CAGE CODE 25500

DEFENSE SYSTEMS DIVISION

LORAL CORPORATION

AKRON, OHIO 44315

3D LASER RADAR VISION PROCESSOR SYSTEM

FINAL REPORT

CONTRACT #NAS9-18187

SUBMITTED TO:
NASA JOHNSON SPACE CENTER

FROM:
LORAL DEFENSE SYSTEMS - AKRON

OCTOBER 1990

Prepared by:

T. M. Sebok
Systems Engineer

Approved by:

D. L. Rohrbacher
Project Engineer

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

LDS-Akron has developed a 3D Laser Radar Vision Processor system capable of detecting, classifying, and identifying small mobile targets as well as larger fixed targets using 3-dimensional laser radar imagery for use with a robotic type system. This processor system is designed to interface with NASA Johnson Space Center in-house EVA Retriever robot program and provide to it needed information so it can fetch and grasp targets in a space-type scenario.

2.0 HARDWARE DESCRIPTION

The 3D Laser Radar Vision Processor system is an IBM-XT compatible computer with an INMOS board containing four transputers inserted in one of the IBM-XT expansion slots. This hardware is illustrated in Figure 2.0-1. The logical connection of the hardware is shown in Figure 2.0-2.

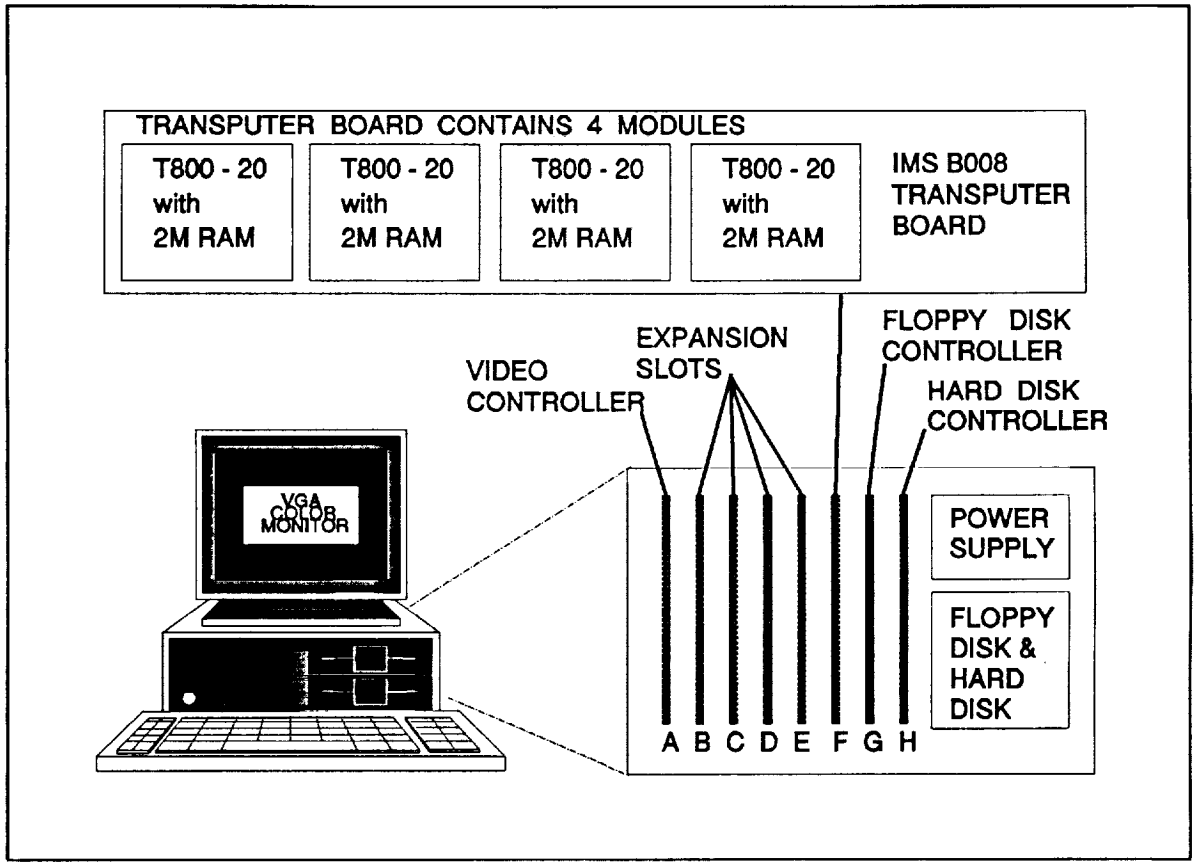

Figure 2.0-1. PC Host Hardware

The delivered hardware consists of the following items:

1. IBM-XT compatible computer
2. Transputer add-in board for the IBM-XT
3. VGA monitor and card

The computer system is an IBM-XT compatible computer which serves as a stand-alone development system for the 3D Laser Radar Vision

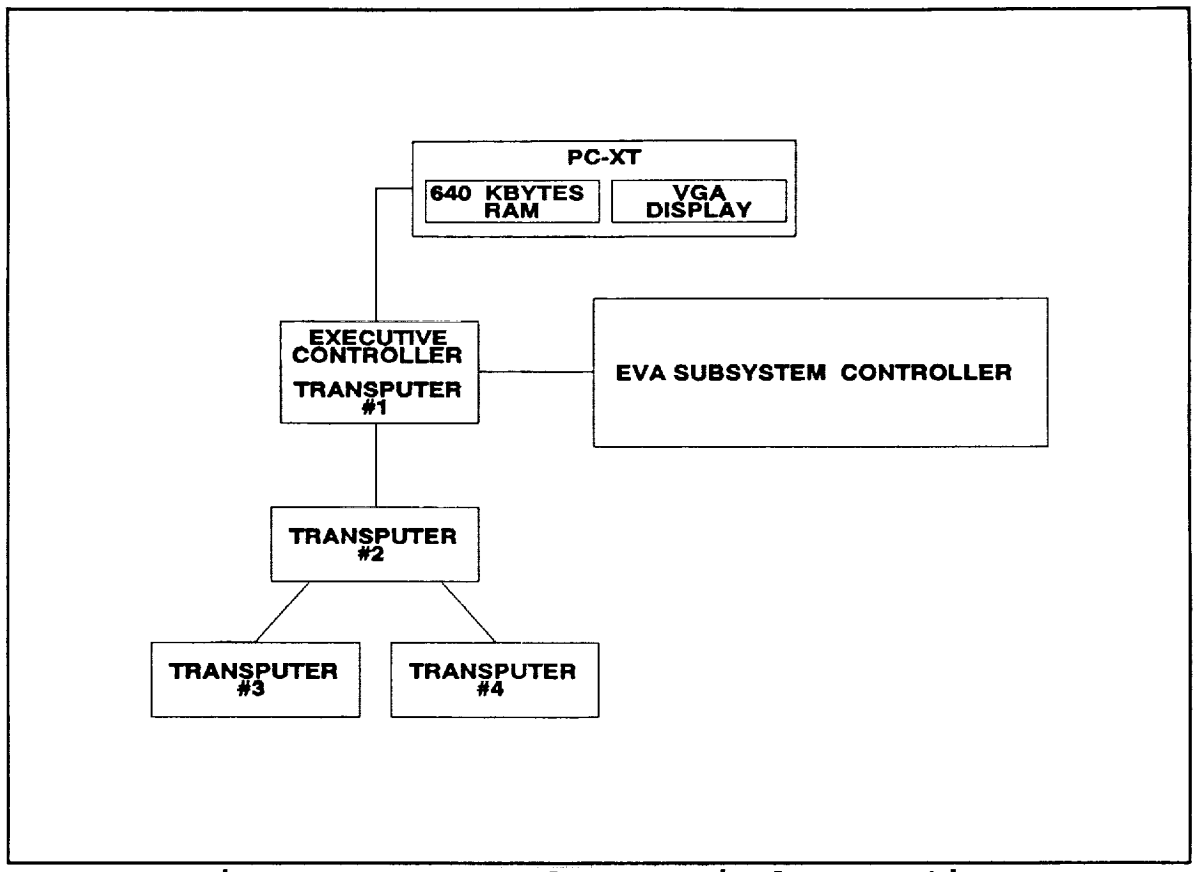

Figure 2.0-2. Hardware Logical Connection

Processor. This computer includes the following: a 10Mhz XT motherboard, 40 MBytes hard disk, one 360K floppy, case, 150W power supply, keyboard, VGA monitor, and VGA display card.

The heart of the Loral's 3D Laser Radar Vision Processor is an IMSB008 Transputer board populated with four IMSB404 modules. The IMSB008 Transputer board is an add-in board for the IBM PC, which takes up one slot in the PC and provides support for up to ten INMOS Transputer modules. This support includes a communication link between the XT and the transputer's network and the interconnection network between the transputers. The transputer interconnection network is provided by an on-board IMS C004 link switch. The IMS C004 allows the user to specify transputer interconnections without doing any physical wiring. Controlling the IMS C004 is an on-board T212 processor.

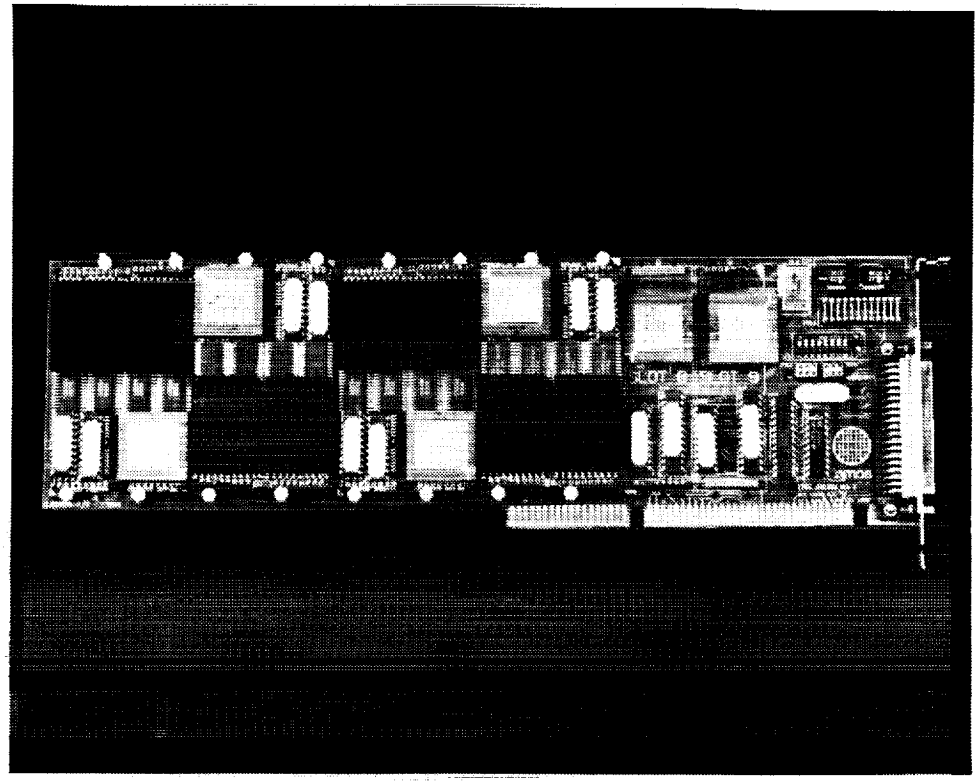

Figure 2.0-3. B008 Transputer Board

The transputer module that will be used with the IMSB008 Transputer board is the IMS404 module. The IMS404 module contains one 20 MHz INMOS T800 Transputer along with 2 MBytes of dynamic RAM memory. The T800 transputer is a 32-bit floating point RISC processor. An integral part of the T800 Transputer is its ability to communicate with up to four other transputers via high speed (2.35 MBytes) serial link.

3.0 SOFTWARE OVERVIEW

3.1 INTRODUCTION

In this section a description of the real time 3-D Laser Radar Vision Processor software will be given. The purpose of this software is to perform three main functions: to provide a human interface for supplying commands and system parameters; to provide a machine interface for communication with the main host system; and to detect, classify, and track targets and objects.

To accomplish this, there are over 12 concurrent parallel processes executing on over five processors. Four of the five processors are capable of operating at over ten million operations per second. The following are the five processors: four INMOS T800 transputers and one INTEL 8088 microprocessor. The INTEL 8088 processor performs the first function which is to provide a human interface for supplying commands and system parameters. The remaining four processors, the transputers, perform all the tasks needed to detect, classify and track targets. A portion of the duty assignments of one of the transputer is also to provide the machine interface for communication with the system's host.

To accomplish target detecting, classifying and tracking in real time, the algorithm functional block diagram (Figure 3.1-1) is

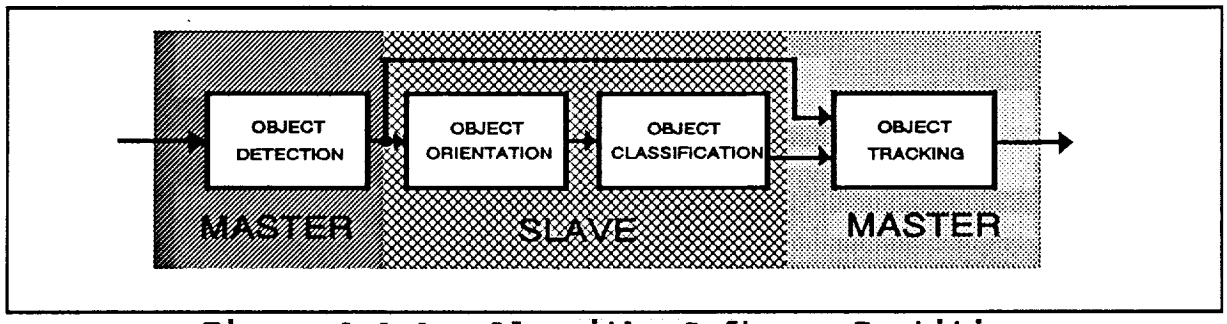

Figure 3.1-1. Algorithm Software Partition

partitioned into software tasks as shown. The rationale in choosing this particular partition is so that the initial object detection can be done by one processor while the more computational complex task of classifying the targets can be divided among many processors.

For each of the five processors in the system, the real time software can be classified into two main categories: algorithm specific software and non algorithm specific software (executive software). Algorithm specific software performs any of the tasks relevant to detecting, classifying, and tracking of targets. The

non-algorithm specific software performs the task of being the software executive.

3.2 SOFTWARE EXECUTIVE OVERVIEW

Contained in each of the five processors in the system is a software executive. The software executive can be thought of as the glue which holds all the algorithm tasks together. It provides to the system an operating environment in which the algorithm tasks can execute efficiently their tasks isolated from the hardware and from each other, and in which all the necessary system timing and task scheduling are provided so that targets can be detected, classified and tracked on a continuous basis. Other important duties provided by the software executive are the following: processing system commands, updating system parameter, obtaining the radar sensor image frames, providing the various algorithm tasks with the appropriate input information, handling the results of the algorithm tasks and the displaying of the final system results.

The 3D Laser Radar Vision Executive provides an OCCAM multi-tasking shell that is wrapped around the FORTRAN Object Detection and Classification core. This executive is hosted on an INMOS B008 motherboard with four (4) T800 transputer modules inter-connected to form the topology of a right sided 'T', as shown in Figure 3.2-1. The base of the 'T' contains the Master Executive

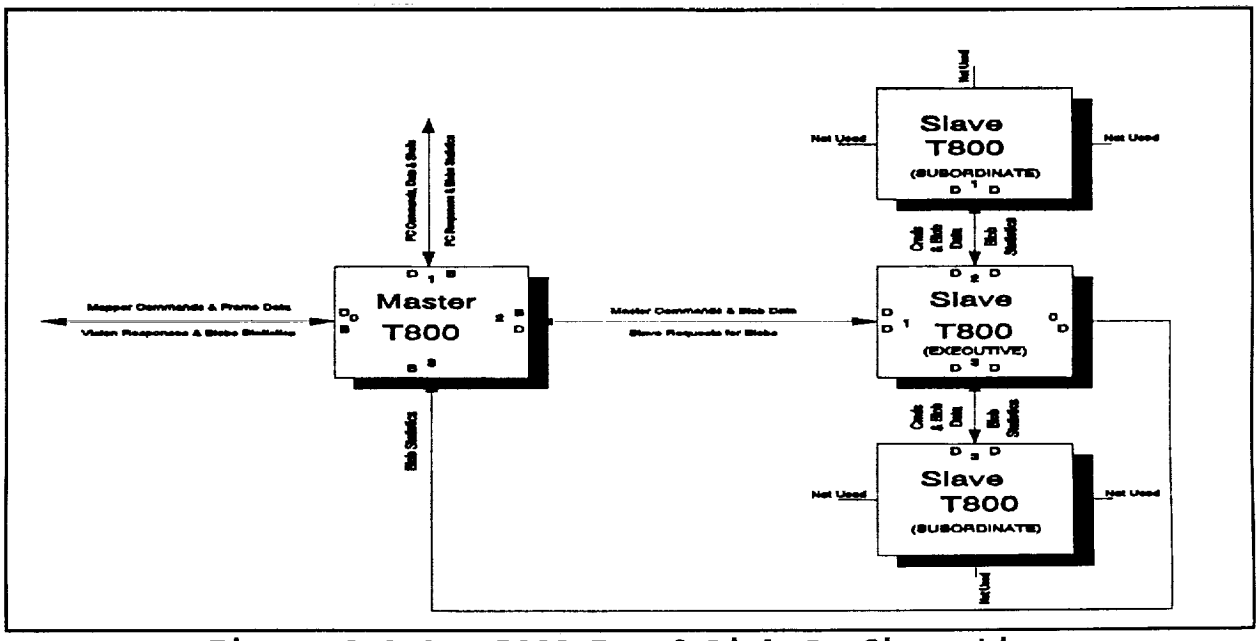

Figure 3.2-1. B008 Board Link Configuration

which wraps around the Object Detection software. The top of the 'T' topology supports the Slave Executive which wraps around the Object Classification software and is composed of the three (3) remaining transputers.

Real Time laser data and initialization parameters are input to the Master transputer and unclassified objects are detected. The detected objects are passed to the Slave transputers where they are compared against the prestored shells and are then classified. The classified objects are then passed back to the Master transputer to be sent to the Mapper interface as Blob Structures, see Figure 3.2-2. The blob structures describe the blobs form and the reliability of a match to a selected model.

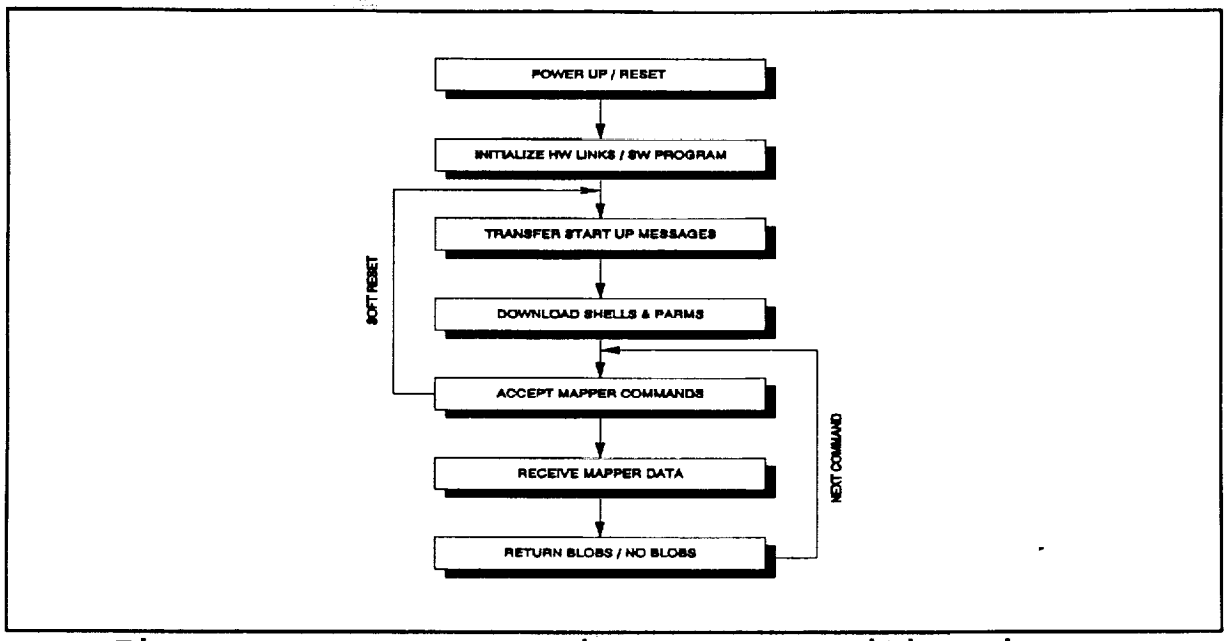

Figure 3.2.1-1. Executive State Transition Diagram

3.2.1 MASTER EXECUTIVE

The Master executive, depicted in Figure 3.2.1-1, provides all of the real-time communication with the Mapper interface as well as the user through the PC. During initialization, see Figure 3.2.1-2, the Master executive establishes communication

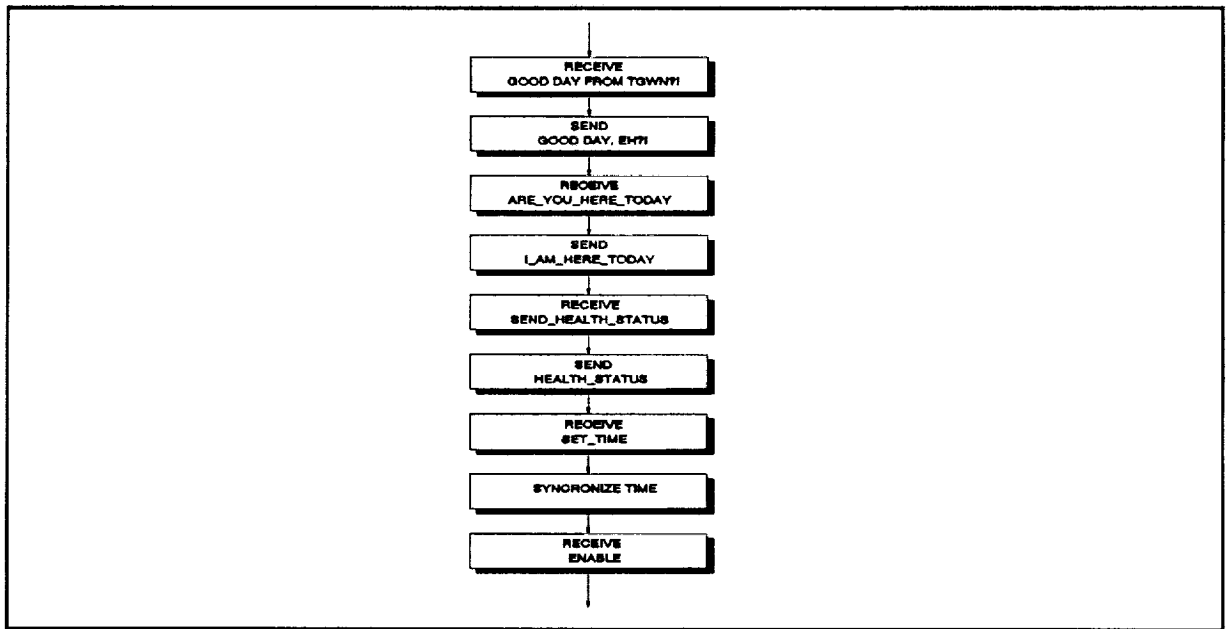

Figure 3.2.1-2. Start Up Messages State Transition Diagram

between the slaves and the Master transputer by sending and receiving a pre-defined message string. The Master executive then reads the pre-stored parameters and shell models passing them to the slaves. Communication is then established with the Mapper interface using the same pre-defined message string. Status information is displayed on the PC to provide step-by-step information on initialization progress. After proper, error free, initialization the Master executive tasks off the Detector and Tracker as well as other Master support functions.

The executive changes the display resolution from 80x25 to 320x240 for Mapper interface command processing. The increased resolution allows the Master executive to display range and intensity/blob information on the PC's VGA display. If DEBUG is turned on, commands are scrolled at the bottom of the screen. Pertinent blob information is displayed on the right side of the display for every frame of laser data.

Commands sent between the Mapper Interface the Master executive are shown in Table 3.2.1-1. These commands provide the protocol

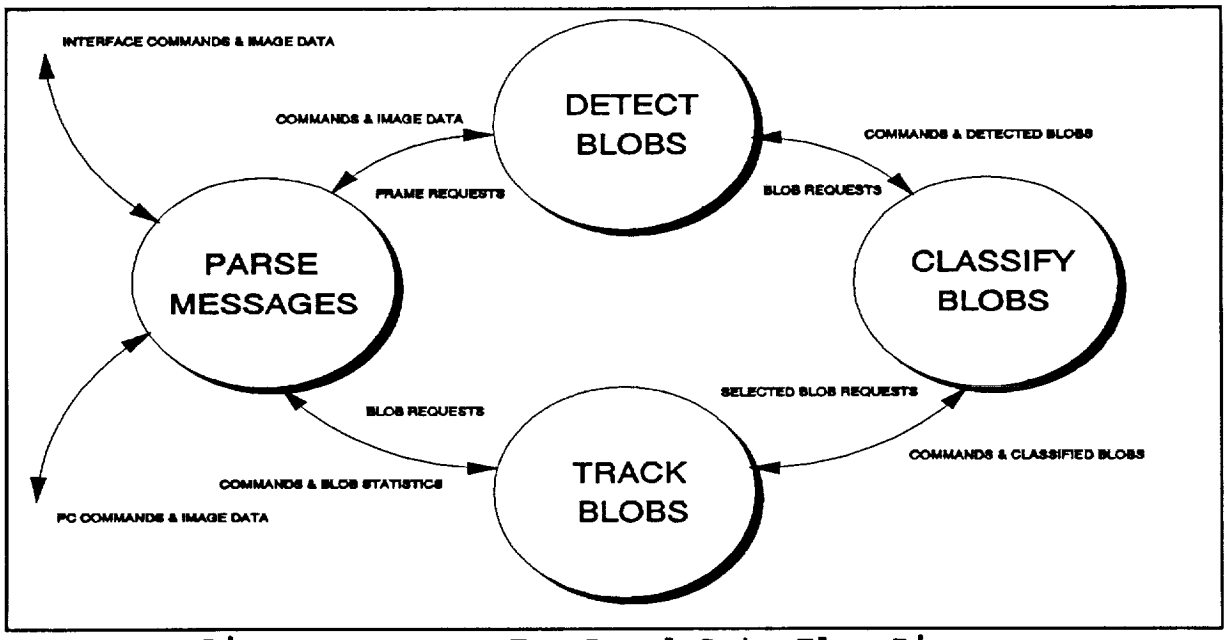

Figure 3.2-2. Top Level Data Flow Diagram

INTERFACE COMMAND	EXECUTIVE RESPONSE	ALTERNATE RESPONSE
RESET		
GOOD DAY FROM TGWN?!	GOOD DAY, EH?!	
ARE_YOU_HERE_TODAY	I_AM_HERE_TODAY	
SEND_HEALTH_STATUS	HEALTH_STATUS	
SET_TIME		
ENABLE		
BLOB_DETECT (AUTO)	MAPPER_XFER_FRAME_OF	
BLOB_TRACK (AUTO)	MAPPER_XFER_FRAME_OF	
SENDING_RAW_MAPPER_D	SENDING_BLOBS	NO_BLOBS_FOUND
TARGET_ACQ_MANUAL	SENDING_BLOBS	NO_BLOBS_FOUND
TARGET_ACQ_AUTO	SENDING_BLOBS	NO_BLOBS_FOUND
STOP_BLOB_TRACK		
DETERMINE_GRASP	NO_GRASP_REGIONS	
MONITOR_GRASP		
DISABLE		

Table 3.2.1-1 System Command and Response Definition

required to set the mode of operation, laser frame requests and other miscellaneous operations between each frame of laser data. The commands are decoded by the Master executive and the required operation is performed.

Each blob detect/track cycle begins with a request for laser data. After the laser data is received the executive displays the range and intensity data. The executive then passes the data to the detector along with prestored parameters for detection adapted to 8 or 9 bit laser data.

The Detector extracts blobs from the image and provides the extents of the blobs as well as the blob object array. This data is then compressed and sent to the Slave executive for Classification processing. After receipt of classified blobs, the executive returns the blob structures to the Mapper interface.

The cycle of requesting laser data and returning blob structures occurs on a single cycle for Target Acquisition Manual and Automatic where Blob Detect and Track produce multiple request for frame data.

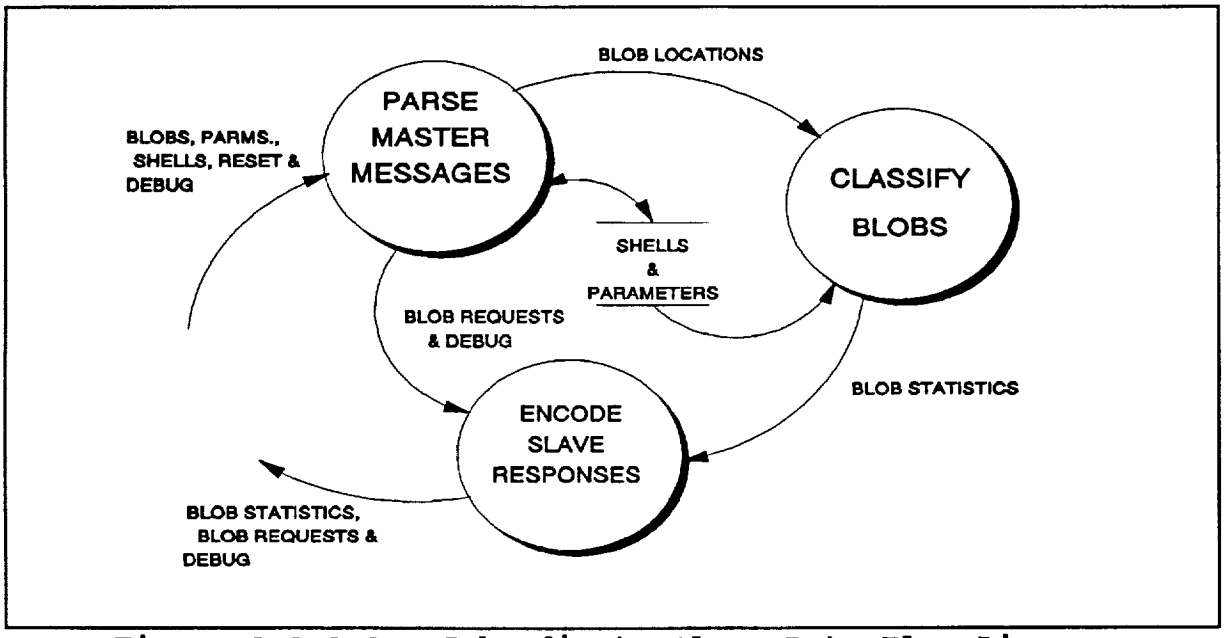

Figure 3.2.2-1. Subordinate Slave Data Flow Diagram

3.2.2 SLAVE EXECUTIVE

The Slave executive, depicted in Figure 3.2.2-1, provides all of the real-time communication with the Master interfaces. During initialization, see Figure 3.2.2-2, the Slave executive establishes communication between the slaves and then the Master transputer by sending and receiving a pre-defined message string. The Slave executive then reads the pre-store parameters and shell models passed to it from the Master Executive.

After proper, error free, initialization the Slave executive tasks off the Classifier in all three slave transputers, see Figure 3.2-1. The Slave executive then waits for detected blob information. When detected blobs are received, they are passed one blob at a time to whichever slave is currently idle. If all of the slaves are currently working on blobs, the remaining blobs are buffered to wait for the next available slave.

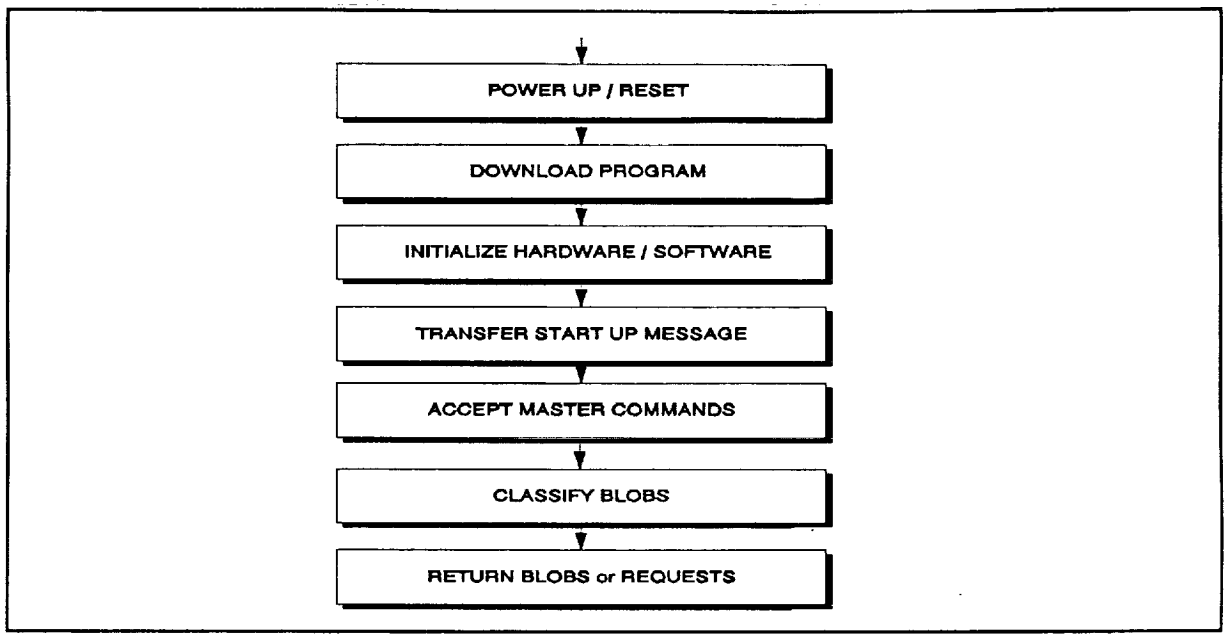

Figure 3.2.2-2. Slave Top Level State Transition Table

As blobs are classified, they are returned to the Master executive to be tracked. Once all the objects are processed by the tracker, they are formatted and sent to the host system via the Mapper interface and also to the IBM PC XT computer for displaying of the results. The blobs are returned over a separate link to increase blob bandwidth.

4.0 ALGORITHM OVERVIEW

4. 1 INTRODUCTION

In fulfillment of the SOW requirements for the 3-D Laser Radar Vision Processor program, LDS-Akron developed and tested a system that detects and classifies targets present in laser radar imagery for a robotics system type application.

Our approach used to perform this task is a model-based vision approach. A model vision approach stores the model representation of the intended target for the robotics system in the processor memory. Each model consists of a surface shell which implicitly stores the 3-D shaped of each target, along with ancillary data. The processor compares what the robot's ladar eye sees to the stored models. Once the processor recognizes the shape and orientation of a object, the robot can then safely act on it. A model-based vision approach for target detection and classification is very versatile since: the target model depends on the sensor characteristics; mission planning is reduced significantly; training sets are not required; target models can be added to the target set quickly.

As illustrated in Figure 4.1-1, the model-based vision algorithms consists of four functional stages: object detection, object

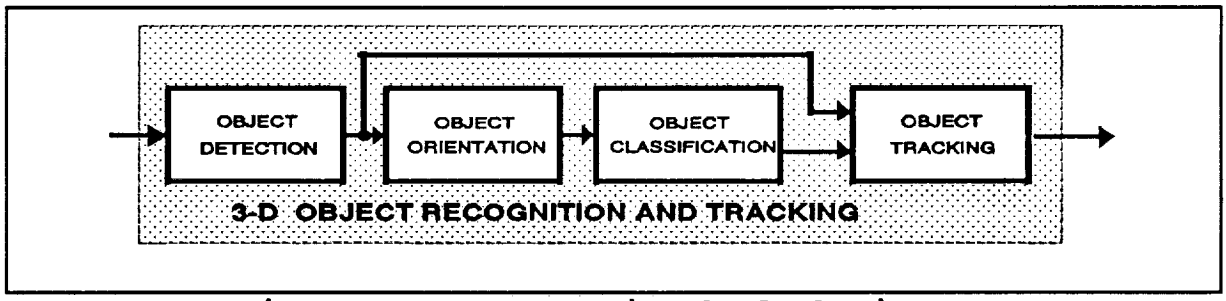

Figure 4.1-1. Functional Block Diagram

orientation, object classification and object tracking. Prior to the operation of the robot, the target models for the particular mission are selected from a pool of target models. After a frame

of data is received, the data is handed off to the object detection stage. The object detection stage examines all of the pixels of the frame of data and determines which of them are object pixels. Once objects are identified, they are passed on to the object orientation stage for determination of the object pose. Using this information, the object classification stage will then orient and match each of the objects against the prestored target models. Once the identities of the objects are known (i.e. type of target or non-target (object blob)), the object tracker stage will take over and predict object movement. This information will then be pass back to the robot for possible evaluation and retrieval.

4.2 OBJECT DETECTION

4.2.1 INTRODUCTION

The object detection stage, the first stage, locates areas of interest (object cores) and then grow these object cores into objects. Two object detection techniques are employed to do this function: intensity detection and range variation detection. This is illustrated in Figure 4.2.1-1. The intensity detection technique,

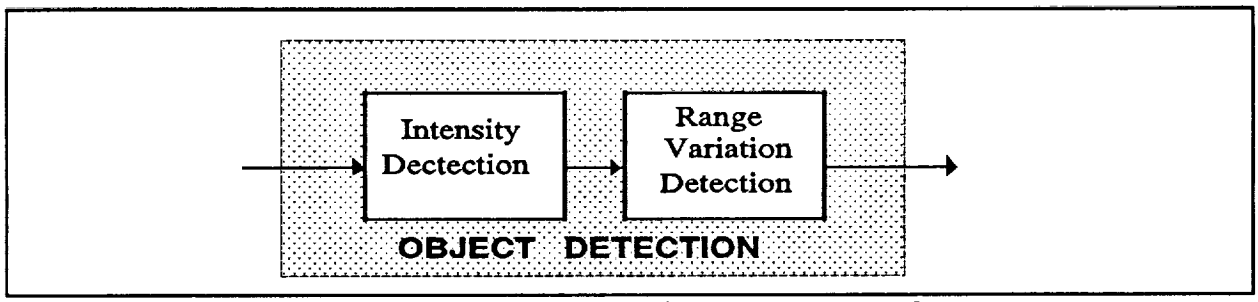

Figure 4.2.1-1. Functional Block Diagram

while simpler computationally, is less accurate than the range variation technique. Each of these detection techniques with their advantages and disadvantages could be used independently to determined and defined objects of interests, but when used together, can also do it more rapidly and accurately. The object detection stage starts out by searching for objects using the

intensity detection techniques. This technique is employed to rapidly determine areas of concentration (object cores) for the more accurate but more complex range variation detection technique. Once an object core is determined, it is then verified and grown using the range variation technique into the final object for the next stage of target classification processing, the object orientation stage. The result of this region growing process are objects that are completely represented by a group of pixels.

4.2.2 INTENSITY DETECTION

The intensity detection technique examines each pixel in the image, and then, based on the intensity differences between object and background at a given range, determines if it belongs to an object or not. This technique assumes that object intensity is lighter than the background intensity. That is, an object appears lighter than the background. This assumption is justified for the application of a space based system since most of the background pixels will be bad returns. If, by chance, a background pixel was falsely identified as belonging to an object, it would then be filtered out in the latter stages of processing. Based on this assumption, an adaptive threshold can be made to determined if a pixel - $P_{i,j}$ belongs to an object or not. That is,

IF

$$I_{P_{i,j}} \geq I_{threshold}(R_{P_{i,j}}) \quad \text{then} \quad P_{i,j} \text{ belongs to an object}$$

otherwise, $P_{i,j}$ belongs to the background

where

$I_{P_{i,j}}$ - pixel intensity at row, column location (i,j)
$R_{P_{i,j}}$ - pixel range at row, column location (i,j)
$I_{threshold}(\ )$ - intensity threshold function

The program uses a table lookup with the range value indexing it to compute the intensity threshold function. The entries in the table are generated by experimentally determining the intensity

threshold at various range locations and then interpolating and extrapolating to generate the rest of the entries at remaining range locations.

4.2.3 RANGE VARIATION DETECTION

After the objects are detected by the intensity threshold technique, the objects are either verified or rejected using the range variation detection technique. The range variation detection technique tries to grow an arbitrary selected intensity detected object pixel, core pixel, into an object. The 'growth' process clusters neighboring pixels using the range data. The process is initiated by testing the core pixel and each of its neighboring pixels for inclusion in to the object core (object cores are grown into objects). All pixels which are added to the object core are likewise tested. When no added pixels remain to be tested, the growth is complete. The details of this algorithm are presented in this section.

Consider a portion of a laser radar image array as portrayed in Figure 4.2.3-1:

i-1	8	1	2
i	7	0	3
i+1	6	5	4
	j-1	j	j+1

FIGURE 4.2.3-1. Object Formation

Each such portion of the array is termed a neighborhood of the center pixel p(i,j). The center pixel is referred to as a test pixel. The objective of the object formation algorithm is to determine those pixels in the neighborhood which lie on the same object as the test pixel.

Each neighbor pixel is first checked to see if it has already been added to another object. If it has, the neighbor pixel is ignored. Otherwise, it is tested for inclusion as part of the same object as the test pixel. The test applied depends on whether the two pixels share a common row, column, or diagonal. These cases are numbered 1 through 8 as shown in Figure 4.2.3-1. In each case, the range difference is computed and the difference is compared to the appropriate threshold. The test given for each of the cases are:

$$\textit{CASE } 1: \quad \Delta x_{i-1,j} = |x_{i-1,j} - x_{i,j}| \le \Delta R_{row\_threshold}$$
$$\textit{CASE } 2: \quad \Delta x_{i-1,j+1} = |x_{i-1,j+1} - x_{i,j}| \le \Delta R_{row\_threshold}$$
$$\textit{CASE } 3: \quad \Delta x_{i,j+1} = |x_{i,j+1} - x_{i,j}| \le \Delta R_{col\_threshold}$$
$$\textit{CASE } 4: \quad \Delta x_{i+1,j+1} = |x_{i+1,j+1} - x_{i,j}| \le \Delta R_{row\_threshold}$$
$$\textit{CASE } 5: \quad \Delta x_{i+1,j} = |x_{i+1,j} - x_{i,j}| \le \Delta R_{row\_threshold}$$
$$\textit{CASE } 6: \quad \Delta x_{i+1,j-1} = |x_{i+1,j-1} - x_{i,j}| \le \Delta R_{row\_threshold}$$
$$\textit{CASE } 7: \quad \Delta x_{i,j-1} = |x_{i,j-1} - x_{i,j}| \le \Delta R_{col\_threshold}$$
$$\textit{CASE } 8: \quad \Delta x_{i-1,j-1} = |x_{i-1,j-1} - x_{i,j}| \le \Delta R_{row\_threshold}$$

The determination of the thresholds $\Delta R_{row\_threshold}$ and $\Delta R_{col\_threshold}$ depends on the laser radar parameters and can be determined heuristically.

For each object created, a table of characteristics is filled out. Then every time a pixel is added to the object, the table is updated. This table is used in the subsequential processing of the object.

4.3 ORIENTATION

4.3.1 INTRODUCTION

If areas of constant gradients (planar surfaces) can be identified from the measured gradients of the laser data then the objects yaw, pitch and roll angles can be determined. The orientation procedure assumes that an object has a primary plane and that the orientation of the object is such that the orientation of the primary plane is

the orientation of the object. The orientation procedure is a four step process as shown by Figure 4.3.1-1.

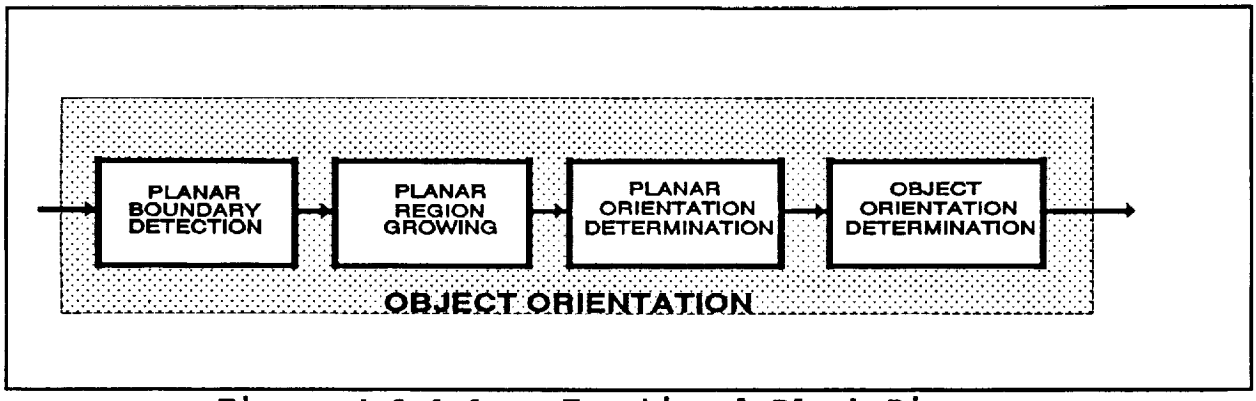

Figure 4.3.1-1. Functional Block Diagram

Once an object has been detected and defined, the orientation process takes over. The orientation process starts by doing an edge detection of the detected object. This step of the process tries to determine planar region boundaries. Once these boundaries are determined, planar regions are grown. Planar regions are define as regions in which the gradients are constant or near to being constant with a reasonable level of variance. For each planar region, the gradient values are entered into the primary plane orientation formulas and the plane's orientation is determined. Only for the primary plane will the primary plane orientation equations yield the correct object orientation. Once all the object planes' orientations are calculated, the size of the planes can be determined. This information is used to resolved the primary plane from the rest of the planes. Once the primary plane orientation is known, then the object orientation is also known.

4.3.2 DEFINITION

When the object is in an unrotated state (yaw, pitch and roll are zero degrees), the primary plane is situated such that the sensor coordinated system's (SCS) x value is equal to a constant and that the longest axis of this plane is parallel the SCS y axis. When the object is rotated, the rotation occurs about the object center of mass. A local coordinate system, called the object coordinate

system (OCS) is defined as a translated SCS with origin at the object center of mass. Figure 4.3.2-1 illustrates this coordinate system. When the object is rotated, the order of rotation will be

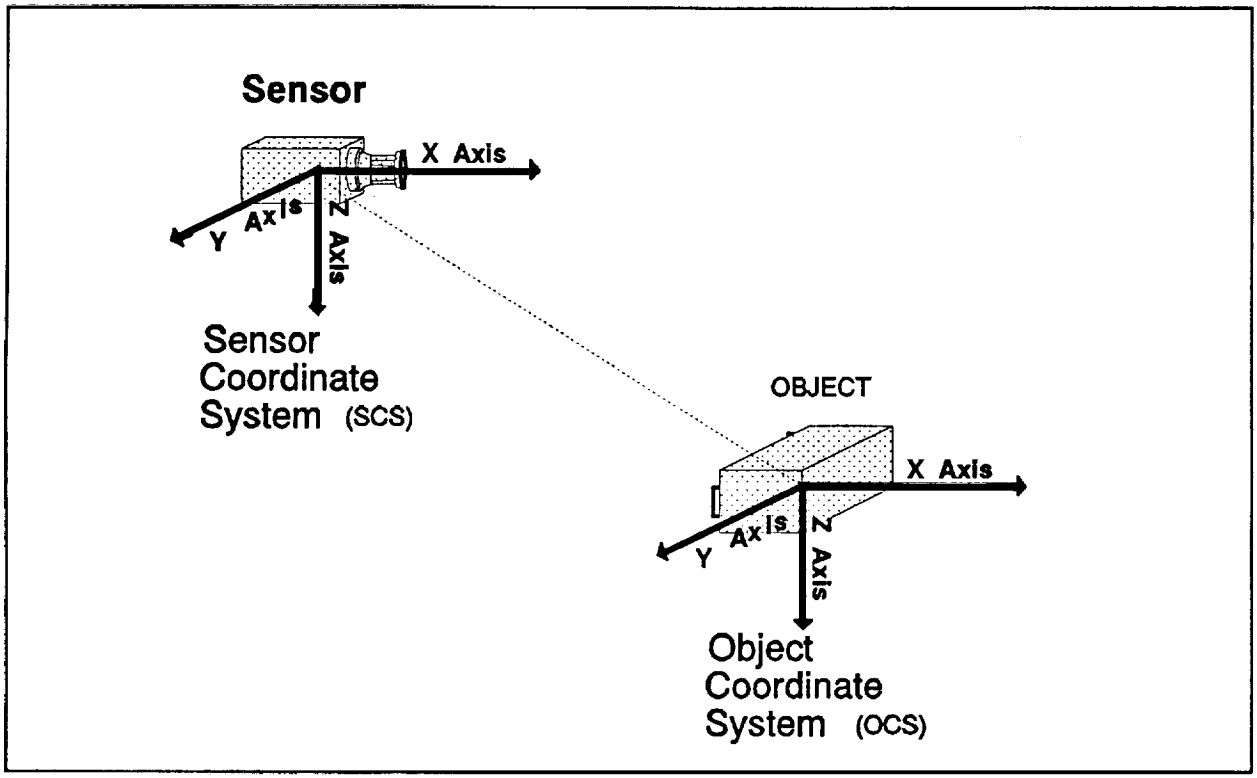

Figure 4.3.2-1. Coordinate System Definition

defined to be in the order of yaw, then pitch and then roll. Yaw, pitch and roll angles are defined to be the clockwise rotation about the z axis, y axis and x axis respectively. This is illustrated by Figure 4.3.2-2. To limit processing time, a current requirement of the algorithm is that the primary plane must always be in the field of view.

4.3.3 PLANAR BOUNDARY DETECTION

The object orientation process starts by first determining locations of planar region boundaries interior to each of the detected objects. To accomplish this, the algorithm does an edge detection. For this edge detection process, the algorithm scans the object, column by column and then row by row, searching for global minimum range values. Each column and row of the object's

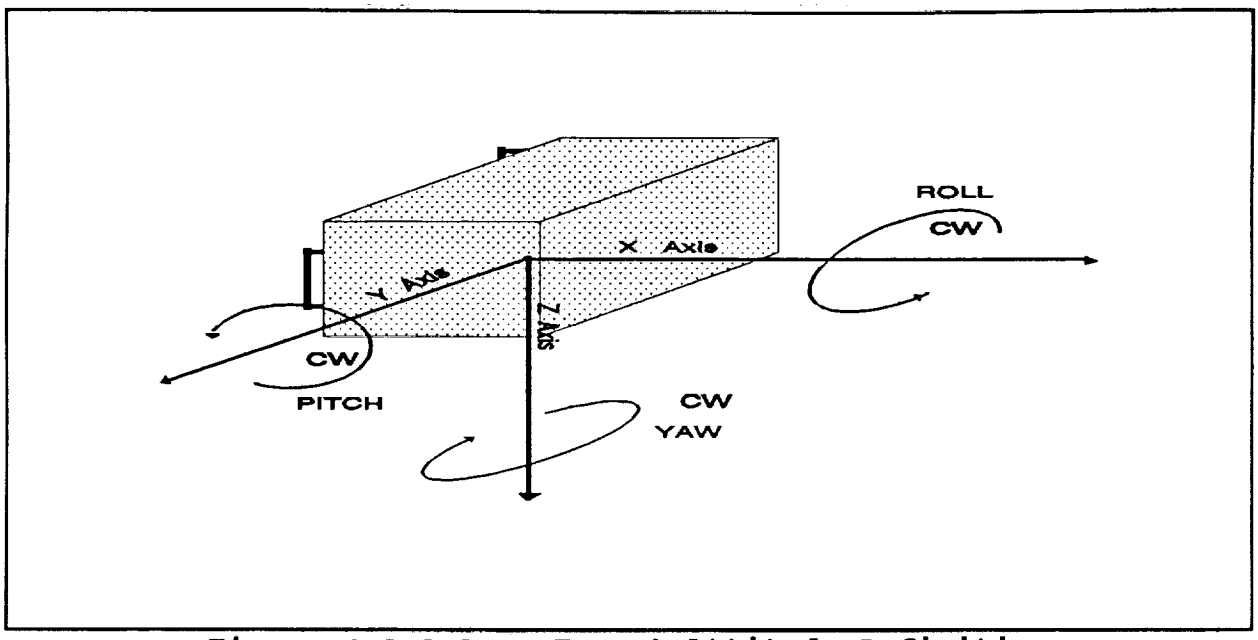

Figure 4.3.2-2. Target Attitude Definition

image is allowed to have only one global minimum range value. Once a global minimum ranged value is found, the pixel at that location is labeled as such. A distinction is made in the labeling so as to indicate either a row global minimum, a column global minimum or both a column and row global minimum. To simplify algorithm complexity and ultimately to reduced processing time, the algorithm searches only for global extremes instead of local extremes. The benefit of this is to eliminated the need for edge thinning. Edge thinning is used to eliminate unwanted edges cause by sensor behavior or error and the unwanted variations in object surface reflectively. Unfortunately, a drawback in searching only for global minimum, is that there might exist pixels which should have been labeled as edge pixels but was not. This problem is eliminated in the planar region growing step of the orientation procedure.

4.3.4 PLANAR REGION GROWING

Once the planar boundaries are determined, the planar regions are grown. The planar region growing portion of the algorithm starts by picking an unlabeled pixel interior to the object. This pixel is called the test pixel. All pixels adjacent to this test pixel

are then compared against this pixel. If the adjacent pixel has not been assigned to a planar region and if it is not a planar boundary pixel, then the planar growing conditions has passed and the adjacent pixel is assigned to the same planar region as the test pixel. Once all neighboring pixels are tested, the same procedure is applied to each of the newly assigned planar region pixels. The testing procedure repeats continually until no new pixels can be added to the planar region. When this occurs, the current planar region is finished growing and a new planar region is started. A planar region is started by picking a new unlabeled, unassigned pixel and then applying the same planar growing procedures. The planar region growing process is concluded when all pixels in the object have either been assigned to a planar region or have been labeled a boundary pixel.

When a plane grows into a boundary region, the algorithm is robust enough not to rely solely on edge pixel labeling. Missing edge labeled pixels might exist in the boundary. The algorithm uses the heuristic of a plane boundary to generate a decision tree for determining where a plane boundary might be.

Once all of the object's planar regions are determined, tests are performed so as to select the planes of acceptable size. Only planes of adequate size are used in next step, plane orientation calculation, of the orientation procedure.

4.3.5 PLANAR ORIENTATION DETERMINATION

After planar regions of acceptable sizes are identified, the orientation angles are then calculated for each of these planes. This is done by using the primary plane orientation equations. The primary plane orientation equations are equations which are used to calculated the yaw, pitch and row orientation angles. Only for the primary plane will the primary plane orientation equations yield the correct orientation values. For all other planes, the orientation calculated will be off by the angular difference

between the planar region of interest and the primary plane. The primary plane orientation equation for yaw (Ψ), pitch (θ) and roll (Φ) angles are expressed below:

$$yaw = \psi = -\tan^{-1}\left(\frac{dx}{dy}\right)$$

$$pitch = \theta = \tan^{-1}\left(\frac{dx}{dz}\cos\psi\right)$$

$$roll = \phi = \frac{1}{2}\tan^{-1}\left(\frac{2\cos\theta\,[\cos\psi\sum y'z' - \sin\psi\sum x'z']}{-\sum z^2 + \cos\theta\,[\cos^2\psi\sum y'^2 + \sin^2\psi\sum x'^2 - 2\cos\psi\sin\psi\sum x'y']}\right) \pm \frac{\pi}{2}$$

where

$$\frac{dx}{dy} = \left.\frac{\partial x}{\partial y}\right|_{z=constant}$$

$$\frac{dx}{dz} = \left.\frac{\partial x}{\partial z}\right|_{y=constant}$$

$$x' = x - \bar{x}|_{plane}$$

$$y' = y - \bar{y}|_{plane}$$

$$z' = z - \bar{z}|_{plane}$$

The formula for the roll angle yields two solutions. The correct solution is the solution that yields a larger $\overline{E^2}$ for the following equation:

$$\overline{E^2} = \frac{\cos^2\Phi\sum z'^2 + F_2\sin^2\phi - F_1\sin 2\phi}{N\cos^2\theta}$$

where

$$F_1 = \cos\theta\left(\cos\psi\sum z'y' - \sin\psi\sum z'x'\right)$$

$$F_2 = \cos^2\theta\left(\cos^2\psi\sum y'^2 + \sin^2\psi\sum x'^2 - \sin 2\psi\sum x'y'\right)$$

N = *number of pixels in the plane*

Once this formula is computed, the metric length and width of the planes can then be calculated. The metric length and width are

calculated by

$$Length \approx 2\sqrt{3}\left(\overline{E^2}\right)_{max}^{1/2}$$

$$Width \approx 2\sqrt{3}\left(\overline{E^2}\right)_{min}^{1/2}$$

To utilize these equations, the pixels in the plane must be transformed and remapped from the sensor spherical (R,α,β) coordinates into a quantized cartesian (x,y,z) coordinates. When the pixels are transformed, the following formulas are used:

$$x = r\cos\alpha\cos\beta$$
$$y = r\sin\alpha\cos\beta$$
$$z = r\sin\beta$$

To calculate α and β, the formulas are given by

$$\alpha = HFOV\left[\frac{1}{2} + \frac{COL\_INDEX-1}{MAX\_COL\_INDEX-1}\right]$$

$$\beta = VFOV\left[\frac{1}{2} + \frac{ROW\_INDEX-1}{MAX\_ROW\_INDEX}\right]$$

where

HFOV is the horizontal field of view of the sensor,
VFOV is the vertical field of view of the sensor,
ROW_INDEX is the row location of the pixel,
COL_INDEX is the column location of the pixel,
MAX_ROW_INDEX is the number of rows in the image,
MAX_COL_INDEX is the number of columns in the image

When the transformation process is done, the x, y and z mean values, the sums and the sums of squares are also then computed.

To compute the yaw and pitch angles, the derivatives $\frac{dx}{dz}$ and $\frac{dx}{dy}$ must be calculated. To do this, the derivatives are approximated

by a first order difference equation:

$$\frac{dx}{dy} = \left.\frac{\partial x}{\partial y}\right|_{z=constant} \approx \frac{\sum_{i=row\_start}^{row\_stop} (x_{i,col\_stop} - x_{i,col\_start})}{\sum_{i=row\_start}^{row\_stop} (y_{i,col\_stop} - y_{i,col\_start})}$$

$$\frac{dx}{dz} = \left.\frac{\partial x}{\partial z}\right|_{y=constant} \approx \frac{\sum_{i=col\_start}^{col\_stop} (x_{i,row\_stop} - x_{i,row\_start})}{\sum_{i=col\_start}^{col\_stop} (z_{i,row\_stop} - z_{i,row\_start})}$$

where

col_start = first column of the object
col_stop = last column of the object
row_start = first row of the object
row_stop = last row of the object

To make these equations valid, the x values must be such that x is a function of only y when the pixel location is varied in the column direction but not in the row direction, and likewise, that x is a function of only z when the pixel location is varied in the row direction but not in the column direction. This is achieved by remapping the x values into a two dimensional 'yz' histogram. The bin size for this histogram is determined by the angular resolution of the sensor. The bin sizes of this histogram for both column and row directions are given by

$$y\_bin\_size = \frac{\bar{R}\, HFOV}{col\_size - 1}$$

$$z\_bin\_size = \frac{\bar{R}\, VFOV}{row\_size - 1}$$

where

$\bar{R}$ = average range value in the plane
VFOV = vertical field of view of the sensor (in radians)
HFOV = horizontal field of view of the sensor (in radians)
row_size = total row size of the live image
col_size = total column size of the live image

The indices into this histogram are given by

$$y\_index = INT\left[\frac{y - y_{min} + \frac{1}{2} y\_bin\_size}{y\_bin\_size}\right] + 1$$

$$z\_index = INT\left[\frac{z - z_{min} + \frac{1}{2} z\_bin\_size}{z\_bin\_size}\right] + 1$$

If two or more x values are mapped into a bin, then the average of the x values are taken for the bin.

4.3.6 OBJECT ORIENTATION DETERMINATION

As mentioned before, the orientation of the primary plane is defined to be the orientation of the object. Once the orientation is calculated for each of the planes, the metric length and width of these planes can then be calculated. As part of the shell model data base, the length and width of the primary plane is given for each of the system target set. The algorithm uses this information along with all of the calculated planes' lengths and widths and tries to find the plane which best matches the primary planes of the prestored target set. The plane that best matches the primary plane of that particular target model is labeled the primary plane of the object for that target model. If none of the planes on the object matches a primary plane for a particular target, the shell is eliminated from any further processing consideration.

4.4 OBJECT CLASSIFICATION

4.4.1 INTRODUCTION

After the objects have been detected and their orientations have been calculated, they are passed onto the object classifier stage for identification. The classifier stage consists of two major functions: target shell generation and target shell matching. This is illustrated in Figure 4.4.1-1.

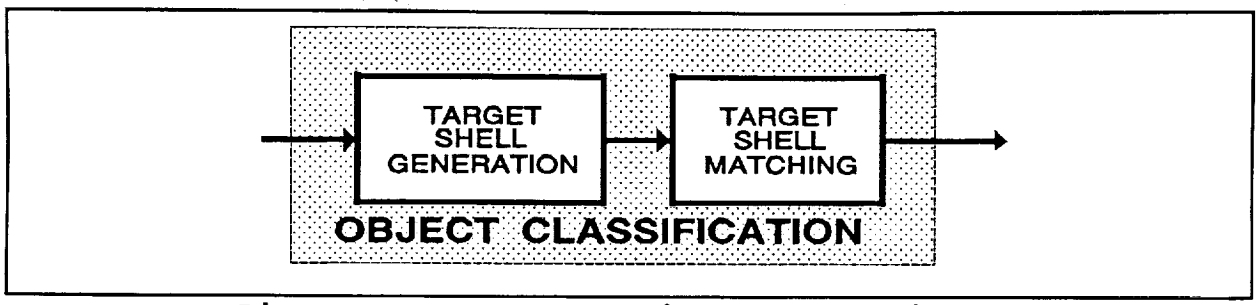

Figure 4.4.1-1. Functional Block Diagram

This stage does the object identification by first forming a hypothesis, preparing a target prediction, and then testing the hypothesis by comparing the prediction to the object. To implement this, the processing stage tries to constructed a set of target model shells for each detected objects from a selective list of prestored target model information. Only the target model shell in which the model's primary plane matches a detected plane for each of the objects will be used. For the other target models, they will be ignored. For each of the detected objects, a comparison is made against a selective list of target candidates for that particular object. The object is identified when a target model best matches the object, and the match is also in acceptable tolerance.

4.4.2 TARGET SHELL GENERATION

Once a target is identified as a possible candidate for a particular object, a shell model is generated. To generate the shell model and to get it ready for the target matching stage, a four step process is performed. First the shell model is scale and oriented to matched that of the object, then the visible surfaces are determined, next a high resolution image array is generated of the shell model and then finally, the elements of the high resolution image array are integrated together to form lower resolution (i.e., sensor resolution) image array. The target shell generation process steps are shown in Figure 4.4.2-1.

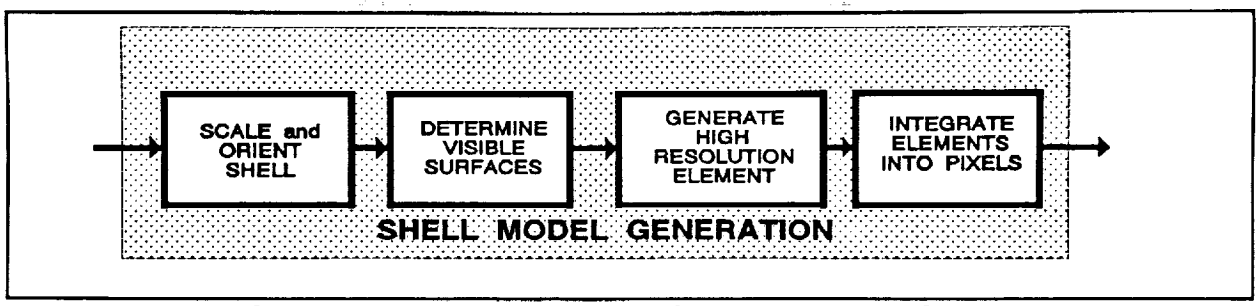

Figure 4.4.2-1. Functional Block Diagram

Each target model consists of a three-dimensional (3-D) surface shell, along with ancillary data. These surface shells, which are polygonal representations of the outer surface of each target, are generated from scale drawings or photographs. They are stored in a Computer Aided Design (CAD) representation in the processor memory. In this format, each surface shell is represented by a number of vertices in a local coordinated system and the four-sided polygons defined by these vertices. The advantage of this is that a complex target model (i.e., on the order of couple hundred polygons) can be stored in a relatively small amount of memory.

In order to perform the match process, the surface shell model must be oriented and scaled to match the orientation and range of the object (i.e., transformed to the sensor coordinate system). First, the visible polygons of the surface shell are determined by calculating the dot product of the vector to the object with the normal of each polygon. The vertices of the visible polygons are rotated in the local coordinate system to match the orientation of the detected object. These vertices are also translated in (X,Y,Z) to match the range, azimuth angle and elevation angle of the object.

Next, a high-resolution range array is created from the target shell model. The image array is index by a row, column address which corresponds to a unique angle, angle location on the object. The formation of this array is started by positioning each of the

shell model vertices into it. The proper bin address is determined by the vertices' (X,Y,Z) values. To fill the array, a fine grid of squares (i.e., approximately 0.1 inches per side in a plane perpendicular to the line of sight for a target approximately six feet from the sensor) is projected onto the visible polygons of the shell. Each pixel of the object has a unique (angle, angle) position and is associated with the grid element closest to that position. To simulate the manner in which the sensor operates, the range value (i.e., the distance from the sensor to the grid element) of the grid element associated with a pixel is averaged with the range values of grid elements in the immediate neighborhood of the grid element to be average. The number of grid elements in the average is determined by the size of the sensor pixel, thus nine grid elements might be averaged together to form a sensor pixel. This averaging process simulates the way the laser radar produces a range return; when the laser radar scans two surfaces, the range return will be the average of the two surfaces, weighted by the ratio of the areas scanned. Averaging the high-resolution grid elements simulates the averaging effect of the laser radar since the range values of a sensor-resolution grid element is maybe from more than one polygon. If portions of two visible polygons lie along the same line of sight they will have certain high-resolution bins in common. In this case, the smaller range value is chosen since this corresponds to the surface actually visible from the fixed origin. In this manner, edges, surfaces, partial pixels (pixels that are averages of returns from two or more surfaces) are predicted to occur at precise locations in space.

4.4.3 TARGET SHELL MATCHING

After the high-resolution bins have been integrated to form sensor-resolution bins, the surface shell array is registered with the object array. The left and right boundaries of both the surface shell array and the object array are compared by a weighted averaging technique to register the two arrays in the azimuth

direction. After registering the two arrays in the azimuth direction they are registered int elevation direction to produce the initial overlay point.

Since the object and the surface shell are now represented in the same reference frame they can be compared directly. A normalized difference algorithm sums the weighted variances between the shell and the object features, divided by the expected variances, to yield a test difference (TD). If the expected variances are correctly estimated, a correct object/surface shell match will produce a TD approximately equal to 100, while an object/surface shell mismatch will produce a TD much less than 100.

The TD calculation has two components, 3-D shape and silhouette. The shape region is defined as all bins with both object and surface shell present. An estimate of the shape similarity is measured by calculating the variance between the range values of the surface shell and the range values of the object on a bin-by-bin basis. The expected range variance is a function of the scanner pointing-angle accuracy, the scanner range precision, and the object orientation. The silhouette region is defined as all bins with either object or surface shell present, but not both. Surface shell/object matches will generally have small silhouette regions, while mismatches will generally produce large silhouette regions. A technique has been implemented for grouping bins in the silhouette region. Groups that have a large number of bins, signify a surface shell/object mismatch. The expected silhouette variance is a function of the expected pointing angle accuracy. The two variance measurements are weighted by the number of pixels in each region to produce the TD Value.

Once the TD value for the target shell model is calculated, the value is retained. After all of the selective target models are processed and the TD values are calculated for each detected object, they are compared. The model with the highest TD value is

selected as the model with the best match. If the TD value for this model has a acceptable match level, then model target type is declared to be the identity of the object in question.

4.5 OBJECT TRACKING

The next and final processing stage of the algorithm is the object tracking stage. Once a object is detected and identified (i.e., type of target or object blob) it is then tracked. If the object is identified in more than one image frame, then the velocity of the object is predicted. The tracker consist of four functional stage as shown by the block diagram in Figure 4.5-1.

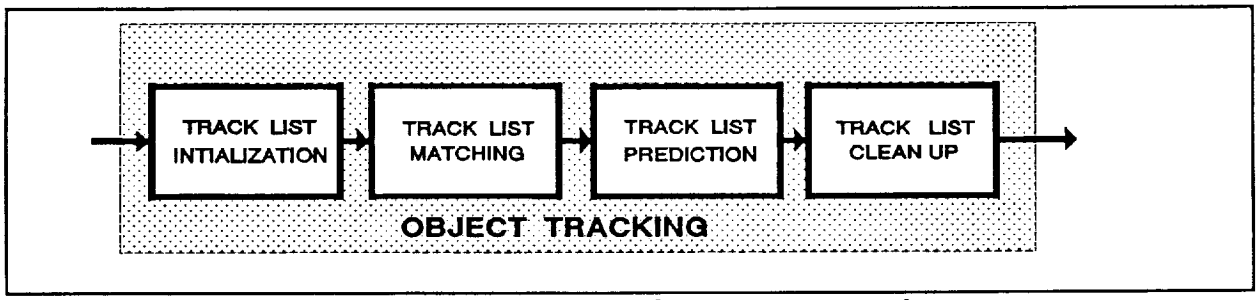

Figure 4.5-1. Functional Block Diagram

The tracker is based around a track data base manager. When the tracker receives an object, it is then assigned to the track data base manager for disposition. The track data base is a double link list of records. Each record consists of many data fields describing an object. One record exists for each object. The track data base manager compares the received object against all the track records and their current predicted locations contained within the data base. If a match exists, then the track record is updated and the velocity is estimated. Otherwise, a new record is created and added to the track list data base.

At the end of a image cycle, the track data base manager updates the track list data base. If any record was not matched in a user specified number of image cycles, then it is removed from the data base track list. Otherwise, the predicted object location field for the track record is updated. The object location field of the

track record is used for track record comparisons. Kalman filtering techniques were investigated, but until better sensor statistics are available, a more non-statistical approach will be used. This approach uses the object's current centroid location and tries to make prediction based upon the object's current velocity and error uncertainties. Geometrically this approach can be thought of as enclosing the centroid within an ellipsoid. The volume within the ellipsoid is the location with uncertainty that the object is expected to be in the next image cycle. The axes of the ellipsoid are determined by the x,y and z velocities multiplied by an error weighting constant. The placement of the foci of the ellipse favors the direction in which the object is moving. For each cycle in which the record is matched, the uncertainty in the object location is reduced and, therefore, the error weighting constant can also be decreased. That is, the more the object is matched the more elliptical the uncertainty region is.

5.0 SAMPLE RUN

The final real time 3-D Laser Radar Vision system was tested on radar imagery obtained from the Odetics 3-D Mapper which is part of the EVA Retriever robot. This imagery consists of both radar range and intensity reflectance. The specification for this image data is given in Table 5.0-1 and by Figure 5.0-1.

Field of view:	60 deg. horizontal x 60 deg. vertical
Frame format:	128 x 128 pixels raster scan
Frame rate:	835 msec/frame, continuous cycling
Range resolution:	1.44 in. (8-bit) or 0.72 n. (9-bit)
Ambiguity interval:	30.74 ft.
Minimum range:	1.5 ft.
Laser:	CW diode laser 820 nm. 0-50mW output
Video:	8/7 bit reflectance, logarithmic scale

Table 5.0-1. Odetics Laser Specification

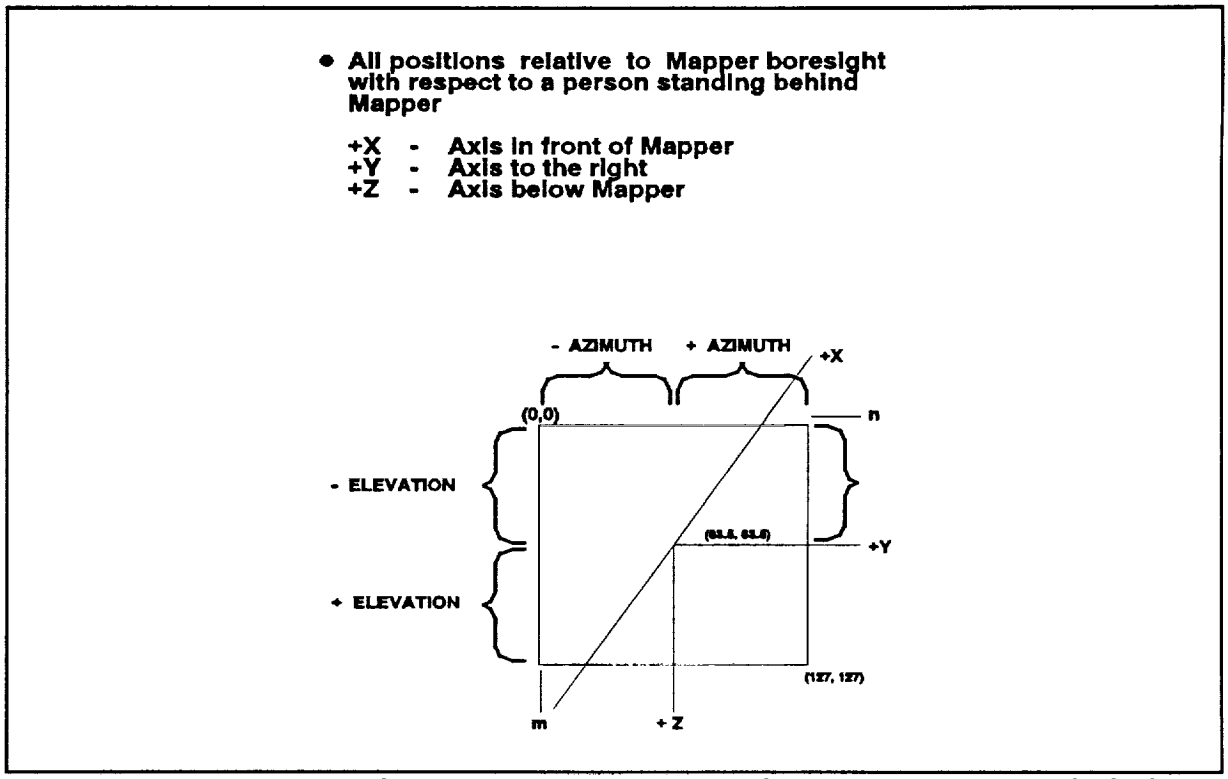

Figure 5.0-1. Odetics 3-D Mapper Coordinate System Definition

An example of an Odetics 3-D Mapper test image of an EVA wench is illustrated by Figure 5.0-2. The left side of the image is reflectance and the right side is range.

ORIGINAL PAGE
BLACK AND WHITE PHOTOGRAPH

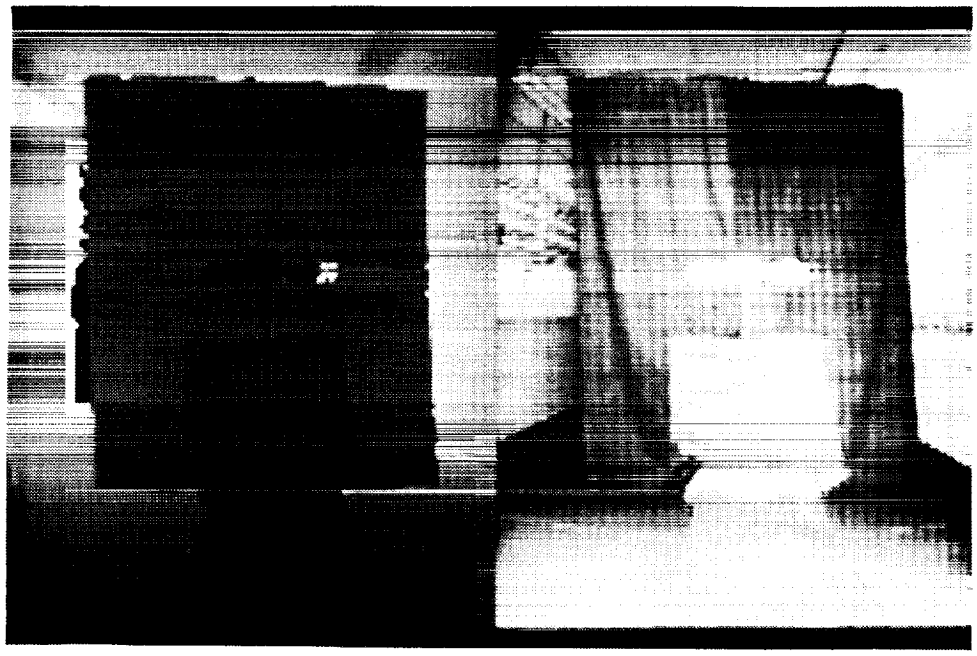

Figure 5.0-2. Odetics Range / Reflectance Imagery of an Eva Wrench

An example of output imagery of the object detection stage of the algorithm is shown in Figure 5.0-3.

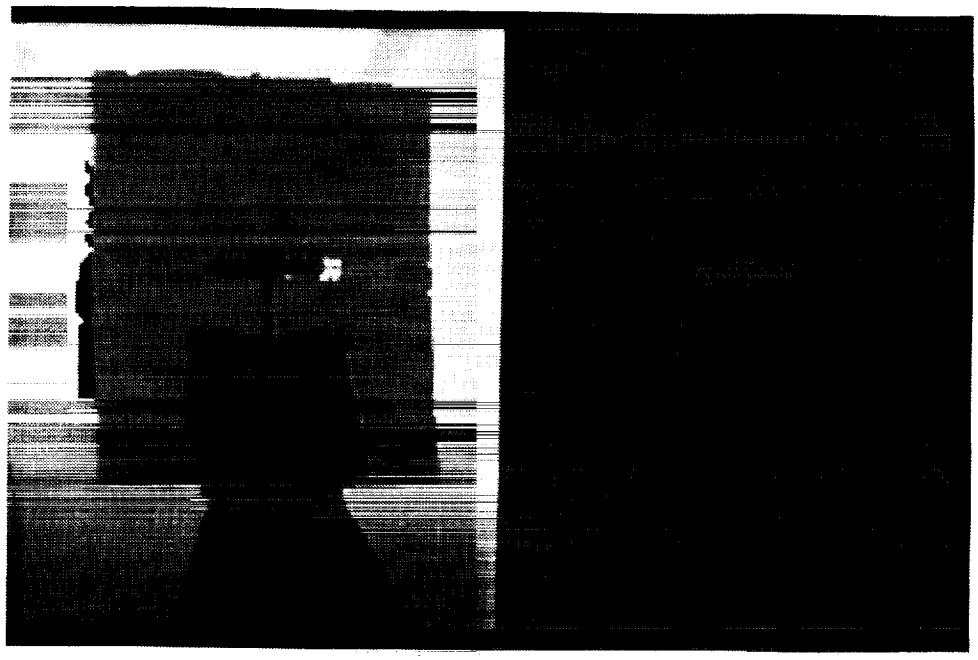

Figure 5.0-3. Odetics Detected Objects / Range Imagery of an Eva Wrench

6.0 RECOMMENDATIONS

The system, as delivered, provides the basic capabilities required by the program. It also provides a structure that will readily permit enhancements to be incorporated. The following is a list of recommended enhancements:

IMPROVED 3D ORIENTATION

Improvements in the 3D orientation software will provide the single greatest improvement in system performance. Both the selection of the appropriate models and the actual comparison with sensor data is greatly influenced by the accuracy obtained from the software that determine the object's orientation and measurements. The routines that will provide the greatest performance improvement are edge detection and plane growing.

IMPROVED OPERATOR INTERFACE

There are a number of operator features that would be highly desirable for the current development environment. These improvements include the ability to display intermediate results such as the object shells and matching results. Images also could be displayed at a higher resolution than the current 320 x 200.

ADDING OF MODELS

The current method of adding target models is operator intensive. A more user friendly method of adding models and displaying model shells is desirable. The existing set of models should be expanded to include a wider variety of targets of interest.

PROCESSING SPEED-UP

Significant improvements can be achieved in the speed of processing the laser radar data. Changes in the data formats will permit more rapid transfer from one transputer to another. The software that allocates detected objects can be modified to permit better utilization of the available transputers resources.

ADAPT TO NEW SENSOR

The Odetics 3D Mapper has limitations that will be eliminated with the new sensor currently being developed. The software should be modified to take advantage of the full range of capability resident in the new sensor. Included in the new capabilities will be higher resolution images, wider field of view and much more flexibility in sensor control.