

AD-A160 693

DEVELOPMENT OF A SIMULATOR FACILITY FOR HELICOPTER  
AIR-TO-AIR COMBAT(U) NATIONAL AERONAUTICS AND SPACE  
ADMINISTRATION MOFFETT FIELD CA AMES RESEARCH CENTER  
M S LEWIS ET AL. 1985

1/1

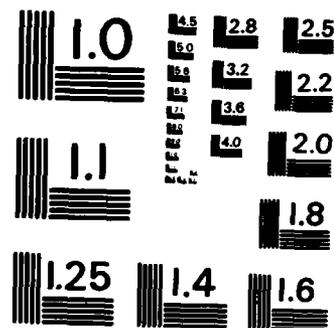
UNCLASSIFIED

F/G 9/2

NL

■■■■■■■■■■■

END  
FORMED  
DTIC



MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

# AIAA'85

①

AD-A160 693

AIAA-85-1733

## DEVELOPMENT OF A SIMULATOR FACILITY FOR HELICOPTER AIR-TO-AIR COMBAT

Michael S. Lewis  
Army Aeroflightdynamics Directorate  
NASA Ames Research Center  
Moffett Field, CA 94035

Dennis G. Yeo  
Software Systems  
San Jose, CA 95126

DTIC  
ELECTE  
S OCT 23 1985 D  
A

DTIC FILE COPY

This document has been approved  
for public release and sale; its  
distribution is unlimited.

AIAA Flight Simulation Technology Conference  
July 22-24, 1985/St. Louis, Missouri

85 10 28 009

## DEVELOPMENT OF A SIMULATOR FACILITY FOR HELICOPTER AIR-TO-AIR COMBAT

Michael S. Lewis\*

Army Aeroflightdynamics Directorate  
NASA Ames Research Center, Moffett Field, CA 94035

Dennis G. Yeof

Software Systems, San Jose, CA 95126

### Abstract

A number of Computer Generated Imagery (CGI) modifications required to develop a simulation facility for low-level, one-on-one helicopter air combat are discussed. These modifications to the NASA Ames Vertical Motion Simulator (VMS) system allowed two aircraft to be independently piloted on a single CGI data base. Air combat simulation engagements were flown by test pilots in a highly realistic fashion.

### Introduction

The NASA Ames Digital Image Generation (DIG) system was purchased from the Link division of Singer Inc. The system provides a simulated out-the-window, visual image for a large-vertical-motion flight simulator. As originally designed, the DIG can support four visual channels, which represent four aircraft windows with a single pilot eyepoint. The visual system was originally designed to support flight simulations of high altitude, high speed, fixed-wing aircraft.

A Helicopter Air Combat (HAC) simulation required modifications to the DIG real-time software to support two helicopters engaged in low speed, low altitude, air-to-air combat. These modifications included a second eyepoint to provide an out-the-window visual image for the second aircraft pilot, new moving-target occulting algorithms, an algorithm to test for the existence of a clear line of sight between the opposing helicopters, and the addition of moving rotors on the helicopters in the scene. All of these modifications were crucial to a realistic simulation for experimental investigations of the helicopter air combat task. A detailed description of the facilities, design, and conduct of the HAC simulation can be found in Refs. 1 and 2. This paper will present only the visual scene developments.

### Multiple Eyepoints

Perhaps the most challenging requirement of the HAC visual simulation was to allow one of the four available visual channels to display the

target (red) aircraft eyepoint, while the remaining three channels were used conventionally to display the ownship (blue) aircraft eyepoint. The requirement was met by a solution that allowed the multiple eyepoint feature to be controlled by operator input to the simulation, so that any number of eyepoints up to four could be supported. Thus, while the four CGI visual channels had previously provided a four-window, out-the-cockpit view from one aircraft only, they can now provide single-window, out-the-cockpit views for up to four independently maneuvering aircraft. As stated previously, a 3 + 1 arrangement was used for the HAC simulation.

The multiple eyepoint problem was difficult because the real-time DIG software does not send the pilot's eyepoint and attitude matrix to the DIG hardware at the rate of one time (per window) per frame. Rather, it is sent once (per window) per scene component. Scene components include the ground, the sky, clouds, each target in the scene, and many other optional scene features. Modification of the software for each scene component to send the correct pilot's eyepoint information would have been a large and complicated project. Instead, the low-level routines used by all scene components to insert the eyepoint information (position vector and attitude matrix) in each frame were modified.

Some background on the DIG system must be given before proceeding. The DIG software outputs object lists each frame (30 times/sec). The object lists contain for each window 1) the pilot's eyepoint position and attitude, and 2) the objects to be displayed and their occulting priorities. The occulting priorities are calculated in background (approximately 6 times/sec), and therefore several frames pass with no change in this information. However, the illusion of motion depends upon the updating of position and attitude data for every frame in the object list. Therefore, this eyepoint information is moved into the object list for each frame, just before the entire object list is sent to the DIG hardware.

Multiple eyepoints were implemented by modifying the low-level routines which insert the position and attitude into the object list on a frame-by-frame basis. If the window which is being inserted has been assigned to an eyepoint other than that of the ownship, the new software inserts the target's eyepoint instead of the ownship's. Since this low-level routine is used by

\*Aerospace Engineer.

†Software Engineer.

This paper is declared a work of the U.S. Government and therefore is in the public domain.

all scene components, only minimal modifications to the software were required.

Along with a second eyepoint, another modification was required for a model of the ownship to be used when it is viewed from the target. This was accomplished by using special-purpose software which captures the ownship position and attitude and stores it as a "target." This target is then displayed only in windows that are not assigned to the ownship.

The result of these modifications is quite impressive. For the first time, the DIG is capable of supporting multiple aircraft and their corresponding eyepoints. In addition, a fixed, second eyepoint position is possible for such uses as an overhead display of the CGI data base, optical sensor simulation, or outside viewing of aircraft approach and landings.

#### Moving-Object Occulting Algorithm

The Singer Link DIG employs the Separating Plane Tree (SPT) algorithm, which is widely used in real-time, three-dimensional graphics systems to rapidly determine the relative occulting among a fixed set of objects.

The SPT algorithm is executed by the DIG software approximately 6 times/sec to assign occulting priorities to objects in the scene relative to the eyepoint. This information is transmitted to the DIG hardware, and is used to resolve occulting of objects which overlap one another when projected on the screen. A substitution similar to that described in the multiple-eyepoint section is done when processing the SPT algorithm. If a visual channel which is being processed is assigned to an alternate eyepoint, that eyepoint is used in the SPT algorithm instead of the ownship eyepoint.

The numeric descriptions in a DIG visual data base contain both displayable object descriptions (for example, the coordinates of an object and each vertex) and a nondisplayable, binary SPT. Each node of the SPT contains the coefficients (A, B, C, D) of a plane which separate one or more objects from other objects in the data base. The endpoints, or terminal nodes, of an SPT contain references to an object in the data base. For the purposes of the SPT algorithm, the equation  $Ax + By + Cz + D = E$  is calculated. As a plane in three dimensions has the equation  $Ax + By + Cz + D = 0$ , substitution of the eyepoint's position in (x,y,z) will yield a value of E greater than, equal to, or less than zero, depending upon whether the eyepoint is on the positive side, exactly on, or on the negative side of the plane.

The processing of an SPT is best shown by example. Figure 2 represents a simple visual data base with four objects as viewed from the top down. Figure 3 represents the same data base with

the invisible separating planes shown. Figure 4 represents the actual SPT. As the DIG software processes the SPT to determine the occulting priority, it first evaluates the eyepoint relative to the top node of the tree--plane L. When the eyepoint (x,y,z) coordinates are multiplied by the coefficients of plane L and summed, a positive value of E results, which indicates the eye is on the positive side of the plane. The negative side of the SPT (in this case a pointer to object 1) is placed in a memory file. The positive side of the SPT node (in this case the equation of plane M) is then evaluated. The eyepoint coordinates evaluated in the equation of plane M yield a negative value of E, so the positive pointer (to plane N) is stored in memory and object 4 is encountered. Object 4 then is assigned priority 0, meaning it has the highest occulting value, and placed in an output list. Since a terminal (object) node of the tree has been reached, the last item stored is removed from memory, which in this case is the pointer to plane N. Continued evaluation of the SPT yields occulting priorities for objects 2, 3, and 1.

The SPT algorithms are used in the DIG for both ground objects and the objects that make up a moving target. Since the DIG was designed for fixed-wing aircraft, it was assumed that moving targets were flying at high altitudes, and therefore were assigned occulting priorities ahead of the ground objects. This means that ground objects would never occult a moving object, regardless of the actual range. While acceptable for high altitude simulations, this was not acceptable for the HAC simulation, where the moving target is a low altitude helicopter. A modification to correctly merge ground and moving-target occulting priorities was needed.

To accomplish this modification, DIG software was modified with the following algorithm: 1) The object closest to the moving target is found by using the target position as the eyepoint, and traversing the ground SPT until the first object is found. 2) The closest object in the SPT is tagged with a flag. 3) The ground SPT for the ownship is processed normally until the tagged object is encountered. 4) A range calculation is made to determine whether that object or the target is closer to the ownship pilot's eyepoint. 5) The target's SPT is then processed to assign occulting priorities continuing with the next priority unused by the ground scene. 6) Processing of the ground SPT continues with the next available priority. This is shown in Figs. 5, 6, and 7.

The algorithm works well in most cases, although some occulting anomalies can exist between the closest object and the target. These anomalies arise because the range calculation from the eyepoint to a ground object is an approximation, since the ground object is identified by a single point. Thus, terrain objects which were as large as 800 m across caused particular problems.

These problems were overcome acceptably by modeling a small feature on the surface of each terrain face; this feature had a separating plane between it and the terrain surface. Thus, the closest object selected was the small feature, rather than the terrain, and the range check was then sufficient to resolve the priority.

#### Moving Rotors

The Singer Link DIG allows a moving object to be displayed in the scene by specifying its object occulting priorities, the  $x,y,z$  positions, and an attitude matrix. To represent moving rotors, the two rotor objects were isolated during the SPT processing of the helicopter model, and a rotation matrix was applied to them which was separate from the rest of the aircraft. The rotors were assumed to have a fixed RPM, and a single-rotor rotation matrix was calculated for each frame which represented the current rotor rotation. The rotor objects were flagged in the data bases, and the SPT routine saved them in a separate list after assigning them the proper occulting priority relative to the rest of the target objects. The hardware was then commanded to display the helicopter model with its correct position and attitude, and to display the rotor as another moving object with the attitude corrected for the rotor rotation. The task was simplified by making the center of gravity of the helicopter models the same as the point of rotation on the rotor. The result was realistically rotating rotors on the target models as seen from an outside eyepoint.

#### Line-of-Sight Calculations

To develop the HAC scoring algorithms, the designers needed to know whether the red and blue aircraft could see one another, or whether the terrain obstructed their line of sight. The occulting priority calculations using the SPTs described above do not yield this information; they calculate only a relative number which is used to solve line-of-sight occulting if the target and terrain are projected onto the same screen pixels. This information is calculated deep in the DIG hardware and is not available to the computer software. Therefore a fast algorithm was developed that could be performed 10 times/sec in the software.

As Fig. 8 shows, the HAC data base is represented by a 3-km square area with 12 peaks. Each peak is a common vertex for a hill with triangular sides. The common edges of the triangles which make up the hills are, in most cases, ridge lines. If the line of sight between the two aircraft intersects below a ridge line, the line of sight is assumed to be blocked by the hill; above a ridge line, the line of sight is assumed to be clear.

An off-line program was developed to format the ridge-line data for processing. The program produces a ridge-line table on a disk containing 1) the coefficients of the line equation in two dimensions projected onto the  $(x,y)$ ,  $(y,z)$ , and  $(x,z)$  planes and 2) the endpoints of each line segment.

Figures 9, 10, and 11 illustrate the algorithm applied to each ridge line in the data base in real time. First, the coefficients of the line representing the line of sight (shown as CD in the figure) are calculated. Next the same theory of inequalities used in the SPT calculations is used, but this time in two dimensions (the  $x,y$  plane). The coordinates of the red ship (C) and blue ship (D) are substituted into the equation for the ridge line (AB). If the signs of these substitutions are the same (Fig. 9), the ridge line is not crossed by the line of sight, and the next ridge line can be examined. If the signs are opposite (Fig. 10), then the  $x,y$  coordinates of the ridge-line endpoints (A and B) are substituted into the line-of-sight equation. If the signs are the same (Fig. 11), the ridge line can be eliminated from further processing since the ridge line is not between the aircraft but displaced to one side. If they are of opposite signs (Fig. 10), the line of sight intersects the ridge line. The two values of  $z$  at the point of intersection then reveal whether the ridge line blocks the line of sight, or whether the two aircraft are above it with a clear line of sight. As shown in the example of Fig. 12, the line of sight is clearly blocked by the ridge line. Figure 13 illustrates this situation in a perspective view.

This algorithm works very well. It takes a small amount of storage, and provides an extremely fast and efficient method for determining whether a clear line of sight exists between two aircraft. If used in a much larger data base with hundreds or thousands of peaks and ridge lines, however, a method would need to be developed to isolate a subset of data around the aircraft which would work in real time.

#### Results and Recommendations

The advantages of conducting research on a simulator as opposed to a flight test are many, including the ability to conduct low-level flights safely, shorter separation distances between aircraft, larger amplitude maneuvers, and variable aircraft configurations. The CGI developments just described were crucial to the successful conduct of what is probably the first real-time simulation of helicopter air combat in the presence of terrain features. Pilot comments indicated the level of engagement realism was high and the flightpaths and tactics compared favorably to flight test maneuvers. Two typical pilot comments were as follows:

You have completely ruined me now. I am flying this mission the way I would a real EVM (evasive maneuvering) engagement. I was flying off the cues that I perceived for off the relative motion of the target aircraft. Even when I was above him in a hover, in a pedal turn, I've adapted enough now that I had him in the center of the right console window, maybe 20° down, and was doing pedal turns keeping him there. I really flew that one the way I flew the ones at Patuxent River in relationship to the other aircraft, disregarding the ground. I never looked at my altimeter one time and I am now assimilating enough cues so I'm flying (the simulator) the way it is flown in the aircraft.

Following another engagement, the comments were similar:

The scenario we just went through, as far as what I have seen, other than the bank angles--the bank angles were larger here--but the maneuvering was as realistic as anything that we have done in here and very representative of what I would expect to see in an encounter like that. (Ref. 2)

Some improvements to the simulation programs are possible. Since the line-of-sight calculations were completely calculated every three frames, an anomaly would occur if the identified occulting feature changed from one ridge line to another. For the one to five frames which would elapse before the next occulting ridge line was found, the opponent aircraft location would be displayed as if there were a clear line of sight. A modification to correct this situation would not be difficult to implement.

Since the x,y,z coordinates of the terrain features were entered manually from a plot of the CGI data base, some errors were made. However,

during the course of the simulation experiment, these errors were gradually discovered and fixed. An automatic program to transfer data from the CGI data base to the line-of-sight program would be desirable.

The modeling of small terrain features near the larger hills to improve the moving-target occulting determination was an acceptable solution. The data base was partially revised to eliminate some anomalies, but further revision is needed. The occurrence of occulting anomalies, or "see-through" hills was rare, however. The cost of adding these terrain features is low both in measures of implementation time and CGI real-time processing, and such an improvement will be pursued for further experiments.

A possible application of a second-eyepoint CGI capability would be to allow present multi-window CGI training simulators to incorporate an air combat mission. However, the benefits of these CGI modifications do not pertain to air combat simulations only. The ability to locate a second eyepoint outside of the ownship cockpit allows any multiwindow CGI system additional flexibility. As mentioned earlier, approach-and-landing tests could be monitored from a ground position for further insight. In addition, an overhead, lookdown eyepoint could be used to trace the groundpath of a particular aircraft. Since the computational cost of these additional eyepoints is minimal, each CGI window could be assigned a different eyepoint during real-time simulation. For any CGI system with a playback-and-record feature, all CGI windows could be utilized for a full-field-of-view flight test and then viewed afterwards from a different eyepoint.

#### References

<sup>1</sup>Lewis, M., and Aiken, E., "Piloted Simulation of One-on-One Helicopter Air Combat at NOE Flight Levels," NASA TM-86686, 1985.

<sup>2</sup>Lewis, M., "A Piloted Simulation of One-on-One Helicopter Air Combat at NOE Flight Levels," 41st Annual Forum of the American Helicopter Society, Ft. Worth, TX, May 1985.

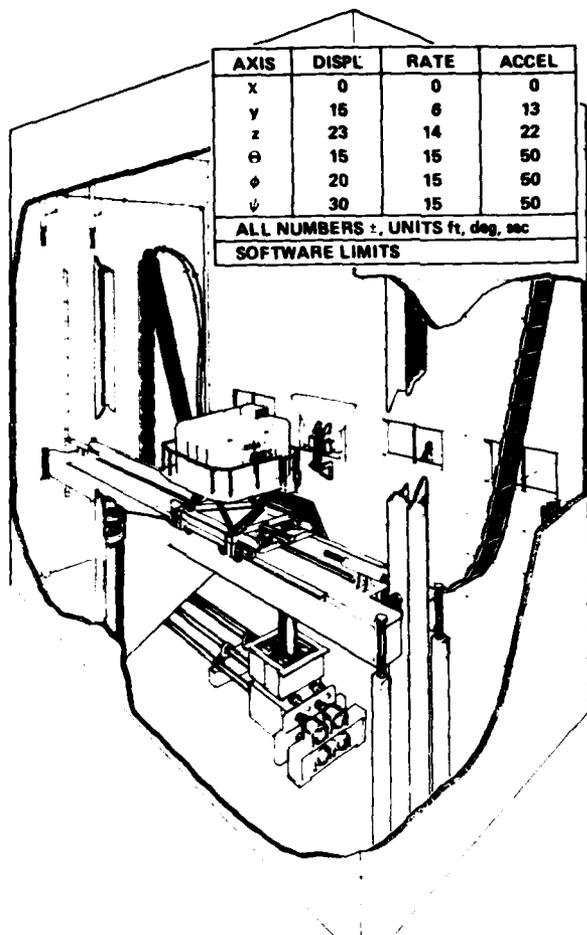


Fig. 1 NASA Ames Vertical Motion Simulator.

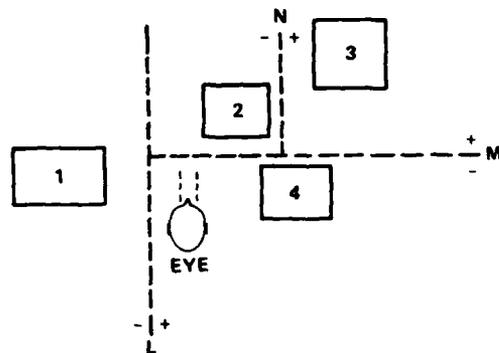


Fig. 3 Separating planes in the same scene.

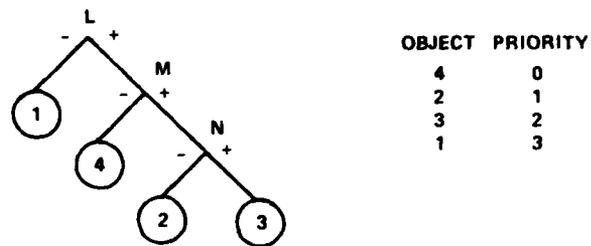


Fig. 4 Separating plane tree and assigned occulting priorities.

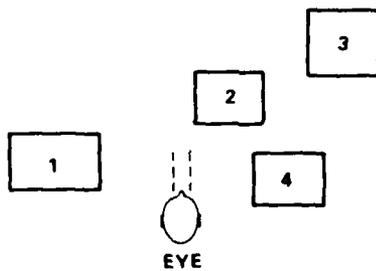


Fig. 2 Top-down view of simple ground scene and eyepoint.

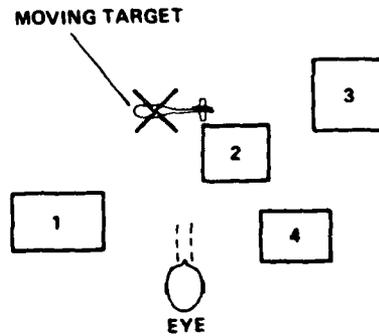


Fig. 5 Top-down view of simple ground scene and eyepoint.

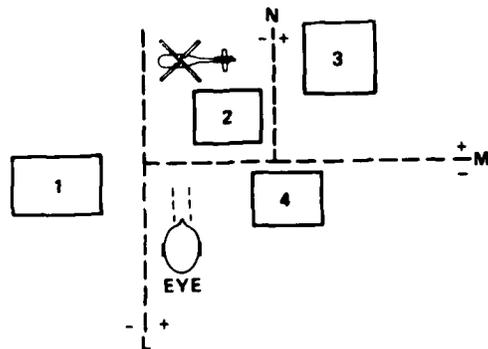
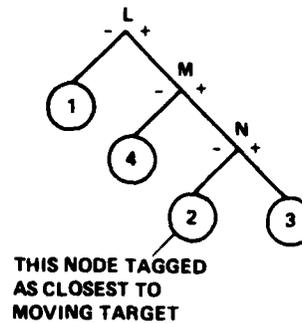


Fig. 6 Separating planes in the same scene.



OBJECT	PRIORITY
4	0
2	1
(PRIOS 2 - 6 FOR TARGET)	
3	7
1	8

Fig. 7 Separating plane tree and assigned occulting priorities.

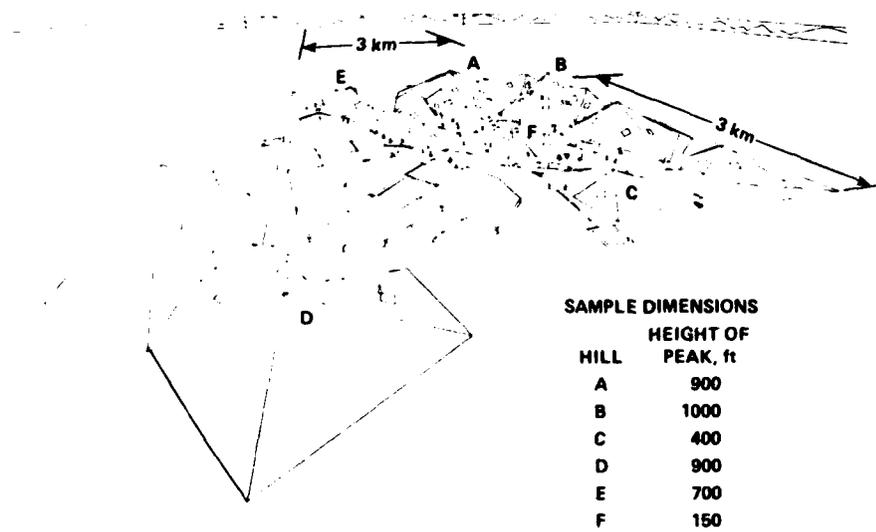


Fig. 8 CGI gaming area.

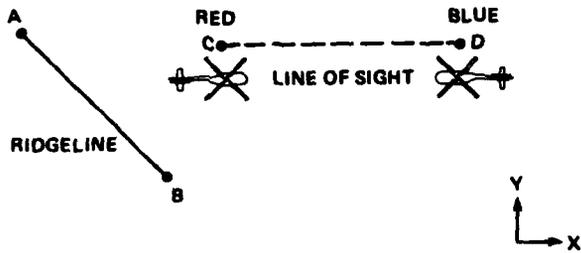


Fig. 9 Line-of-sight CD not blocked by ridge-line AB.

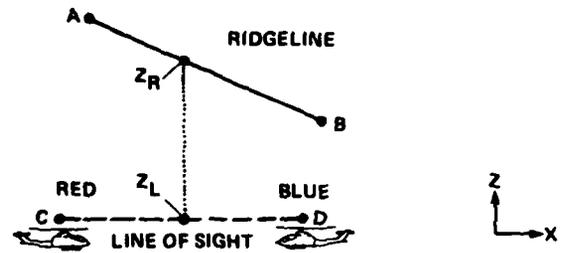


Fig. 12 When the case in Fig. 10 is viewed in ZX plane, it is clear that the ridge line blocks the line of sight because  $Z_L$  is less than  $Z_R$ .

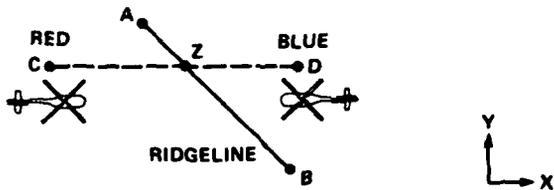


Fig. 10 Possible blockage of line-of-sight CD by ridge-line AB depending on Z values at point of intersection.

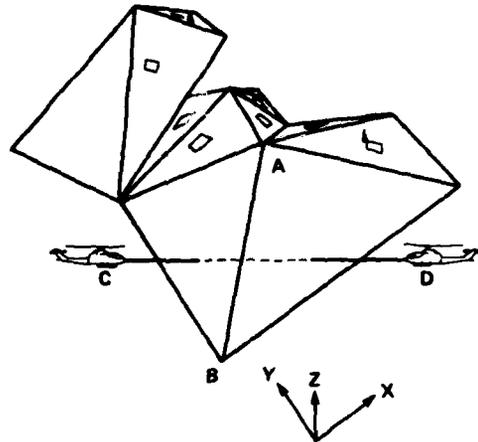


Fig. 13 Perspective view of occluded line of sight.

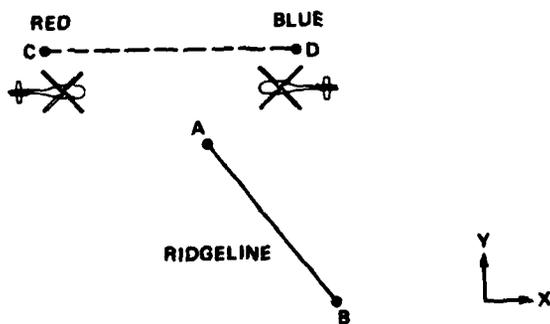


Fig. 11 Line-of-sight CD not blocked by ridge-line AB.



DTIC COPY INSPECTED	
By Distribution:	
Availability Codes	
Dist	Avail and/or Special
A-1	

**END**

**FILMED**

**12-85**

**DTIC**