyr =(g$yrmax)-(g$yrmin)
    LET wxrmin=fg$xrmin-dxr
    LET wxrmax=fg$xrmax+dxr
```

```
    LET wyrmin=g$yrmin-.15$dyr
    LET wyrmax=g$yrmax+.15$dyr

    PRINT #2:"ps4;"
    PRINT #2:"sp1"
    PRINT #2:"in;ip250,600,10900,7600";";"
    PRINT #2:"papu";250,600,"pd",250,7600;
    PRINT #2:"papd";250,7600,10900,7600;
    PRINT #2:"papd";10900,7600,10900,600;"pu";
    PRINT #2:"papd";10900,600;250,600;"pu";";"
    PRINT #2:"iw";650,800,10000,7200;";"
    PRINT #2:"sc";wxrmin;wxrmax;wyrmin;wyrmax;";"
    FOR i=1 to n
        PRINT #2:"pa";fg$xr1(i)+c;g$yr1(i);";"
        PRINT #2:"pdpa";fg$xr2(i)+c;g$yr2(i);"pu";";"
        CALL arrowr
        PRINT #2:"pa";fg$xr2(i)+c;g$yr2(i);"pd";fg$xx3(i)+c;g$yy3(i);"pu";";"
        PRINT #2:"pupa";fg$xr2(i)+c;g$yr2(i);"pd";fg$xx4(i)+c;g$yy4(i);"pu";";"
    NEXT i
    !    $ plots the outline $
    IF ans= 121 or ans=89 then
       PRINT #2:"papu";fg$xcr(1)+c;g$ycr(1);";"
       FOR i=2 to np
           IF dc(i-1)=8 then
              PRINT #2:"papd";fg$xcr(i)+c;g$ycr(i);";"
           ELSE IF dc(i-1)<>8 then
              PRINT #2:"papd";fg$xcr(i)+c;g$ycr(i);";"
           END IF
       NEXT i
    END IF
    !       $ plots the details $
    PRINT #2:"iw"
    PRINT #2:"si-.187,-.269";
    PRINT #2:"papu",(wxrmax+c)-5,wyrmax-5
    PRINT #2:"di-1,0;pd";"lb RIGHT HAND PSEUDO PHOTO   "
    PRINT #2:chr$(3)
    PRINT #2:"papu",(wxrmax+c)-5,wyrmax-8
    PRINT #2:"di-1,0;pd";"lb Space Vectors magnified x";mag
    PRINT #2:chr$(3)
    PRINT #2:"papu",(wxrmax+c)-5,wyrmax-9
    PRINT #2:"di-1,0;pd";"lb The patient is : ";name$
    PRINT #2:chr$(3)
    PRINT #2:"pu;sp0";
END SUB

!$$$$$$$$$$$$$$$$$$$$$$  calculates the min and max x,y  $$$$$$$$$$$$!

SUB minmax
    FOR i=1 to n
        IF xl(i) = xlmin then
           LET xlmin = xl(i)
        ELSE IF xl(i) < xlmin then
           LET xlmin=xl(i)
        ELSE
           IF xl(i)=xlmax then
              LET xlmax=xl(i)
           ELSE IF xl(i) > xlmax then
              LET xlmax = xl(i)
```

```
        END IF
      END IF
   NEXT i

   FOR i=1 to n
      IF xr(i) = xrmin then
        LET xrmin = xr(i)
      ELSE IF xr(i) < xrmin then
        LET xrmin=xr(i)
      ELSE
        IF xr(i) = xrmax then
          LET xrmax = xr(i)
        ELSE IF xr(i) > xrmax then
          LET xrmax = xr(i)
        END IF
      END IF
   NEXT i

   FOR i=1 to n
      IF yl(i) = ylmin then
        LET ylmin = yl(i)
      ELSE IF yl(i) < ylmin then
        LET ylmin = yl(i)
      ELSE
        IF yl(i)= ylmax then
          LET ylmax = yl(i)
        ELSE IF yl(i) > ylmax then
          LET ylmax = yl(i)
        END IF
      END IF
   NEXT i

   FOR i=1 to n
      IF yr(i) = yrmin then
        LET yrmin = yr(i)
      ELSE IF yr(i) < yrmin then
        LET yrmin = yr(i)
      ELSE
        IF yr(i) = yrmax then
          LET yrmax = yr(i)
        ELSE IF yr(i) > yrmax then
          LET yrmax = yr(i)
        END IF
      END IF
   NEXT i
END SUB
END
```

APPENDIX 20 IMP.C - TAG PROGRAM TO PRINT IMAGES ON TEKTRONIX

```
#include <stdio.h>
#include <stdlib.h>

void imp()
{
 void mode();
 FILE *fpim,*hardcopy;
 int argc;
 char argv[3][20];
 char ext[5],ext1[5];
 char s[15];
 char ch,cht;
 int i,j;
 long percent;
 unsigned char chksum,c;
 char kill = 't';


 while( kill != 'q')  {
    printf(" CONVERTING AND PRINTING IMAGES FOR TEKTRONIX PRINTER \n");
    printf(" ================================================== \n\n");
    printf(" PATIENT'S NAME : ");
    scanf("%s",s);                          /* s = deb */
    strcpy(argv[1],"\\pip\\i\\");           /* argv[1]=\pip\i\ */
    strcat(argv[1],s);                      /* argv[1]=\pip\i\deb */
    printf("  \n");
    mode();
    printf(" CONVERTING AND PRINTING IMAGES FOR TEKTRONIX PRINTER \n");
    printf(" ================================================== \n\n");
    printf(" 1. for OBJECT image \n");
    printf(" 2. for CONTROL  \n");
    printf(" 3. for STEREO VECTOR image \n");
    printf(" \n\n");
    printf(" Enter number of your choice : ");

    ch=getch();
    printf(" %c \n",ch);
    switch(ch)  {
      case'1':
        mode();
        printf(" OBJECT IMAGE FOR TEKTRONIX PRINTER \n");
        printf(" ================================== \n\n");
        printf(" \n");
        strcpy(ext1,"o");
        strcat(argv[1],ext1);                   /* argv[1]=\pip\i\debo */
        printf(" Enter breath number and in or out i.e. 1i : ");
        scanf("%s",ext1);
        strcat(argv[1],ext1);                /* argv[1]=\pip\i\deboli */
        break;
      case'2':
        mode();
        printf(" CONTROL IMAGE FOR TEKTRONIX PRINTER \n");
        printf(" ================================== \n\n");
        printf(" \n");
        strcpy(ext1,"c");
        strcat(argv[1],ext1);                   /* argv[1]=\pip\i\debc */
        break;
      case'3':
```

```
        mode();
        printf(" STEREO VECTOR IMAGE FOR TEKTRONIX PRINTER \n");
        printf(" ========================================== \n\n");
        printf(" \n");
        strcpy(ext1,"vlo");
        strcat(argv[1],ext1);                    /* argv[1]=\pip\i\debvlo */
        break;
        }

    printf("argv[1] = %s",argv[1]);
    strcpy(argv[2],"\\pip\\imp\\Z");                    /* argv[2]=Z  */
    printf("\n\n BUSY CONVERTING IMAGE FOR TEKTRONIX \n");



    /* signon message
    cprintf("%c%cMatrox to Tektronix 4693D image converter%c%c",
                                          0x0d,0x0a,0x0d,0x0a);
    cprintf("Copyright (C) N. Parkyn - 1988%c%c",0x0d,0x0a);          */
          .
          .(not listed due to copyright infringment)
          .
     printf("\n\n SENDING TO TEKTRONIX \n\n");
     printf(" Ensure Tektronix is connected and switched on \n");
     printf(" Watch Tektronix display \n");
     printf(" Press SPACE BAR to continue : ");
     cht = getch();
     while( cht != 32 ) {
         cht = getch();
         }
     printf("\n TRANSFERING --- WATCH TEKTRONIX IMAGE DISPLAY \n");
     system("\\pip\\imp\\tpi \\pip\\imp\\Z ");
     printf(" \n");
     printf(" \n");
     printf(" Do you wish to do another image for a patient y/n : ");
     while( cht != 'y' && cht != 'n' && cht != 'Y' && cht != 'N' ) {
         cht = getch();
         }
     if( cht=='y' || cht=='Y' ) {
        kill='t';
        }
     if( cht=='n' || cht=='N' ) {
        kill='q';
        }
     }
}
```

APPENDIX 21 GRAPH1.C - TAG PROGRAM TO SUPPLY GRAPHICS FUNCTIONS

```
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <dos.h>

void graph()
{

 /* To find ASCII character codes look in Dr. Hahn's TRUE BASIC book p313 */
 /* or hold down Alt key and type in code no., release Alt key for code   */

 void mode();

 char c1 = (196);                    /* code 196  = - */
 char c2 = (179);                    /* code 179 = | */
 char c3 = (218);                    /* code 218 = ┌ */
 char c4 = (191);                    /* code 191 = ┐ */
 char c5 = (192);                    /* code 192 = └ */
 char c6 = (217);                    /* code 217 = ┘ */
 char c7 = (205);                    /* code 205  = = */
 char c8 = (186);                    /* code 186 = ‖ */
 char c9 = (201);                    /* code 201 = ╔ */
 char c10 = (187);                   /* code 187 = ╗ */
 char c11 = (200);                   /* code 200 = ╚ */
 char c12 = (188);                   /* code 188 = ╝ */

 char ch;
 int t = 0;


 /****  BABY PROGRAM HEADING ************************/

 printf("%c", c9);
 for(t=1; (t<70) && (printf("%c",c7 )); t++);
 printf("%c \n",c10);
 printf("%c                                                          %c  \n", c8, c8);
 printf("%c                                                          %c  \n", c8, c8);
 printf("%c                    NEAR REAL TIME MEASUREMENT            %c  \n", c8, c8);
 printf("%c                    ==========================            %c  \n", c8, c8);
 printf("%c                                                          %c  \n", c8, c8);
 printf("%c                      OF RESPIRATORY MOTION               %c  \n", c8, c8);
 printf("%c                      ====================                %c  \n", c8, c8);
 printf("%c                                                          %c  \n", c8, c8);
 printf("%c                                                          %c  \n", c8, c8);
 printf("%c                          ┌─┐                             %c  \n", c8, c8);
 printf("%c                          │ │                             %c  \n", c8, c8);
 printf("%c                          └─┘                             %c  \n", c8, c8);
 printf("%c                         -[_]-                            %c  \n", c8, c8);
 printf("%c                          J L                             %c  \n", c8, c8);
 printf("%c                                                          %c  \n", c8, c8);
 printf("%c                                                          %c  \n", c8, c8);
 printf("%c                    Copyright(c) B.A.GTSCHOW  1989        %c  \n", c8, c8);
```

```
printf("%c                                                              %c  \n", c8, c8);
printf("%c                    Department of Surveying                   %c  \n", c8, c8);
printf("%c                    University of Cape Town                   %c  \n", c8, c8);
printf("%c                                                              %c  \n", c8, c8);
printf("%c                                                              %c  \n", c8, c8);
printf("%c", c11);
for(t=1; (t<70) && (printf("%c",c7)); t++);
printf("%c \n",c12);


for(t=1; (t<1) && (printf(" \n")); t++);

printf("\n Press SPACE BAR to continue : ");
ch = bdos(7,0,0);
while( ch != 32 ) {
  ch = bdos(7,0,0);
  }
printf(" \n");
mode();
}



/* set screen mode i.e. TEXT B/W, dimensions 80x25, monochrome */
void mode()
{
  union REGS r;
  r.h.ah = 0;                   /*  sets video mode     */
  r.h.al = 2;                   /*  80x25 B/W           */

  int86(0x10, &r, &r);
}

/* clear the screen  */
void cls()
{
  union REGS r;
  r.h.ah = 6;                   /*  see page 625 onwards    */
  r.h.al = 0;                   /*  screen scroll code      */
  r.h.ch = 0;                   /*  start row               */
  r.h.cl = 0;                   /*  start column            */
  r.h.dh = 24;                  /*  end row                 */
  r.h.dl = 79;                  /*  end column              */
  r.h.bh = 7;                   /*  blank line is black     */

  int86(0x10, &r, &r);
}

/*  send cursor to x,y  */
void gotto(x,y)
int x,y;
{
  union REGS r;
  r.h.ah = 2;                   /*  cursor addressing function    */
  r.h.dh = x;                   /*  row co-ordinate               */
  r.h.dl = y;                   /*  column co-ordinate            */
  r.h.bh = 0;                   /*  video page                    */
```

```
  int86(0x10, &r, &r);
}



/* CURSOR */
/* GRAPHIC ON SCREEN INSTRUCTIONS FOR CURSOR MOVEMENT */
void keys(str1,str2)
char str1[20],str2[20];
{
  /* To find ASCII character codes look in Dr. Hahn's TRUE BASIC book p313 */
  /* or hold down Alt key and type in code no., release Alt key for code   */

  void mode();

  char c1 = (196);                      /* code 196  = ─ */
  char c2 = (179);                      /* code 179 = │ */
  char c3 = (218);                      /* code 218 = ┌ */
  char c4 = (191);                      /* code 191 = ┐ */
  char c5 = (192);                      /* code 192 = └ */
  char c6 = (217);                      /* code 217 = ┘ */
  char c7 = (205);                      /* code 205  = ═ */
  char c8 = (186);                      /* code 186 = ║ */
  char c9 = (201);                      /* code 201 = ╔ */
  char c10 = (187);                     /* code 187 = ╗ */
  char c11 = (200);                     /* code 200 = ╚ */
  char c12 = (188);                     /* code 188 = ╝ */
  char c13 = (255);                     /* code 255 = blank */


  char ch;
  int t = 0;

  /****  KEY INSTRUCTIONS ************************/
  mode();
  printf("%-22c ┌┐┌┐┌┐┌┐┌┐┌┐┌┐┌┐┌┐┌┐ ┌───┐ ┌───┐ ┌───┐ \n",c13);
  | \n","PARTIAL KEYBOARD VIEW");
  printf("%-22s └┘└┘└┘└┘└┘└┘└┘└┘└┘└┘ │left │ │   │ │ 25 │ \n","with new key functions");
              │HOME │ │   │ │PgUp │ \n","for the cross movement");
  printf("%-22s     screen centre      └───┘ └───┘ └───┘ \n","on the PHILIPS monitor");
  printf("%-22s                                          \n","═════════════════════");
  printf("%-22c  │ INS │ │HOME │ │Pg Up│   │ 1 │ │   │ │ 1 │ \n",c13);
  printf("%-22c  │     │ │top  │ │     │   │   │ │   │ │   │ \n",c13);
│   │ │ │   │ │   │ \n","PRESS < ENTER > ");
  printf("%-22s └───┘ └───┘ └───┘   └───┘ └───┘ └───┘ \n","TO SAVE POINT WHEN ");
  printf("%-22s                                          \n","AT CORRECT POSITION");
  printf("%-22c  │ DEL │ │ END │ │Pg Dw│   │botta│ │ 1 │ │   │ \n",c13);
  printf("%-22s  │     │ │botta│ │     │   │right│ │   │ │ 25 │ \n","< q > ");
  printf("%-22s  │ 25 │ │right│ │ 25 │   │ END │ │   │ │PgDw │ \n","TO QUIT DIGITISING" );
  printf("%-22c └───┘ └───┘ └───┘   └───┘ └───┘ └───┘ \n",c13);
  printf("%-22c         ┌───┐     ┌──────────┐ ┌───┐ \n",c13);
1│     │    └───┘ │ 25 │ \n",c13);
  printf("%-22c                  │          │ │   │ \n",c13);
  printf("%-22c                  │ INS   25 │ │DEL│ \n",c13);
  printf("%-22s   │1││1││1 │      └──────────┘ └───┘ \n",str1);
  printf("%-22s   └─┘└─┘└─┘                          \n",str2);
  printf("                                           \n");
```

```
 printf(" Digitise pt no              x     y    grey        No.    x     y    \n");
 for(t=1;(t<79) && (printf("%c",c7)); t++);
 printf(" \n");
}

/* MOUSE */
void mice(str1,str2)
char str1[20],str2[20];
{
 /* To find ASCII character codes look in Dr. Hahn's TRUE BASIC book p313 */
 /* or hold down Alt key and type in code no., release Alt key for code  */

 void mode();

 char c1 = (196);                    /* code 196  = - */
 char c2 = (179);                    /* code 179 = | */
 char c3 = (218);                    /* code 218 = ┌ */
 char c4 = (191);                    /* code 191 = ┐ */
 char c5 = (192);                    /* code 192 = └ */
 char c6 = (217);                    /* code 217 = ┘ */
 char c7 = (205);                    /* code 205  = = */
 char c8 = (186);                    /* code 186 = ‖ */
 char c9 = (201);                    /* code 201 = ╔ */
 char c10 = (187);                   /* code 187 = ╗ */
 char c11 = (200);                   /* code 200 = ╚ */
 char c12 = (188);                   /* code 188 = ╝ */
 char c13 = (255);                   /* code 255 = blank */
 char ch;
 int t = 0;

 mode();

 printf("%c          MOUSE MOVEMENTS FOR CHEST OUTLINES          %c  \n", c13, c13);
 printf("%c          ==================================          %c  \n", c13, c13);
 printf("%c                                                      %c  \n", c13, c13);
 printf("%c                                                      %c  \n", c13, c13);
 printf("%c     LOGITECH                                         %c  \n", c13, c13);
 printf("%c                                                      %c  \n", c13, c13);
 printf("%c       CURSOR          ENTER        QUIT              %c  \n", c13, c13);
 printf("%c       MOVEMENT                                       %c  \n", c13, c13);
 printf("%c                                                      %c  \n", c13, c13);
 printf("%c       1. 1 PIX.                                      %c  \n", c13, c13);
 printf("%c       2. 5 PIX.                                      %c  \n", c13, c13);
 printf("%c          JUMPS                                       %c  \n", c13, c13);
 printf("%c                                                      %c  \n", c13, c13);
 printf("%c                                                      %c  \n", c13, c13);
 printf("%c     MOVE MOUSE TO MOVE CURSOR ON SCREEN              %c  \n", c13, c13);
 printf("%c        USE 5 PIX. JUMPS FOR OUTLINE                  %c  \n", c13, c13);
 printf("%c                                                      %c  \n", c13, c13);
 printf("%c                                                      %c  \n", c13, c13);
 printf("%s                                  \n", str1);
 printf("%s                                  \n", str2);
 printf("%c                                                      %c  \n", c13, c13);
```

```
printf(" Digitise pt no              x     y   grey        No.    x     y    \n");
for(t=1;(t<79) && (printf("%c",c7)); t++);
printf(" \n");

}
```

APPENDIX 22 README.DOC - TAG README DOCUMENT

README.BAT

```
-----------
rem This batch file runs the typing of readme.doc
rem for TAG on screen.

echo on
rem  The README.DOC will be called up on screen
rem
rem  PRESS PAUSE TO STOP THE SCROLLING OF THE DOCUMENT
rem  PRESS ENTER TO CONTINUE SCROLLING
rem  PRESS Ctrl Break TO EXIT OUT OF README.DOC
rem
rem  For a hardcopy on the printer type >> type readme.doc >lpt1
rem
rem
pause
type readme.doc
```

README.DOC

The readme.doc can either be read by calling it up on an editor such as Minder or PC Tools or on a programming editor such as True Basic. Readme.doc can also be called up on screen by using a batch file Readme.bat.

Both the Readme.doc and the Readme.bat are in the directory c:\barb\tag on the IBM computer.

To run TAG the user has to be in the TAG directory, i.e c:\barb\tag The user then has to type the words TAG and press the enter button. It is assumed that the user of the system has an understanding of the process involved at the hospital and knows something about photogrammetry and how to view the stereophotographs in three dimensions.

(In the Readme.doc this process of typing in the word and pressing the enter button will be referred to as simply: Enter TAG)

TAG

TAG can be viewed as a glorified digitiser. It replaces the traditonal process of developing the photographs, placing them in a stereoscomparator and measuring the points on the images, using the computer connected to the stereocomparator to calculate 3D co-ordinates. TAG replaces this time consuming process with a single computer system that only takes minutes to obtain final results in graphical form.

Once TAG has been called up, the Title page is called up on screen. At the bottom of the page the wording "Press SPACE BAR to continue" appears. On pressing the space bar the main menu of TAG is called up.
(Throughout TAG single response keys do not need to be entered, they are immediately accepted when the key is hit. If there is no program response, the wrong key has been entered. If the user is prompted to press the space bar to continue, the program will not progress until the space bar and only the space bar has been

eneterd. This prevents the user from inputting an incorrect response. The only draw back is that the user can not quit the program at such a response stage as the program will not accept the Ctrl Break command. The Ctrl Break command will only be accepted at a command that requires several keystrokes and an enter command.)

MAIN MENU

The main menu consists of:

```
"  1. Image transfer to IBM from video
   2. Menu : Digitising
   3. Menu : Calculations
   4. Menu : GRaphics
   5. Image printing to Tektronix
   0. Quit
---------------------------------------
   Enter number of your choice : _      "
```

The main menu follows the sequence of events that need to occur to obtain the final results. By entering the number of the desired option, that option is called up.

"1. Image transfer to IBM from video "

Images of a video session of a patient have to be transferred from the video machine to the IBM. On screen instructions are given on how to connect the video machine to the IBM computer.

The patient's name then has to be eneterd (4 or less keystrokes must be used for the patient's name, as the other 4 permissible keystrokes for a file name are used for labelling the different data files for the patient). The user is then asked whether he wishes to "Transfer a control frame image ? y/n". If yes on screen instructions follow on how to operate the video machine and freeze the correct image. Pressing the SPACE BAR activates

the image transfer process.

After the control image has been transferred, the user is asked whether he wishes to "Transfer an object (patient) image ? y/n". If yes the user is asked the number of the breath and whther it´s at the extreme of the breath in or out of the breathing cycle, e.g. 1i - breath: 1, extreme breath in: i.

The same procedure is followed for the transfer of object images as it was for control images.

The user is given the option to transfer further object images of that patient or to process another video session of another patient. If no more image transfers are required, the user is returned to the main menu.

"2. Menu : Digitising"

Option 2 of the main menu is for digitising the images that have been transferred to the IBM.

Menu : Digitising consists of the following options:

1. Stereo control image - LEFT
2. Stereo control image - RIGHT
3. Stereo object (patient) image - LEFT
4. Stereo object (patient) image - RIGHT
5. Single image
6. Patient´s outline
0. Return to MAIN MENU

Suboptions 1 to 5 all operate in a similar manner, all using the same subprogram to perform the digitising. Suboption 6 can only be operated when 3D co-ordinates for points on the image, that is being used to digitise the outline, exist. The 3D co-ordinates are required for the transformation of the outline.

Suboption 1 - Stereo control image - LEFT

ips display monitors on. The user then has to enter the patient's name and what control frame was used the small or the large one.

An image of the control frame for that patient is brought up on the left monitor. On the right monitor the image of the control frame specified by the user is brought up, with all the control points numbered. This serves as a guide for digitising the patient's control frame image. On both monitors the right part of the image is blocked out and on the left monitor the words "DIGITISE LEFT IMAGE" appear so that the user will only digitise the left part of the stereo image.

The user then has to enter the number of the points that are not visible on the patient's control frame image in ascending order. From this the computer can allocate the correct numbers to the points digitised to correlate them with the previously determined 3D co-ordinates of the control points.

The keyboard menu is then called up on the IBM screen to show the user how to operate the keys to move the cursor on the image on the control image. The last two lines of the menu consist of:

"Digitise pt. no	x	y	grey	No.	x	y
2	103	214	212	1	10.4	51.2"

and show the user what the co-ordinates of the present cursor position are, the number and co-ordinates of the last point to be entered and the number of the next control point to be digitised.

The cursor is moved form control point to control point, in the order specified by the computer. The cursor must be placed at the approximate centre of a control point before pressing the enter button.

At the end of the digitising sessionpress "q" for quitting. The user is returned to the digitising menu.

Suboption 2 - Stereo control image - RIGHT

This suboption is the "mirror image" of suboption 1, performing the same routine except on the right part of the stereo control image instead of the left.

Suboption 3 - Stereo object (patient) image - LEFT

The patient's name and the breath number e.g. 1i must be entered. The image is then diplayed on the left Phillips monitor.
The user is then asked whether this is the first breathing frame of the patient. The patient object images all have four fiducial marks to relate the different object images to one another, as a video shift may have occured on transfer of the video images to the IBM. If it is the first frame the fiducial co-ordinates of the first frame are written to a separate file to serve a reference for subsequent images. The user is then instructed to first digitise the four fiduical marks and then the left object points. These points must be always observed in the same order for eacg object image of that patient, so that they are always numbered in the same order. Before digitising the user has to check that points to be digitised are clearly visible on both the left and right stereo image. Both the left and right iamge of a point must be digitised to be able to calculate three dimensional co-ordinates for that point.

The keyboard menu and the digitising process is the same as described in suboption 1 of the digitising menu.

Suboption 4 - Stereo object (patient) image - RIGHT

This suboption is very similar to suboption 3 of the digitising menu, except that it digitises the right object image instead of the left object image of a stereo pair.

Suboption 5 - Single image

This suboption was created for research and checking purposes and

is not as user friendly as the other options of the TAG system as it does npt have a specific purpose. The user must enter the full image name, i.e. "namec" or "nameo1i" for control and object images respectively. The file name and extension to which the data is to be written must also be enterd. (All data created by TAG is written as text to text files. A experienced computer user will not have any problems editing data files using a text editor. Normally this should not be done, but in research work it is sometimes the quickest way to achieve results.)
The keyborad menu and the digitising process is the same as the previous suboptions.

Suboption 6 - Patient´s outline

As mentioned earlier this suboption is only to be run if 3D co-ordinates for this image have been calculated.

The user must enter the patient´s name and the breath number of the image he wishes to digitise. The image is displayed on the left hand monitor. If 3D co-ordinates do not exsist for this image the user is asked whether he wishes to calculate them. If yes the user is taken to the calculating menu to calculate them else he is returned to the digitising menu.

If 3D co-ordinates exist the user is asked to enter the numbers of two diagonally opposite object points, one in hip and one in shoulder region. These points are required to transform the patient´s outline.

The user is then informed that the digitiser works on point or stream, in point mode only points that are enterd are written to a data file, in stream mode every cursor position is written to file.

The cursor still operates on the normal keyboard layout, but for the patient´s outline the mouse can also be used and the mouse

layout is on screen during digitising. First the two diagonally opposite points must be digitised and the user is prompted by the instructions on screen. The mouse layout is then slightly altered showing the user that the digitiser is in point mode and can be changed to stream mode by entering the key z and back to stream mode by entering the key x. The mode, in which the cursor is presently is displayed in, is displayed on screen.

In point mode the cursor is moved to the start of the outline using the mouse. Changing to stream mode the user can then start to digitise the outline. The right mouse menu, as indicated by the mouse layout is used to quit at the end of the session. Processing time to return to the digitising menu may appear slow. This is due to the transformation calculations taking place at the end of the program to transfrom the outline into the 3D co-ordinate system of the patient.

"3. Menu : Calculations "

Two suboptions exsist in this menu:

1. B parameters for a camera set up
2. 3D co-ordinates of object points
0. Return to MAIN MENU

Suboption 1 - B parameters for a camera set up

Suboption 1 of the calculating menu is used to calculate the B parameters for a camera set up, i.e.for the set up for that particular patient. The B parameters must be calculated before the 3D co-ordinates as they are needed to calculate those co-ordinates.

The user has to enter the patient´s name, whether the large or

small control frame was used and whether a soft and hardcopy output of accuracies is required. Soft copy - on screen, Hard copy - on a printout, if no hardcopy is required the results will only appear on screen.

If a hard copy is required the user is asked to ensure that the printer is connected and ON LINE. As both the printer and the Tektronix image printer use the same printer cable, this has to be moved from printer to Tektronix. The printer devices should only be connected if they are switched off, else they might blow their fuses.

Results that appear on screen (and printer) only 18 accuracies are printed at a time to give the user time to peruse them. Press Space Bar continues the accuracy output. Only accuracies of common control points can be calculated and printed due to the nature of the transformation.

Suboption 2 - 3D co-ordinates of object points

The user has to enter the patient´s name and the breath number, e.g. 1i. Again the user is asked whether a soft and hardcopy of the results is required. The same procedure for the printer is followed as in the previous suboption, if a hardcopy is required.

"4. Menu : Graphics"

Option 4 of the main menu, menu graphics, consists of the following:

1. Stereo vectors on stereo video frame
2. Displacement & 3D vectors - soft and hardcopy
0. Return to MAIN MENU

Suboption 1 - Stereo vectors on stereo video frame

The user must enter the patient´s name and the breath out and breath in for which the vectors are to be drawn, e.g. 1o and 1i.

Vectors are plotted on the image of the breath out, i.e. on 1o, on the left monitor. Usually vectors plotted at their original size are very small and therefore a different magnification is required. The user is prompted for a different magnification? y/n. If yes the video image is refreshed and the vectors are plotted at the new magnification. The user is then asked whether he wishes to save this image. This process of changing magnification and storing is repeated until the user is happy with the vectogram. This vectogram is then saved.

The user is then asked whether he wishes to process another vectogram. If yes the whole process is repeated, if no the user is returned to the graphics menu.

Suboption 2 - Displacement & 3D vectors - soft and hardcopy

The user is informed that the program plots the displacement vectors in the X-Y plane and the 3D vectors, using the breath in and breath out data, both on screen and on the HP plotter.

This program is written in True Basic and is called up by TAG which is written in C language, therefore the calling of this program and the calculation time of this program are slower in comparison to the other programs.

The user has to enter the patient´s name, the breath number e.g. 1, and whether an outline is required.

The user is then asked wether he requires to view the displacement vectors. If yes, the XY displacement vectors and the Z displacement vectors are plotted on screen.

The user is given the option to plot the displacement vectors on the HP plotter. If plots are required the user is given time to change the paper between plots.

The user is then given the option to view the stereo vectors. The user is asked whether this is a left or a right picture. As with the displacement vectors the stereo vectors are first plotted on screen and the option exsists to plot them on the plotter.

The last option of this program is an anaglyph. The user is informed that this is only for colour screens and therefore not visible on the IBM, which has a monochrome screen. On a computer with a colour screen the left an right vectors are plotted in red and green respectively. The vectors can be observed in three dimensions through a pair of spectacles with one green and one red lens. At the end of the program the user is returned to the graphics menu.

"5. Image printing to Tektronix"

The user must enter the patient´s name. The image printing menu is then called up on screen:

1. for OBJECT image
2. for CONTROL image
3. for STEREO VECTOR image

The program to process the images to convert them for the Tektronix takes a couple of minutes and uses up 787K of memory. If the harddrive is already full the processing of the image can not take place. The program has to be stopped and space made on the harddrive before the program can be run.

Once the image is processed, the user is prompted to ensure that the Tektronix has been connected and switched on and to watch the display panel on the Tektronix. Pressing the SPACE BAR starts the

transfer of the image from the IBM to the Tektronix. The display panel indicates to the user the stage of processing the image print.

After the image has been sent to the Tektronix the IBM is "freed". The user is then able to process another image or return to the main menu.

END OF README.DOC
