NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN INEXPENSIVE REAL-TIME FLIGHT SIMULATOR FOR THE UNITED STATES MARINE CORPS' AIRBORNE REMOTELY OPERATED DEVICE

by

Steven Paul Martinson

June 1988

Thesis Advisor: Harold A. Titus

REPORT DOCUMENTATION PAGE

. REPORT SECURITY CLASSIFICATION JNCLASSIFIED	1b RESTRICTIVE MARKINGS
SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
. DECLASSIFICATION / DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
. NAME OF PERFORMING ORGANIZATION aval Postgraduate School 6b OFFICE SYMBOL (If applicable) 62	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
. ADDRESS (City, State, and ZIP Code) onterey, California 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
. NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

. TITLE (Include Security Classification) AN INEXPENSIVE REAL-TIME FLIGHT SIMULATOR FOR THE UNITED STATES MARINE CORPS' AIRBORNE REMOTELY OPERATED DEVICE

2. PERSONAL AUTHOR(S)
MARTINSON, Steven Paul

3a. TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1988 June	54

6. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

7. COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	flight simulator; RPV

9. ABSTRACT (Continue on reverse if necessary and identify by block number)

A flight simulator is developed for the Airborne Remotely Operated Device used by the United States Marine Corps. Real-time interactive simulation is performed on a high speed graphics workstation. Accurately modeled dynamics are incorporated to reflect actual vehicle flight. The resulting system gives an operator the on board impression of flying through three dimensional terrain. This will provide realistic flight training at a fraction of the cost of a commercial simulator.

0 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
2a. NAME OF RESPONSIBLE INDIVIDUAL Harold A Titus	22b TELEPHONE (Include Area Code) 408-646-2560 22c. OFFICE SYMBOL 62Ts

Approved for public release; distribution is unlimited

An Inexpensive Real-time Flight Simulator for the United States Marine Corps' Airborne Remotely Operated Device

by

Steven Paul Martinson
Captain, United States Marine Corps
B.S.I.E. North Dakota State University, 1980

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

June 1988

ABSTRACT

A flight simulator is developed for the Airborne Remotely Operated Device used by the United States Marine Corps. Real-time interactive simulation is performed on a high speed graphics workstation. Accurately modelled dynamics are incorporated to reflect actual vehicle flight. The resulting system gives an operator the on board impression of flying through three dimensional terrain. This will provide realistic flight training at a fraction of the cost of a commercial simulator.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I would like to thank Professor H.A. Titus, LT J.L. Cunningham, USN, CAPT T. Rich, USMC, 1stLT D.R. Pratt, USMC and LTJG H. Jamali, Moroccan Royal Navy for their friendship and support.

I. INTRODUCTION

A. JUSTIFICATION

New technology and equipment can have a dramatic effect on the way we fight but only if it is used properly. To ensure proper use, the most effective concepts of employment must be found and those who will fight with this new equipment must be properly trained in its use.

When the equipment is expensive or when safety becomes the primary concern, adequate training is often considered secondary to cost or injury prevention. However, the military must have effective, realistic training before they are tasked to perform a mission.

Ref. 1 is a dynamic simulation model for a remotely piloted vehicle (RPV) currently under development for the United States Marine Corps (USMC). This reference points out the need for further work in implementing the derived model into a flight simulator. Ref. 2 is a flight simulator developed in the Computer Science Department at the Naval Postgraduate School (NPS). In this study, the need for using more realistic flight dynamics in the simulator is identified.

Flight simulators are available for most current aircraft. When tailored to specific needs these training simulators often cost millions of dollars. Operators use them to learn or review

flight procedures safely and at a fraction of the cost of actual flight time. Used routinely, they provide training environment, decrease long-term costs, increase pilot proficiency, and develop the habits necessary for battlefield success [Ref. 3]. Therefore, combining the above mentioned works into a realistic relatively inexpensive flight simulator is a logical extension and fulfills a need for the USMC.

B. WHY ROBOTICS?

In 1980, the United States Marine Corps sponsored a project to find military robotics and remotely controlled devices applicable to the Fleet Marine Force (FMF) Mission. These devices are to be used by ground combat forces to accomplish their mission of locating, closing with, and destroying the enemy. During this project, as shown in Ref. 4, tele-operated systems demonstrated the potential to increase mission capabilities while possibly lowering the threat to front line Marines. With this in mind, the Marine Corps established the Ground-Air Tele-Robotics Systems (GATERS) program to develop and test these systems. The goal is to develop a reliable, easy to use, tele-robotic vehicle to give Marines enhanced combat capabilities in a hazardous environment while keeping the operator in a safe remote location.

Only Marines in infantry units will be trained to operate these systems. In this way no addition personnel are needed and outside support is kept to a minimum, thereby reducing the

logistics burden of supporting this system at the frontline units.

C. RPV DEVELOPMENT

RPV's have been used in one form or another since the 1890's when cameras were mounted on kites for observation [Ref. 5]. Other early versions were mostly used only as target drones and did not have the capability for providing combat support.

It was not until the last few years that rapid advances in technology have been applied to the design of RPV's. This research and development is proceeding at a remarkable pace for tactical systems during peace time. Within the Marine Corps alone, there are currently several different programs under development.

With the rapid movement from the laboratory to deployment, it is easy for the hardware to get ahead of the tactics and training. RPV's, as with all new weapon systems, force the military to undergo a two-step evolutionary process: first, methods of fighting must be altered in order to exploit the new weapons capabilities to their maximum; second, both active and passive means must be found to limit the increased effectiveness of the enemy's use of these new weapons [Ref. 6].

The Israelis have demonstrated the effectiveness of RPV's in combat. Because of the increased cost and vulnerability of military aircraft, they have developed tactics to hinder the

surface to air (SAM) missile threat. As battlefield scenarios reach higher levels of intensity, more missions will be transferred to RPV's.

II. THE AROD

A. THE MISSION

One of the tele-robotic systems under development by GATERS is the airborne remotely operated device (AROD) shown in Figure 2.1 [Ref. 1].

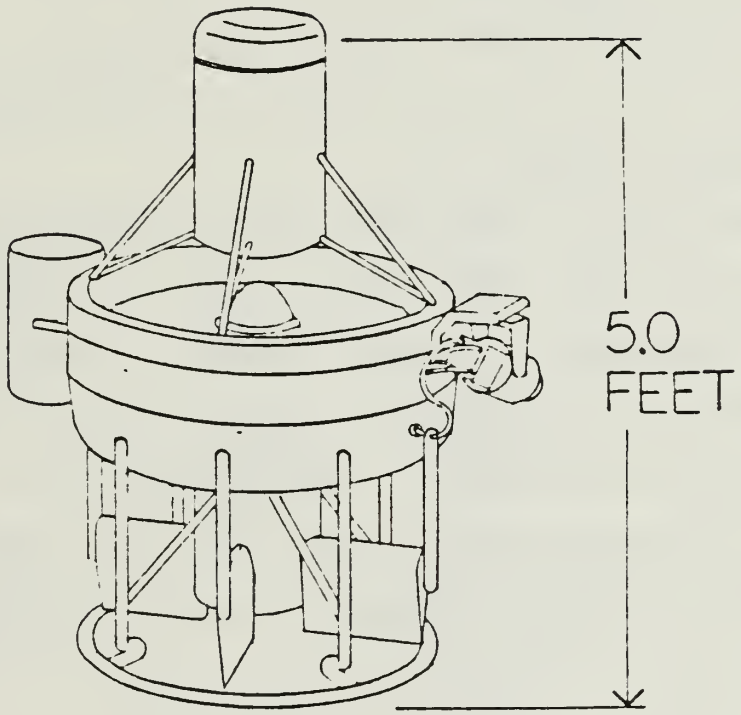

Figure 2.1

The Airborne Remotely Operated Device

The objective of the AROD program as stated in [Ref. 4] is to provide a lightweight, unmanned, fiber-optic tethered, low-altitude flying device that can provide an "over-the-hill" and "around-the-corner" observation capability to the frontline unit commander. Potential AROD applications include, but are not limited to, reconnaissance and surveillance, NBC monitoring and reconnaissance, radio relay, target location/designation, electronic warfare, and mine detection. The purpose of this program is to provide the frontline commander with the capability to perform "over-the-hill" or "around-the-hill" surveillance quickly without risking Marine lives.

B. GROUND CONTROL

This RPV is controlled from a portable hand-carried ground station through a fiber-optic link by a trained operator. This operating station, shown in Figure 2.2 [Ref. 4], consists of the necessary flight controls, camera controls, and a video display to control the vehicle. While watching the video display and using joysticks for attitude and camera control, the operator flies the vehicle as if he is actually on board.

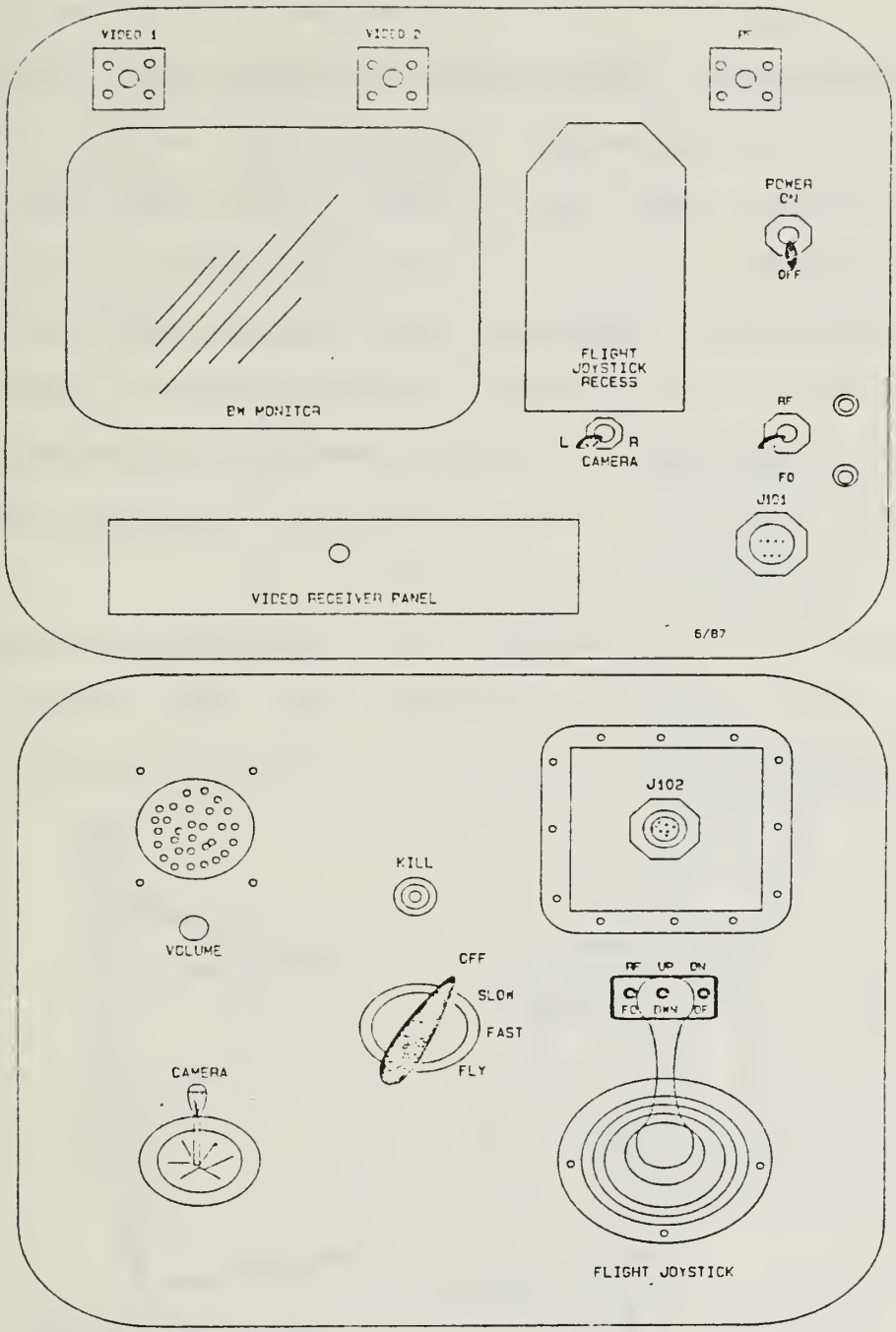

Figure 2.2

The Ground Control Station

C. THE CONTROL SURFACES

AROD's flight characteristics are much different from those of the more common fixed wing RPV. It is in many ways similar to a helicopter with movement possible in any of the three angular and three linear directions. There is the added advantage of a stationary hover.

A lightweight two cycle, two stroke gasoline engine is connected directly to a three-bladed propeller. This propeller turning at high speed develops a downwash which produces lift. As rpms are increased, lift is increased. Therefore, the throttle control is also the elevation control.

Located in the downwash are the servo operated control surfaces shown in Figure 2.3 [Ref. 7]. Any movement in these control surfaces produce changes in vehicle attitude.

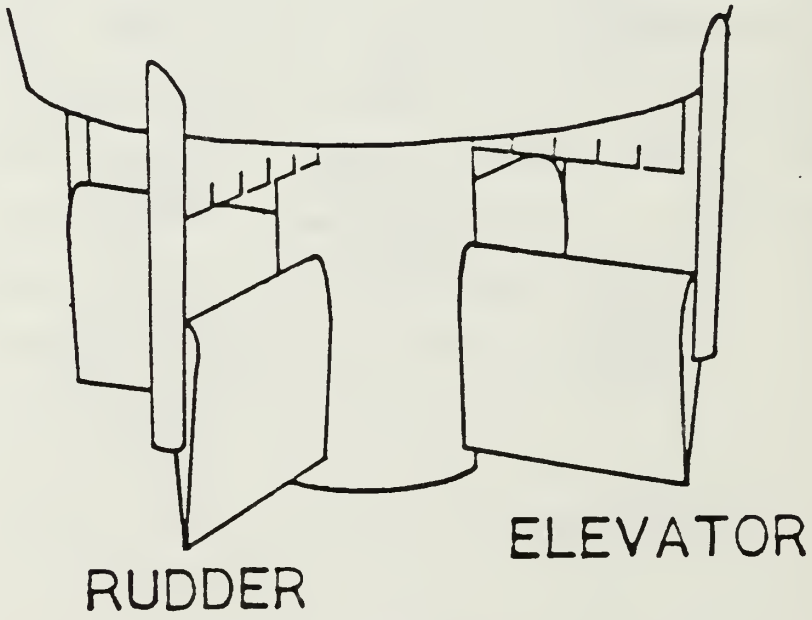

Figure 2.3

The Control Surfaces

When the pair of elevators are deflected into the downwash, the vehicle pitches forward or backward. This vehicle rotation directs the downwash away from vertical, causing translational movement and some loss of lift. This downwash provides forward and backward speeds of up to 30 knots. When the pair of rudders are deflected, the vehicle yaws from side to side giving lateral speeds also of up to 30 knots. These four control surfaces also work together as ailerons for roll control. Roll changes the vehicle heading, which determines the direction the camera is pointing.

III. THE FOG-M FLIGHT SIMULATOR

A. THE OBJECTIVE

The abstract in Ref. 2 describes this project as "a prototype flight simulator for the Fiber-Optically Guided Missile (FOG-M). This prototype demonstrates the practicability and feasibility of using low-cost graphics hardware to produce acceptable simulation of flight over terrain generated from Defense Mapping Agency (DMA) digital terrain elevation database (DTED). The flight simulator displays a dynamic, three-dimensional, out-the-window view of the terrain in real-time while responding to operator control inputs. The total system cost (software and hardware) of the simulator is an order magnitude less than most flight simulation systems in current use."

B. THE HARDWARE

This project used a high-performance, high-resolution Silicon Graphics, Incorporated IRIS-2400 Turbo graphics workstation. The (IRIS) system is shown in Figure 3.1 [Ref. 8].

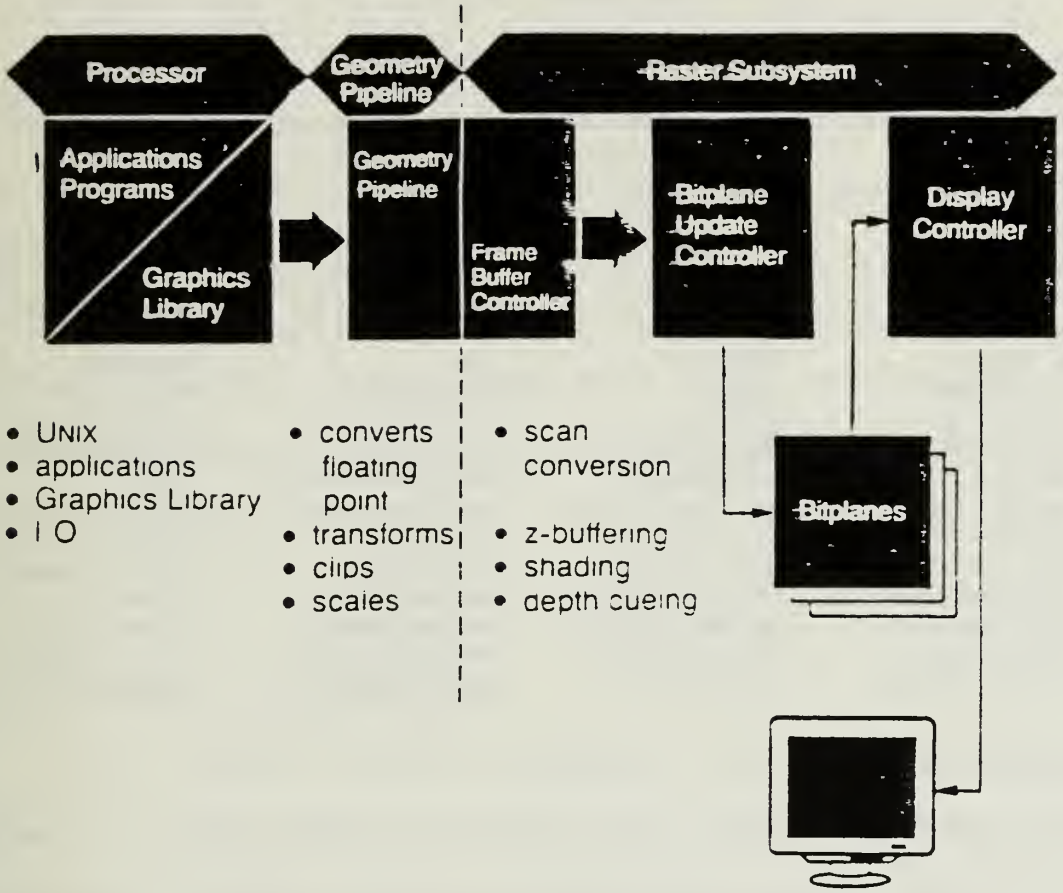

Figure 3.1

The IRIS System

C. THE SOFTWARE

The modified FOG-M flight simulation software consists of the files shown in Table 3.1. These files were written in the C programming language common to most computer graphics applications. The original files were used to develop the AROD files which are available in the NPS IRIS graphics library.

TABLE 3.1

THE AROD FILES

a.out	do_boundary.o	init_iris.c	npoly_orient.c
add_vertex.c	edit_indbox.c	init_iris.o	npoly_orient.o
arod	edit_indbox.o	interp_elev.c	prelaunch.c
arod.c	edit_navbox.c	interp_elev.o	prelaunch.o
arod.h	edit_navbox.o	letter.c	randnum.c
arod.o	exit_arod.c	letter.o	randnum.o
billboard.c	exit_arod.o	lightorient.c	readcontrols.c
billboard.o	explosion.c	lightorient.o	readcontrols.o
buildterrain.c	explosion.o	line_inter2.c	readcontrols.tmp
buildterrain.o	filelist	line_inter2.o	readdata.c
changeum	files.h	lpt	readdata.o
colorramp.c	fpsm.c	makefile	readin.data
colorramp.o	fpsm.o	makeindbox.c	sort_array.c
compass.c	gammaramp.c	makeindbox.o	sort_array.o
compass.o	gammaramp.o	makeinstrbox.c	trig.c
default.poly	gnd_level.c	makeinstrbox.o	trig.o
disp_terrain.c	gnd_level.o	makemap.c	up_look_pos.c
disp_terrain.o	grid_level.c	makemap.o	up_look_pos.o
dispfpsbox.c	in_this_poly.c	makenavbox.c	up_msl_pos.c
dispfpsbox.o	in_this_poly.o	makenavbox.o	up_msl_pos.o
dist_to_los.c	init_ctrls.c	makescreens.c	view_bounds.c
do_boundary.c	init_ctrls.o	makescreens.o	view_bounds.o

D. TERRAIN DISPLAY

The terrain used is DMA digital terrain elevation database for Fort Hunter-Ligget, California, with elevation points spaced twelve and one-half meters apart. However, because of the frame rate restrictions described in Ref. 2, elevation points spaced one hundred meters apart are used.

The colors on the two-dimensional terrain map in Ref. 2 represent a vegetation code. Areas with little or no vegetation are colored brown and heavily vegetated areas are colored green. In the actual flight simulation, three dimensional elevation-keyed shading was used with lighter colors used for higher elevations and darker colors used for lower elevations.

E. FUTURE WORK

The original FOG-M flight simulator did not contain the actual flight dynamics of the missile. A rough approximation was used with the option left open for a more accurate model to be input later.

The system was slowed to a less than desirable three frames per second due to the large number of computations involved. This was adequate for the above stated objective but the desired motion picture speed of 24 frames per second is the goal for a flight simulator.

F. SIMULATOR IMPROVEMENTS

After adding vehicle dynamics and other characteristics unique to AROD, the frame rate was slowed to approximately one frame per second. This was unsatisfactory and does not give the impression of smooth flight. The next objective is to increase the simulation speed.

G. NEW HARDWARE

The first step to improve the frame update speed was to move the simulation system to a faster machine. The Naval Postgraduate School Computer Science Department's Graphics and Video Laboratory recently purchased the IRIS-4D series workstation. Ref. 9 adapts the files to the new system and approximately doubles the simulator frame rate. This is done

while at the same time using a larger higher resolution screen. Improvements continue on these revised files listed under WORK in the graphics library.

H. DATA FILE FORMAT

Ref. 10 has several solutions for decreasing the number of computations and the processing time. These are briefly explained below and are incorporated into the AROD simulation.

The Defense Mapping Agency digital terrain elevation data is used as in the previously mentioned simulators [Ref. 2 and 10] for portraying the three-dimensional scene. This data is stored in such a way that it can only be read through the use of nested loops. These loops slow up the data processing time. Therefore, the terrain elevation data was reformatted and stored with scaling and metric conversion calculations previously performed. This allows for much faster reading.

I. TERRAIN POLYGON CONSTRUCTION

The DTED consists a ten kilometer by ten kilometer square which is subdivided into one hundred by one hundred meter sections. Each of these sections consists of two triangles. A three dimensional contour is obtained by coloring these triangular polygons. Every displayed frame is constructed from a collection of filled polygons. To minimize the number of polygons generated, only those actually in the vehicles field-of-

view are constructed. By limiting this to fifty-five degrees, the approximate maximum camera viewing angle, the frame rate is not severely degraded.

IV. THE LINEAR AROD MODEL

A. THE CONTROL THEORY

In this chapter, a discussion of modern control theory is given implementing the linear AROD model developed in Ref. 1. This model is used as the basis for the simulation routine in the next chapter.

AROD is designed as a computer-controlled system. This system design shown in Figure 4.1 utilizes the full potential of computer control by incorporating powerful digital algorithms. Three main reasons for using computer-control are: 1) by using a digital computer, the number of control laws available is greatly increased over analog feedback techniques, 2) digital computer controlled systems can outperform those working in continuous-time, and 3) distortion is reduced by the use of digital filters.

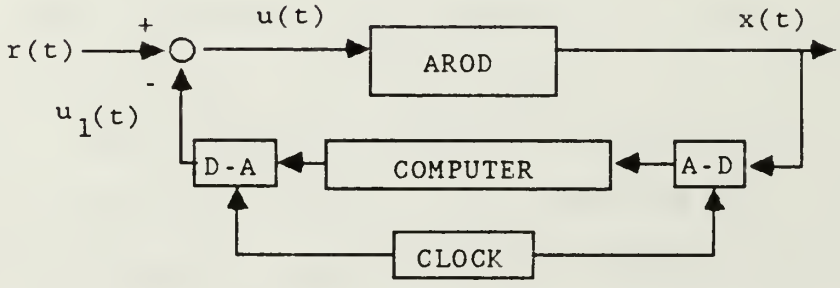

Figure 4.1

The Computer Controlled System

B. THE AROD MODEL

The goal is to design a control process that responds as desired to disturbances and command signals. In this problem feedback is used to maintain stability and drive the system to these commanded input values. The sequence of operations in this control scheme is from [Ref. 11]:

1. Commanded input is sent to the system.
2. The process is excited and the output measured.
3. Wait for a clock pulse.
4. Perform analog to digital conversions.
5. Compute control variables.
6. Update the state of the regulator.
7. Perform digital to analog conversion.
8. Go to step 1.

Step 1:

Commanded input to the system, r(t), is summed with the feedback values $u_1(t)$ to give the process input u(t). In AROD these values are all continuous time and the process consists of the vehicle's servos.

Step 2:

This servo input causes control vane and throttle displacement moving the vehicle as desired. As stated in Ref. 1, the control vane positions are measured as well as information from the three single axis rate gyros, a vertical rate gyro, a magnemeter and a barometric altimeter. These measured state values become the control output y(t).

Steps 3 and 4:

In this system, signals are converted from analog to digital (A-D) for use in the computer and from digital to analog (D-A) for use in the physical process. Sampling of the continuous-time system converts the output signal into a sequence of numbers that are specific state system values separated by a time interval. These discrete signal values are obtained at the sampling times with values in between disregarded. However, this makes the system time dependent. To prevent hidden oscillations and to ensure all information found in the continuous-time system is transferred to the discrete signal, sampling must be at least twice the highest frequency present Eq. (4.1). As explained in [Ref. 1] this is known as the Nyquist rate.

$$f_s = \frac{1}{\Delta T} \geq 2f_H \qquad (4.1)$$

where

f_s is the sampling frequency,

ΔT is the sampling period,

f_H is the highest frequency found in the system.

For more accuracy in actual systems, prefilters are used to block unrepresentable frequencies and noise included high-frequencies. Ref. 12 explained that engineering experience has shown better results in sampled systems when the sampling frequency is equal to 10 times the highest frequency component.

Therefore the sampling frequency given in Ref. 1 for this control problem is 25Hz which gives a ΔT of .04 seconds.

Steps 5 and 6:

After sampling the continuous-time system, the discrete model is found. Using Eq. (4.2) from Ref. 1 the discrete-time model of the continuous-time system is computed and written in state-space form. This form is desirable because needed future predictions can be made from initial conditions and weighted input.

$$x(k+1) = \phi x(k) + \Gamma u_c(k) \qquad (4.2)$$

with

$$\phi = e^{A\Delta T}$$

and

$$\Gamma = \int_0^{\Delta T} e^{As} ds B$$

where

$k = k\Delta T$

$k+1 = k\Delta T + \Delta T$

ΔT is the sampling period,

ϕ is the discrete-time version of the plant matrix A,

Γ is the discrete-time version of the control distribution matrix B,

e is the natural logarithm operator,

s is the Laplace operator,

ds is the derivative with respect to s.

Solving these equations gives the discrete-time system [Ref. 1] found in Table 4.1 and used for computer control in Figure 4.2.

TABLE 4.1

THE DISCRETE TIME SYSTEM

Φ

$$\begin{bmatrix}
1 & 0 & .0400 & -.0030 & -.0167 & 0 & -.0002 & 0 \\
0 & 1 & 0 & .0350 & 0 & .0004 & 0 & 0 \\
0 & .0009 & 1 & -.1466 & -.8217 & -.0015 & -.0134 & 0 \\
0 & 0 & 0 & .9608 & 0 & .0189 & 0 & .0003 \\
0 & 0 & 0 & 0 & .9010 & 0 & .0275 & 0 \\
0 & 0 & 0 & 0 & 0 & .9010 & 0 & .0275 \\
0 & 0 & 0 & 0 & -4.334 & 0 & .4132 & 0 \\
0 & 0 & 0 & 0 & 0 & -4.334 & 0 & .4132
\end{bmatrix}$$

$$\begin{bmatrix}
1 & 0 & .0395 & -.0054 & -.0113 & .0012 & -.0001 & .00001 \\
0 & 1 & .0054 & .0395 & -.0010 & -.0130 & -.00001 & -.0002 \\
0 & 0 & .9636 & -.2667 & -.5530 & .0879 & -.0090 & .0010 \\
0 & 0 & .2681 & .9636 & -.0768 & -.6357 & -.0009 & -.0104 \\
0 & 0 & 0 & 0 & .9010 & 0 & .0275 & 0 \\
0 & 0 & 0 & 0 & 0 & .9010 & 0 & .0275 \\
0 & 0 & 0 & 0 & -4.334 & 0 & .4132 & 0 \\
0 & 0 & 0 & 0 & 0 & -4.334 & 0 & .4132
\end{bmatrix}$$

Γ

$$\begin{bmatrix}
-.0003 & 0 \\
0 & .00001 \\
-.0299 & -.00003 \\
0 & .0007 \\
.0990 & 0 \\
0 & .0990 \\
4.334 & 0 \\
0 & 4.334
\end{bmatrix}$$

$$\begin{bmatrix}
-.0002 & .00001 \\
-.00001 & -.0002 \\
-.0203 & .0016 \\
-.0014 & -.0234 \\
.0990 & 0 \\
0 & .0990 \\
4.334 & 0 \\
0 & 4.334
\end{bmatrix}$$

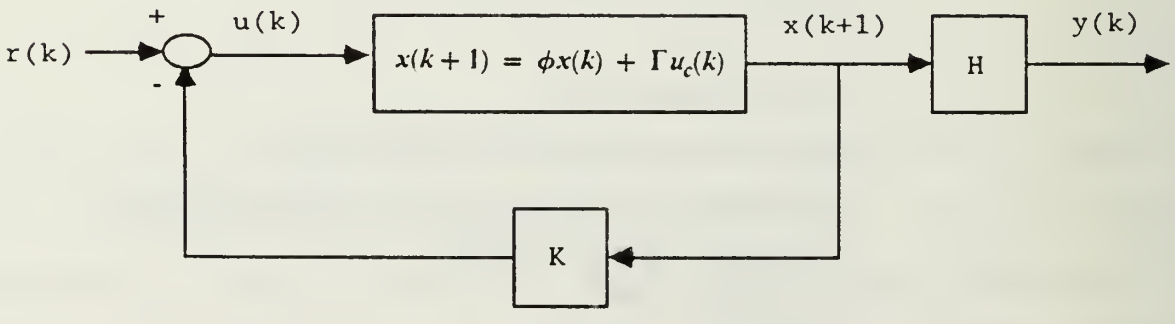

Figure 4.2

The next step is to find the control system variables that dampen out disturbances in minimum time with minimum overshoot. This design solution is known as the optimal regulator. Because this regulator is a multi-input, multi-output system, the problem is not well suited for solving by classical control techniques [Ref 13]. Therefore, the values in the closed loop feedback matrix, K, are found using optimal control methods.

As the name implies, optimal control theory provides the best possible control solutions provided the proper performance criteria are chosen. From Ref. 1 these are:

1. Minimize the transient response time.
2. Minimize the state overshoot.
3. Determine a constant gain schedule, K.
4. Operate within the physical constraints of the system.

For AROD, these criteria are used in the performance measure Eq. (4.3) from Ref. 1:

$$J = \sum_{k=0}^{k_c} [x(k)^t Q x(k) + u_c(k)^t R u_c(k)] \qquad (4.3)$$

where

J is the cost function,
k is the time step index,
k_c is the time step when J converges,
Q is the state weighting matrix,
R is the control cost weighting matrix,
t is the matrix transpose operator,
u_c is the control schedule.

To minimize the control effort required, while staying within the above listed performance criteria, optimal values are chosen for the Q and R matrices. This gives the control schedule $u_c(k)$ for the smallest possible J.

The solution is found by solving the recurrence relation Eq. (4.4) found in Ref. 1.

$$K = [R + \Gamma^t P \Gamma]^{-1} \Gamma^t p \phi \qquad (4.4)$$

and

$$P = M^t P M + K^t R K + H^t Q H$$

and

$$M = \phi - \Gamma K$$

where

K is the constant gain schedule,
H is the measurement matrix.

The weights of the Q and R matrices define the cost function. With this function, the optimal gains, K from Ref. 1 and listed in Table 4.2 are computed. These AROD gains were used as shown in Figure 4.3 and found to be within the constraints of Ref. 1: 1) a settling time of two seconds, 2) less than 10% overshoot, and 3) states within constrained limitations. The optimal gain

matrix converged in 14 iterations for the roll/throttle system in .04 seconds per iteration and 13 iterations for the pitch/yaw system.

TABLE 4.2

THE OPTIMAL GAINS

$$K = \begin{bmatrix} -1.99 & -.603 & -.717 & .274 & 1.75 & .00994 & .140 & .00016 \\ -.585 & 111. & -.205 & 28.9 & .582 & 1.67 & .00962 & .137 \end{bmatrix}$$

$$\begin{bmatrix} -1.61 & -1.61 & -.553 & .320 & 1.05 & -.208 & .127 & -.00264 \\ 1.57 & -1.60 & -.305 & -.566 & .107 & 1.23 & .00096 & .130 \end{bmatrix}$$

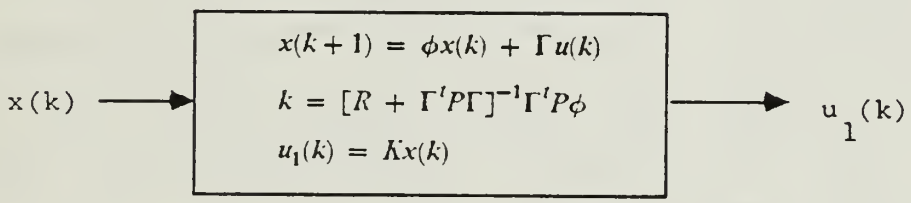

Figure 4.3

The Feedback Matrix K

The H matrix, Figure 4.2, also known as the measurement matrix, isolates the sixteen desired states. These are the angle states, the altitude rate and the control vane displacements and displacement rates. These states Table 4.3, defined in Ref. 1 are needed to control the process.

TABLE 4.3

THE CONTROL STATES

X1 = roll (earth fixed)
X2 = vertical body fixed velocity
X3 = body fixed roll
X4 = engine thrust force
X5 = aileron displacement
X6 = aileron displacement
X7 = aileron velocity
X8 = aileron velocity
X9 = pitch (earth fixed)
X10 = yaw (earth fixed)
X11 = body fixed pitch
X12 = body fixed yaw
X13 = aileron displacement
X14 = aileron displacement
X15 = aileron velocity
X16 = aileron velocity

Step 7:

Once the control variables have been found by the digital computer, they must be converted back to continuous-time for use by the process. This reconstruction is done by zero order hold. In this method, each value from the discrete sequence is held constant until the next sampling instant, resulting in a piecewise constant signal.

Step 8:

As the control values are sent to the summing junction, current state information in the computer is used to determine future state values. This sequence is repeated each clock cycle.

V. SIMULATOR RESPONSE

A. PROCEDURE

For the simulator to approximate the actual AROD response, the computer-controlled model must be incorporated into the graphics routine. In this chapter, the linear model of the AROD flight dynamics from Chapter IV is incorporated into the FOG-M flight simulator.

B. THE FLIGHT CONTROLS

To begin, the simulator needs input controls that match those the AROD operator will use. Commands to this model are given through a control box and mouse instead of the actual ground station joystick control. On this control box, four dials are used. Dial zero is the heading control and causes AROD to roll through the range 0-360 degrees. Dial one is the throttle control which determines the altitude from ground level to 8000 ft. Dial two is the pitch control, this changes the vehicles forward and backward speed from zero to thirty knots. Dial three is the yaw control for lateral speeds also from zero to thirty knots.

C. THE COMMANDED INPUT

In the simulator system, operator commanded input values are read by the C function READCONTROLS, a portion of which is shown beginning with Table 5.1. This function reads the control box every frame cycle to check for changes in the input. These commanded input values are x1c for roll, x2c for throttle, x3c for pitch, and x4c for roll.

TABLE 5.1

INPUT IS READ FROM CONTROL DIALS

```
/*                    +--------------------------------+
                      |                                |
                      |        readcontrols.c          |
                      |                                |
                      +--------------------------------+          */
/* reads the values from the operator's controls (mouse and dials) */


   /*    readcontrol dial input  */

         x1c = (float)getvaluator(DIAL0) /DIRSENS;     /* roll     */
         x2c = (float)(getvaluator(DIAL1)/THROTSENS);  /* throttle */
         x3c = (float)(getvaluator(DIAL2)/PITCHSENS ); /* pitch    */
         x4c = (float)(getvaluator(DIAL3)/YAWSENS );   /* yaw      */
```

D. THE ERROR STATE VECTOR

After reading the controls and adjusting for dial sensitivity the function DYNAM in Table 5.2 is called. The first step in this function is to subtract the four commanded input values from their corresponding AROD state values: roll, throttle, pitch, and yaw. This subtraction from the current state values x[1], x[2],

x[3], and x[4] give the error states ux[1], ux[2], ux[9], and ux[10]. The twelve remaining states, ux[3] through ux[8] and ux[11] through ux[16] are left unchanged. These sixteen states, two eight state subsystems, become the error state vector.

TABLE 5.2

THE ERROR STATE VECTOR

```
dynam(x1c,x2c,x3c,x4c,xk);


ux[1] = x[1]-x1c; /* check for control input;the error state vector */
ux[2] = x[2]-x2c;
ux[9] = x[9]-x3c;
ux[10] = x[10]-x4c;
```

E. THE TRACKING PROBLEM

This error state is used in what in control theory is known as the tracking problem Figure 5.1.

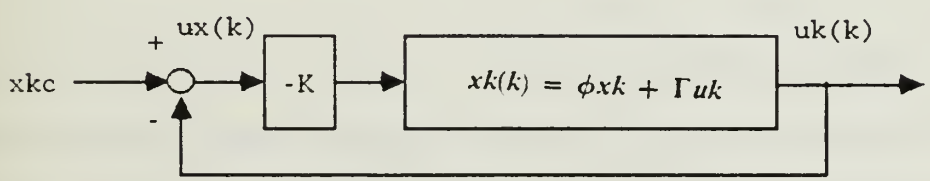

Figure 5.1

Tracking Control Block Diagram

In this context, the objective is to drive the system states x[k] to the control box input values xkc in the minimum amount of time. This is done by multiplying the error state vector ux[k] by the optimal steady-state gain matrix -K. These values are added together in Table 5.3 to give the input matrices u1, u2, u3, and u4.

TABLE 5.3

THE INPUT MATRICES

```
/*      apply steady-state gain schedule to commanded input      */

u1=(1.99*ux[1])+(.603*ux[2])+(.717*x[3])-(.274*x[4])-
   (1.75*x[5])-(.00994*x[6])-(.140*x[7])-(.00016*x[8]);

u2=(.585*ux[1])-(111.0*ux[2])+(.205*x[3])-(28.9*x[4])-(.582*x[5])-
   (1.67*x[6])-(.00962*x[7])-(.137*x[8]);

u3=(1.61*ux[9])+(1.61*ux[10])+(.553*x[11])-(.32*x[12])-(1.05*x[13])+
   (.208*x[14])-(.127*x[15])+(.00264*x[16]);

u4=(-1.57*ux[9])+(1.6*ux[10])+(.305*x[11])+(.566*x[12])-(.107*x[13])-
   (1.23*x[14])-(.00096*x[15])-(.130*x[16]);
```

F. THE CONTROL MATRIX

The control matrix Φ is multiplied by the input matrix uk and added to the product of the plant matrix Γ, and the states xk. As can be seen in Table 5.4, this gives the new state values xk[k] that change with the commanded input.

TABLE 5.4

AROD DISCRETE-TIME STATE SPACE REPRESENTATION

```
    /*        multiply states by discrete time system matrices          */

  xk[1] =x[1]+(.04*x[3])-(.003*x[4])-(.0167*x[5])-(.0002*x[7])-(.0003*u1);
  xk[2] =x[2]+(.035*x[4])+(.0004*x[6])+(.00001*u2);
  xk[3] =(.0009*x[2])+x[3]-(.1466*x[4])-(.8217*x[5])-(.0015*x[6])-
         (.0134*x[7])-(.0299*u1)-(.00003*u2);
  xk[4] =(.9608*x[4])+(.0189*x[6])+(.0003*x[8])+(.0007*u2);
  xk[5] =(.9010*x[5])+(.0275*x[7])+(.099*u1);
  xk[6] =(.901*x[6])+(.0275*x[8])+(.099*u2);
  xk[7] =(-4.334*x[5])+(.4132*x[7])+(4.334*u1);
  xk[8] =(-4.334*x[6])+(.4132*x[8])+(4.334*u2);

  xk[9] =x[9]+(.0395*x[11])-(.0054*x[12])-(.0113*x[13])+(.0012*x[14])-
        (.0001*x[15])+(.00001*x[16])-(.0002*u3)+(.00001*u4);
  xk[10] =x[10]+(.0054*x[11])+(.0395*x[12])-(.001*x[13])-(.013*x[14])-
         (.00001*x[15])-(.0002*x[16])-(.00001*u3)-(.0002*u4);
  xk[11] =(.9636*x[11])-(.2667*x[12])-(.553*x[13])+(.0879*x[14])-
         (.009*x[15])+(.001*x[16])-(.0203*u3)+(.0016*u4);
  xk[12] =(.2681*x[11])+(.9636*x[12])-(.01024*x[13])-(.6357*x[14])-
         (.0009*x[15])-(.0104*x[16])-(.0014*u3)-(.0234*u4);
  xk[13] =(.901*x[13])+(.0275*x[15])+(.099*u3);
  xk[14] =(.901*x[14])+(.0275*x[16])+(.099*u4);
  xk[15] =(-4.334*x[13])+(.4132*x[15])+(4.334*u3);
  xk[16] =(-4.334*x[14])+(.4132*x[16])+(4.334*u4);
```

G. SYSTEM RESPONSE

These steps are repeated with each frame cycle driving the error state to zero. The settling time is reflected by the transient behavior of AROD as displayed on the screen.

The AROD has the ability to move in a lateral direction with no forward speed. In the simulation, this is done by using the yaw control. When the vehicle yaws to the right or left, the downwash forces movement in that direction. This is reflected on the screen by a tilt of the horizon and movement over the terrain.

The forward and lateral speeds are a nonlinear function of the vehicle's pitch and yaw angles. A parabola is used as a best fit curve for these values and the equation is shown in Table 5.5.

TABLE 5.5

THE AROD VELOCITY EQUATIONS

```
/* velocity is a nonlinear function of pitch/yaw angles */

*speed = sqrt( 4 *fabs(xk[9]) * RTOD * 5.63 );
*latspd = sqrt( 4 *fabs(xk[10]) * RTOD * 5.63 );
if (xk[9] < 0.0)  *speed = -(*speed);
if (xk[10] < 0.0) *latspd = -(*latspd);
```

The unique design of the AROD allows for 360 degrees of movement while keeping the camera direction unchanged. The vehicle position is computed in Table 5.6 and displayed on the contour map during simulation.

TABLE 5.6

THE AROD POSITION EQUATIONS

up_arod_pos.c

```
/* compute distance AROD moves */

if (lastsec  == -999){
   deltadist = 0.0;
   deltalat = 0.0;
   }
else{
   deltadist = (speed/FPS_TO_KTS)*(seconds - lastsec);
   deltalat = (latspd/FPS_TO_KTS)*(seconds - lastsec);
    }
lastsec = seconds;         /* save for next pass */

if (designate) {

if (deltalat*deltalat < 0.01) {
deltalat = 0.0;
}

/* compute new position due to pitch */

     *vx += deltadist * cos(*direction);
     *vz -= deltadist * sin(*direction);

/* compute new position due to yaw */

     theta = *direction - HALFPI;

     *vx += deltalat * cos(theta);
     *vz -= deltalat * sin(theta);


/* crash if altitude equals ground level */

   gndlevel = gnd_level(*vx, *vz);
   if (*vy < gndlevel) {  /* crash */
   *flying = FALSE;
   lastsec = -999; /* no value for next launch */
   }
   }
```

H. INPUT LIMITS

The remaining portion of the READCONTROLS function is found in the Appendix and reflects physical constraints on the

vehicle's control surfaces. The AROD control vanes are limited from Ref.1 to a deflection of plus or minus .5256 radians, the servos to a velocity of plus or minus .8727 radians per second. Also, the throttle control can not exceed a deflection or velocity of plus or minus 100 radians per second.

VI. THE AROD SIMULATOR USER'S GUIDE

A. OVERVIEW

This section explains in detail the operation of the simulator. The background information presented here supplements that given on the screen. The format developed [Ref. 2:Chap. IX] is generally followed with the specifics to AROD included.

B. STARTING THE SIMULATOR

The first step to begin simulation is to logon to the IRIS workstation and enter the AROD directory. This is done by giving the following commands:

SCREEN	TYPE
IRIS1 console login	<usr name>
Password	<password>
IRIS 1%	/usr/work/<usrname>/work

The command 'arod' begins the execution. A welcome message will remain on the screen until the middle mouse button is pressed. Two additional screens are obtained, again by pressing the middle mouse button. Both of these contain beginning instructions. The operator can exit the simulation at any time by pressing all three mouse buttons simultaneously.

C. PREFLIGHT INFORMATION

At this point, the operator chooses a launch position and a direction of flight by moving the cursor. The first mouse button sets the launch position when pressed. This position can be changed at any time by moving to the new launch position and pressing the left mouse button. Pressing the right mouse button gives the line-of-sight distance to any position on the map. This distance is read in meters and the heading in degrees with both given in the lower right corner of the screen. Although the AROD ground station does not have this distance reading ability, it is used here as a reminder not to exceed the five kilometer round trip limit of fiber optic cable.

D. PRELAUNCH DISPLAY

From Ref. 2 the prelaunch display is divided into three sections: an instruction box, statistics box, and a two dimensional contour map. Each of the square grids on the map represent a one square kilometer area. The green areas indicate terrain with low elevation and the brown areas indicate terrain with higher elevations. Within these two colors, variations in the elevation are indicated by the intensity of the colors, the brighter the colors the higher the elevation. This is the opposite of the FOG-M simulator but is more natural to the operator. Pressing the middle mouse button launches the AROD.

E. FLIGHT CONTROLS

After launching, the display changes to reflect a variation to the control screen used for the FOG-M. This screen display provides the operator with more information than is currently available on the AROD ground station. Since the additional information should make initial learning easier, no effort was made to remove it.

The left side of the screen contains [Ref. 2]:

1. A three-dimensional view of the terrain as seen from the AROD camera.
2. A slider bar scale along the bottom edge indicating the camera pan angle.
3. A slider bar scale along the left hand edge indicating the camera tilt angle.
4. Camera cross hairs.

The upper right hand corner of the screen contains a frame rate display. This immediate feedback is helpful while the simulator continues to be refined.

Below this is a smaller version of the previously shown contour map. The AROD's position and look direction are displayed and updated every frame cycle.

The middle right portion of the screen lists the speed, look direction, altitude above ground level (AGL), altitude above mean sea level (MSL), and the camera settings.

The lower right portion of the screen contains a graphic description of the camera and attitude controls.

Instead of the small joystick found on the AROD ground control station, this simulator uses a mouse for all camera controls.

Flight control begins after launch. The vehicle quickly accelerates to approximately one hundred meters AGL. This is higher than on the actual flight system but the added ground clearance allows for more simulator transients without impacting the ground. If the altitude drops below ten meters, warning beeps are given, and if it drops below zero, an explosion results and the simulation ends. The operator is then returned to the prelaunch screen to prepare for another simulation.

The four dials on the control box control vehicle attitude and throttle.

Dial zero is the heading control and causes AROD to roll through the range 0-360 degrees. Turning the dial to the right causes the vehicle to rotate to the right and turning the dial left causes the vehicle to rotate to the left. The current heading in degrees is displayed on the screen.

Dial one is the throttle control which determines the altitude from ground level to 8000 meters. Turning the dial to the right increases the altitude and turning to the left decreases it.

Dial two is the pitch control, this changes the vehicles forward and backward speed from zero to thirty knots. Turning the dial to the right rotates the vehicle forward and causes forward movement. This forward tilt is shown on the screen as a

rise in the horizon due to a lower camera angle. Turning the dial left rotates the vehicle back and causes a rearward movement. This appears on the screen as a drop in the horizon due to a higher camera angle.

Dial three is the yaw control for lateral speeds also from zero to thirty knots. Turning the dial to the right rotates the vehicle clockwise giving a lateral velocity to the right. This is reflected on the screen as a counter clockwise twist in the horizon. Turning the dial to the left rotates the vehicle counter clockwise giving a lateral velocity to the left. This twists the horizon clockwise.

F. VEHICLE ANIMATION

A convoy of jeeps, trucks, and tanks from the FOG-M simulator will be added. This convoy begins on the eastern edge of the map and travels west at 15 knots. Terrain of any type is crossed without delay until the map boundary is reached. At this point the convoy reverses direction and continues.

VII. CONCLUSIONS AND RECOMMENDATIONS

A. PROJECT CONTINUATION

Before further work is done in the evolution of the AROD flight simulator, several questions should be answered: 1) Is the USMC going to continue development of this GATORS project or will it be suspended in light of the recent budget cuts? 2) If the answer to the first question is yes, are high speed graphics workstations really needed for training and will they be purchased.

B. FUTURE WORK

A comprehensive training syllabus is needed to cover initial and refresher training for AROD operators. This flight simulator is not intended as an alternative to flight experience with the system hardware. Instead, it can be incorporated into a training schedule to complement other training methods. Even in the optimum case a simulator can only approximate vehicle flight and may ignore the psychological factors. As Richard Gabriel wrote in his book Military Incompetence, "Any fool can function in a simulator" [Ref. 10].

To use this flight simulator for training purposes, the first step is to add realistic flight controls. The dynamics of this simulator all come from a linear model. This model was developed

using wind tunnel results and was successfully simulated on the computer. However, a comparison with the actual flight vehicle is needed. An operator who has flown the vehicle could match control sensitivities and increase the handling similarities. Instead of a control box and mouse, the ground station joysticks are needed for camera and attitude control. This would insure the simulator response closely approximates the AROD response for a given input.

The frame rate still needs to be increased with the motion picture rate of twenty-four frames per second the goal. With faster frame rates, more complexity can be added to the graphics. The first and most desirable upgrade is to increase the terrain resolution. The Fort Hunter Liggett DTED has eight times the resolution currently used.

C. RECOMMENDATION

AROD has the potential to give tactical capabilities that are otherwise not currently available. As further testing and development are completed on the vehicle, any changes should be incorporated into the AROD files. This will provide a real-time flight simulation system that will be ready when needed.

```
/*    This appendix is the heart of the flight simulation dynamics.*/
/* Operator input is read from the control box and sent to the     */
/* AROD discrete time model.  The system output response is then   */
/* used by other files to simulate vehicle movement on the screen. */

/*                      +-------------------------------+
                        |                               |
                        |        readcontrols.c         |
                        |                               |
                        +-------------------------------+          */
/* reads the values from the operator's controls (mouse and dials) */


#include "gl.h"                     /* graphics lib defs */
#include "arod.h"                    /* arod constants */
#include "device.h"                 /* device definitions */
#include "math.h"

read_controls(designate, greyscale, flying, active, speed,latspd,
              direction,compassdir, alt, pan, tilt, fovy,xk)

int *designate, *greyscale, *flying, *active, *fovy;
float *speed,*latspd, *compassdir,*xk;
double *direction, *pan, *tilt;
Coord *alt;

{
  extern float randx, randy, randz;
  float randnum(),x1c,x2c,x3c,x4c;
  Colorindex colors[1];

  /* quit if all three mouse buttons are pushed */
  if(getbutton(MOUSE1) && getbutton(MOUSE2) && getbutton(MOUSE3)) {
      *flying = FALSE;
      *active = FALSE;
  }
  else {
     if (getbutton(MOUSE3) && !(getbutton(MOUSE2))) {    /* Zoom In */
        *fovy = (*fovy < (80 + DELTAFOVY)) ? 80 : *fovy - DELTAFOVY;
     }

     if (getbutton(MOUSE1) && !(getbutton(MOUSE2))) {    /* Zoom Out */
        *fovy = (*fovy > (550 - DELTAFOVY)) ? 550 : *fovy + DELTAFOVY;
     }



      *pan = DTOR * (double)(-getvaluator(MOUSEX)) / PANSENS;
      *tilt = DTOR * (double)(getvaluator(MOUSEY)) / TILTSENS;



/*    readcontrol dial input  */

      x1c = (float)getvaluator(DIAL0) /DIRSENS;      /* roll     */
      x2c = (float)(getvaluator(DIAL1)/THROTSENS);   /* throttle */
      x3c = (float)(getvaluator(DIAL2)/PITCHSENS );  /* pitch    */
```

```
    x4c = (float)(getvaluator(DIAL3)/YAWSENS );    /* yaw      */


    /* keep *direction between 0 and 360, update valuator if changed */

    while (x1c >= 360.0) {
       x1c -= 360.0;
       setvaluator(DIAL0,(int)(x1c*DIRSENS), (int)(-360*DIRSENS),
                   (int)(720*DIRSENS));
    }

    while (x1c < 0.0) {
       x1c += 360.0;
       setvaluator(DIAL0,(int)(x1c*DIRSENS), (int)(-360*DIRSENS),
                   (int)(720*DIRSENS));
    }
  /*convert *direction from compass degrees to trigonometric radians */
    *direction = (x1c <= 90.0) ? DTOR * (90.0 - x1c) :
                                 DTOR * (450.0 - x1c);
    x1c = x1c * DTOR;


    dynam(x1c,x2c,x3c,x4c,xk);


    *compassdir = xk[1]*RTOD;
    *alt = (Coord)(xk[2]*1000.0);

    /* velocity is a nonlinear function of pitch/yaw angles */

    *speed = sqrt( 4 *fabs(xk[9]) * RTOD * 5.63 );
    *latspd = sqrt( 4 *fabs(xk[10]) * RTOD * 5.63 );
    if (xk[9] < 0.0)  *speed = -(*speed);
    if (xk[10] < 0.0) *latspd = -(*latspd);

  }
  }
  dynam (x1c,x2c,x3c,x4c,xk)
  float x1c,x2c,x3c,x4c,*xk;
             /* x1c input control;ailerons control heading        */
             /* x2c input control;throttle controls elevation     */
                  /* x3c input control;elevators control forward speed */
                  /* x4c input control;rudder controls lateral speed   */


{ float u1,u2,u3,u4,x[17],ux[15];


  int cntr, cntr1;


 /*                 state space representation                     */
 /*     discrete form; x(k+1) = plant matrix*x+control matrix*u    */


  for (cntr1 = 1; cntr1<=5; ++cntr1)    /* temp. for slow frame rate  */
  {


      for (cntr = 0; cntr<17; ++cntr) /* set new state values before */
```

```
      x[cntr] = xk[cntr];        /*       incrementing system          */

   ux[1] = x[1]-x1c; /* check for control input;the error state vector */
   ux[2] = x[2]-x2c;
   ux[9] = x[9]-x3c;
   ux[10] = x[10]-x4c;

     /*      apply steady-state gain schedule to commanded input       */

u1=(1.99*ux[1])+(.603*ux[2])+(.717*x[3])-(.274*x[4])-
   (1.75*x[5])-(.00994*x[6])-(.140*x[7])-(.00016*x[8]);

u2=(.585*ux[1])-(111.0*ux[2])+(.205*x[3])-(28.9*x[4])-(.582*x[5])-
   (1.67*x[6])-(.00962*x[7])-(.137*x[8]);

u3=(1.61*ux[9])+(1.61*ux[10])+(.553*x[11])-(.32*x[12])-(1.05*x[13])+
   (.208*x[14])-(.127*x[15])+(.00264*x[16]);

u4=(-1.57*ux[9])+(1.6*ux[10])+(.305*x[11])+(.566*x[12])-(.107*x[13])-
   (1.23*x[14])-(.00096*x[15])-(.130*x[16]);

   /*       multiply states by discrete time system matrices          */

xk[1] =x[1]+(.04*x[3])-(.003*x[4])-(.0167*x[5])-(.0002*x[7])-(.0003*u1);
xk[2] =x[2]+(.035*x[4])+(.0004*x[6])+(.00001*u2);
xk[3] =(.0009*x[2])+x[3]-(.1466*x[4])-(.8217*x[5])-(.0015*x[6])-
       (.0134*x[7])-(.0299*u1)-(.00003*u2);
xk[4] =(.9608*x[4])+(.0189*x[6])+(.0003*x[8])+(.0007*u2);
xk[5] =(.9010*x[5])+(.0275*x[7])+(.099*u1);
xk[6] =(.901*x[6])+(.0275*x[8])+(.099*u2);
xk[7] =(-4.334*x[5])+(.4132*x[7])+(4.334*u1);
xk[8] =(-4.334*x[6])+(.4132*x[8])+(4.334*u2);

xk[9] =x[9]+(.0395*x[11])-(.0054*x[12])-(.0113*x[13])+(.0012*x[14])-
       (.0001*x[15])+(.00001*x[16])-(.0002*u3)+(.00001*u4);
xk[10] =x[10]+(.0054*x[11])+(.0395*x[12])-(.001*x[13])-(.013*x[14])-
        (.00001*x[15])-(.0002*x[16])-(.00001*u3)-(.0002*u4);
xk[11] =(.9636*x[11])-(.2667*x[12])-(.553*x[13])+(.0879*x[14])-
        (.009*x[15])+(.001*x[16])-(.0203*u3)+(.0016*u4);
xk[12] =(.2681*x[11])+(.9636*x[12])-(.01024*x[13])-(.6357*x[14])-
        (.0009*x[15])-(.0104*x[16])-(.0014*u3)-(.0234*u4);
xk[13] =(.901*x[13])+(.0275*x[15])+(.099*u3);
xk[14] =(.901*x[14])+(.0275*x[16])+(.099*u4);
xk[15] =(-4.334*x[13])+(.4132*x[15])+(4.334*u3);
xk[16] =(-4.334*x[14])+(.4132*x[16])+(4.334*u4);

/*                        servo outputs                                */

/* check the control vane displacement, ensure it doesn't exceed +-max */

if (xk[5]>maxc)
   xk[5] = maxc;
   if (x[5]<-maxc)
      xk[5] = -maxc;
```

```
if (xk[13]>maxc)
   xk[13] = maxc;
   if (xk[13]<-maxc)
      xk[13] = - maxc;

if (xk[14]>maxc)
   xk[14] = maxc;
   if (xk[14]<-maxc)
      xk[14] = -maxc;

if (xk[7]>maxv)
   xk[7] = maxv;
   if (xk[7] <-maxv)
      xk[7] = -maxv;

if (xk[15]>maxv)
   xk[15] = maxv;
   if (xk[15] <-maxv)
      xk[15] = -maxv;

if (xk[16]>maxv)
   xk[16] = maxv;
   if (xk[16] <-maxv)
      xk[16] = -maxv;

   /*    check the throttle setting,ensure it doesn't exceed +-max    */

if (xk[6]>maxt)
   xk[6] = maxt;
   if (xk[6]<-maxt)
      xk[6] = -maxt;

if (xk[8]>maxtv)
   xk[8] = maxtv;
   if (xk[8] <-maxtv)
      xk[8] = -maxtv;

       /* return system response for given input and iterate through  */
       /* steady state */
}
}
```

LIST OF REFERENCES

1. Basset, W. G., A Dynamic Simulation and Feedback Control Scheme For The U. S. Marine Corps' Airborne Remotely Operated Device (AROD), M. S. Thesis, Naval Postgraduate School, Monterey, California, September 1987.

2. Smith, D. B. and Streyle, D. G., An Inexpensive Real-Time Interactive Three-Dimensional Simulation System, M. S. Thesis, Naval Postgraduate School, Monterey, California, June 1987.

3. Hanson, S. D., "Low Altitude Training Needs A New Approach", Marine Corps Gazette, December 1987.

4. Lloyd, S., notes on USMC GATERS Program, Quantico, Virginia.

5. Herskovitz, Sheldon B., "Planes Without People", Journal of Electronic Defense, Page 39, June 1988.

6. Dupuy T. N. Colonel, U. S. Army, Ret., The Evolution of Weapons and Warfare, Bobs-Merrill, 1980.

7. Lloyd, S.D., An Auto Pilot Design for the USMC's AROD, M.S. Thesis, Naval Postgraduate School, Monterey, California, September 1987.

8. Silicon Graphics, Inc., Iris User's Guide, Mountain View, California, 1986.

9. Pratt, D., Class Project, CS4202 Computer Graphics, Prof. M. J. Zyda, Naval Postgraduate School, Monterey, California, Spring Quarter, 1988.

10. Oliver, M. R. and Stahl, D. J. Jr., Interactive, Networked, Moving Platform Simulators, M. S. Thesis, Naval Postgraduate School, Monterey, California, December 1987.

11. Astrom, K. J. and Wittenmark, B., Computer Controlled Systems, Prentice-Hall, Inc., 1984.

12. Class notes, EC4310, Digital Control Systems, Prof. R. Cristi, Naval Postgraduate School, Monterey, California, Spring Quarter, 1987.

13. Kirk, D.E., Optimal Control Theory, Prentice-Hall, Inc., 1970.

14. "The Military's New Stars", U.S. News & World Report, Page 40, April 18, 1988.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Commandant of the Marine Corps Code TE 06 Headquarters, U. S. Marine Corps Washington, D.C. 20360-0001	1
2.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
3.	Library (Code 0142) Naval Postgraduate School Monterey, California 93943-5000	2
4.	Chairman (Code 62) Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
5.	Curricular Officer (Code 32) Electronics and Communications Programs Naval Postgraduate School Monterey, California 93943-5000	1
6.	Ground/Air Tele Robotics Project Office Development Center Marine Corps Combat Development Command Quantico, Virginia 22340	1
7.	Professor Harold A. Titus (Code 62Ts) Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	2
8	Professor Michael J. Zyda (Code 52Zk) Computer Science Department Naval Postgraduate School Monterey, California 93943-5000	1

9. Thomas Hughes
532 NAVAL OCEAN SYSTEMS CENTER 1
P.O. Box 997 Kailua, Hawaii 96734-0997

10. CAPT S. P. Martinson 3
Route 1 4KS
Bismarck, North Dakota 58501

Thesis
M359282 Martinson
c.1 An inexpensive real-time flight simulator for the United States Marine Corps' Airborne Remotely Operated Device.