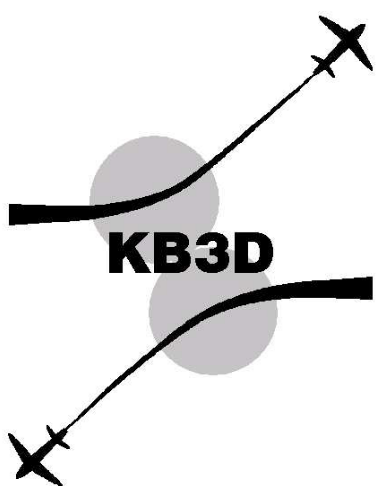
NASA/TM-2005-213769

KB3D Reference Manual – Version 1.a

Cesar Munoz and Radu Siminiceanu
National Institute of Aerospace, Hampton, Virginia

Victor A. Carreno
Langley Research Center, Hampton, Virginia

Gilles Dowek
École Polytechnique, Palaiseau, France



June 2005

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Phone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

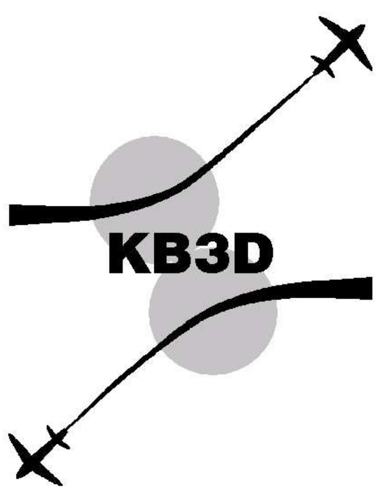
NASA/TM-2005-213769

KB3D Reference Manual – Version 1.a

Cesar Munoz and Radu Siminiceanu
National Institute of Aerospace, Hampton, Virginia

Victor A. Carreno
Langley Research Center, Hampton, Virginia

Gilles Dowek
École Polytechnique, Palaiseau, France



National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

June 2005

Acknowledgments

The development of KB3D was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-97046 and NASA Cooperative Agreement NCC1-02043. The KB3D logo was designed by Mahyar Malekpour at NASA LaRC.

Available from:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 605-6000

Contents

Chapter 1

Introduction

KB3D [3] is a pair-wise conflict detection and resolution (CD&R) algorithm developed at the former research institute ICASE at NASA Langley Research Center. The input to KB3D is the position and velocity vectors of two aircraft. KB3D distinguishes the host aircraft as the *ownship* and the traffic aircraft as the *intruder*. The output is a list of resolution maneuvers for the ownship. The maneuvers computed by KB3D are new velocity vectors that involve the modification of a single parameter of the ownship's original flight path: vertical speed, heading, or ground speed. KB3D is not computationally intensive and, therefore, is ideally suitable for distributed airborne deployment.

KB3D is characterized by the following features:

- *Distributed*: Each aircraft solves its own conflicts with respect to traffic aircraft.
- *Three dimensional*: It proposes horizontal and vertical resolutions.
- *State-based*: It solves conflicts based only on the state information of each aircraft, i.e., current position and velocity vector.
- *Tactical*: It uses a short lookahead time, typically 5 minutes or less.
- *Geometric*: It finds analytical solutions, in a Cartesian Coordinate system, assuming linear trajectory projections of current aircraft states.

KB3D's logic is comparable to CD&R algorithms such as geometric optimization [1] and modified voltage potential [7]. In contrast to those algorithms, KB3D resolutions are simultaneously *independent* and *coordinated*. Independent means that each resolution maneuver effectively solves the conflict assuming that only the ownship maneuvers. Coordinated means that separation is also achieved when both aircraft maneuver. These features provide additional layers of safety. Indeed, independent maneuvers mitigate potential conflicts when one aircraft does not maneuver due to equipment failure or other factors. Coordinated maneuvers guarantee that aircraft will not maneuver into each other when solving a conflict.

The mathematical proprieties of KB3D have been extensively studied and formalized in the Prototype Verification System (PVS) [9] (see for example [2, 4, 5, 6, 8]).

1.1 KB3D Concepts

KB3D considers a Cartesian 3-D airspace where the *state* of an aircraft is given by its current position and velocity vector. The *protected zone* is a cylinder of diameter D and height H centered around each aircraft. A *loss of separation*, or *violation*, occurs when the protected zones of two aircraft overlap. A *conflict* is a predicted loss of separation in the future up to a *lookahead time* T. If a conflict is detected, KB3D returns the time interval of loss of separation.

A *resolution* is a new velocity vector for the ownship that achieves separation assuming that the intruder does not maneuver, or that it maneuvers according to its independently computed KB3D resolution. KB3D returns three kinds of resolutions:

- *Vertical speed only*: The aircraft keeps the horizontal component of its velocity vector, but modifies its vertical speed.
- *Heading only*: The aircraft keeps its ground speed and vertical speed but modifies its heading.
- *Ground speed only*: The aircraft keeps its heading and vertical speed but modifies its ground speed.

Not all conflict situation has all three kinds of resolutions. However, if the aircraft are not originally in violation, the KB3D algorithm guarantees at least one theoretical vertical solution for each aircraft.

1.2 CD&R System

The KB3D algorithm has been designed as the kernel of a CD&R system. As such, it provides the basic functionality for conflict detection and resolution. However, KB3D does not filter its inputs for possible errors, nor does it check the resolution maneuvers for physical feasibility. This kind of functionality has to be implemented at a higher-level based on the performance parameters of the aircraft and additional external parameters.

Figure 1.1 illustrates an abstract view of a possible integration of KB3D in a CD&R system. Hardware systems such as sensors and data links provide the state information of the ownship and traffic aircraft needed by KB3D. This information is first checked for consistency and formatted according to KB3D requirements. During this process, data errors are filtered out. Once KB3D is executed, the result of its conflict detection logic goes to an alerting module that determines if, when, and how an alert should be displayed. Then, KB3D's resolutions go to a resolution advisory module that selects the maneuvers that are physically feasible for the aircraft. The alerting and resolution advisory modules provide inputs to avionics systems such as cockpit displays, flight management, and navigation system. The dashed arrow from the avionics systems to the hardware systems represents the closed control loop of a CD&R system, i.e., the state of the ownship and traffic aircraft are permanently monitored by the CD&R system for potential conflicts.

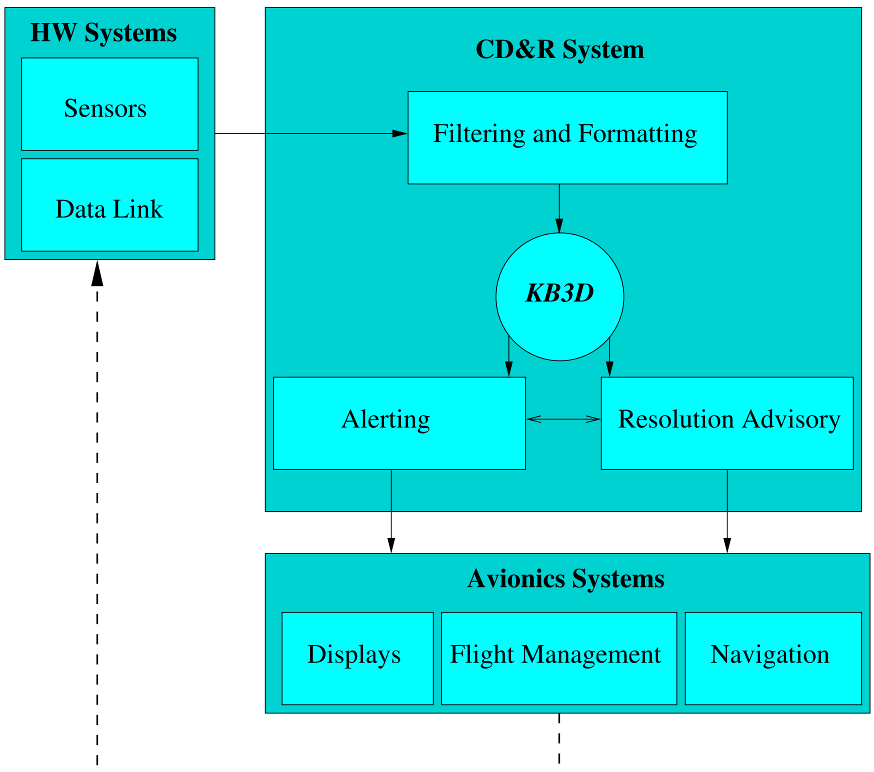


Figure 1.1: KB3D in a CR&R System

1.3 Implementation Issues

This manuscript documents *prototype implementations* written in C++ and Java of the KB3D algorithm. It is important to note that the correctness of the implementations has not been formally verified. Although the prototypes have been carefully reviewed with respect to the functional specification of the algorithm, there are fundamental differences between the actual code and the algorithm. These differences may result in unexpected behaviors. In this section we examine some of these issues.

Units and Coordinate System

The conflict detection and resolution logics in the KB3D algorithm are unit independent. However, they assume that the inputs are given in a 3 dimensional Cartesian coordinate system. The KB3D prototypes provide basic transformation formulas from Geodesic coordinates (latitude, longitude, and altitude) to Rectangular coordinates [10]. These formulas fail in the vicinity of either pole and at large distances. The prototypes also implement unit conversions from nautical miles and kilofeet to kilometers, and from knots and kilofeet per minute to kilometers per second.

The algorithms for coordinate transformations and unit conversions are not part of the KB3D algorithm. Therefore, their correctness have not been formally verified. They are provided as a convenience and they are expected to be reimplemented to satisfy specific user needs.

Computation Errors and Uncertainties

The KB3D algorithm assumes that all computations are exact. This assumption is clearly unsatisfiable by any implementation as machine numbers are approximations of real numbers. Due to numerical errors, resolutions computed by the KB3D prototypes are approximations of theoretical solutions.

In any case, computation errors are negligible compared to the uncertainties introduced by the accuracy of the measurement of aircraft position and velocity, pilot response, maneuverability time, communication delays, and the actual trajectory flown by the aircraft. For simplicity and efficiency, the KB3D algorithm, as most state based algorithms, assumes accurate measurement, immediate resolution response, perfect communication, and straight line trajectories in a flat earth geometry.

All these errors and uncertainties may lead to loss of separation. The severity of these violations is mitigated by an appropriate configuration of the protected zone and the look-ahead time. Usually, D is 5 nautical miles, H is 1000 feet, and T is 5 minutes. However, in KB3D, these parameters are configurable.

Actual Coordination

Effective coordination is achieved by KB3D assuming that both aircraft have the same view of the conflict situation. Due to multiple sources of surveillance data, data errors, communication delays and interruptions, and computational errors, the information that two aircraft use to detect and solve an actual conflict may be different. Different state data could cause the resolutions produced by KB3D to no longer be coordinated. Symmetrical and near symmetrical encounters are specially sensitive to surveillance and other errors. For example, an aircraft will turn right because the intruder aircraft is perceived to be left of its center line. If the intruder is actually right of the center line, the maneuver will exacerbate the conflict and coordination will not be achieved.

Whether or not this condition represents a problem in an operational environment depends on implementation issues. The source of surveillance data, accuracy, consistency, and integrity will play a role, in addition to encounter geometry probabilities and other operational factors.

Chapter 2

Application Program Interface

Prototypes of KB3D have been implemented in C++ and Java. The C++ package is called `KB3D++` and the Java package is called `KB3Dj`. Except for language differences, there is a one to one correspondence between both codes. Each package contains the following modules:

KB3D++	**KB3Dj**	**Description**
`CD3D`	`CD3D`	Conflict detection algorithm
`KB3D`	`KB3D`	Conflict resolution algorithm
`Geodesic`	`Geodesic`	Functions for transforming geodesic to rectangular coordinates
`util`	`Util`	Utility and unit conversion functions
`check`	`Check`	Functions for validation of KB3D resolutions
`main`	`Main`	Main function

Remark: The modules `CD3D`, `KB3D`, and `Geodesic` are implemented as classes and, therefore, can be reused or integrated to other applications. These modules require the utility and conversion functions defined in `util` in `KB3D++` and `Util` in `KB3Dj`.

2.1 Units and Coordinate System

`CD3D` and `KB3D` are unit independent. However, distance, time, and speed values must be given in a consistent way. Angles are given in radians in True North clockwise convention.

The KB3D's CD&R logic uses a relative Euclidean coordinate system where the intruder is at the origin, the axis x points to the East, and the axis y points to the North. Let $\mathbf{s}_o$, $\mathbf{v}_o$, and $\mathbf{s}_i$, $\mathbf{v}_i$, be the 3-D position and velocity vector of the ownship and intruder, respectively. The *relative position* of the ownship is $\mathbf{s} = \mathbf{s}_o - \mathbf{s}_i$. Similarly, the *relative velocity* of the ownship is $\mathbf{v} = \mathbf{v}_o - \mathbf{v}_i$.

Remark: KB3D assumes a flat earth geometry. The class `Geodesic` provides the functionality needed to transform geodesic coordinates to rectangular coordinates suitable for KB3D.

In the figures presented in the next sections, continuous arrows represent floating point values, whereas dashed arrows represent Boolean values. Boxes with square corners are

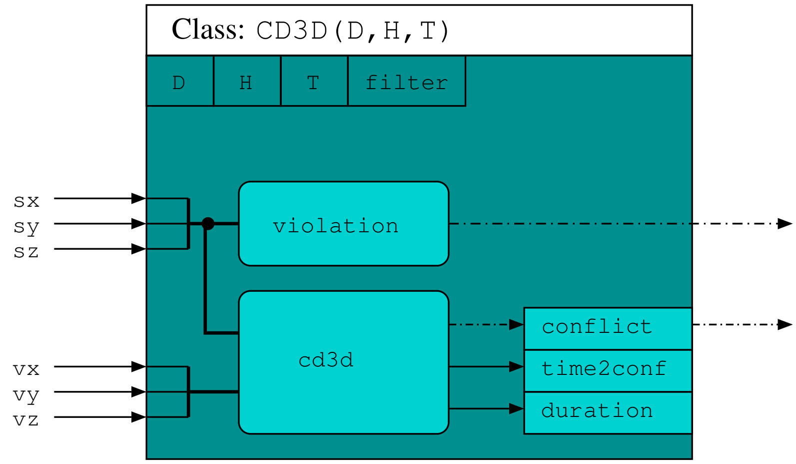


Figure 2.1: Class `CD3D`

fields and boxes with round corners are methods. A lighter background is used for public elements. Furthermore, private fields can be accessed and modified with functions of the form **get_*field*** and **set_*field***, respectively. Public output fields can be accessed directly, but they must be used in read-only mode.

2.2 CD3D

The class `CD3D` implements the conflict detection algorithm. Figure 2.1 illustrates the interface of the class.

Input fields

Field	Type	Description	Initial value
`D`	`double` (> 0)	Diameter of protected zone	Given at object creation
`H`	`double` (> 0)	Height of protected zone	Given at object creation
`T`	`double` (> 0)	Lookahead time	Given at object creation
`filter`	`double` (≥ 0)	Minimum duration time of a conflict	0

Input parameters

Parameter	Type	Description	Used by
`sx`	`double`	The x-component of **s**	`violation`, `cd3d`
`sy`	`double`	The y-component of **s**	`violation`, `cd3d`
`sz`	`double`	The z-component of **s**	`violation`, `cd3d`
`vx`	`double`	The x-component of **v**	`cd3d`
`vy`	`double`	The y-component of **v**	`cd3d`
`vz`	`double`	The z-component of **v**	`cd3d`

Output fields

Field	Type	Description	Written by
`conflict`	`bool`	True when a conflict has been detected	`cd3d`
`time2conf`	`double` $(0 \ldots \mathtt{T})$	Time to conflict	`cd3d`
`duration`	`double` $(\geq \mathtt{filter})$	Duration of conflict	`cd3d`

Remark: The values of `time2conf` and `duration` are undefined when `conflict` is false.

Methods

Method	:	`violation(sx,sy,sz)`
Output type	:	`bool`

Returns true if the aircraft are in violation.

Method	:	`cd3d(sx,sy,sz,vx,vy,vz)`
Output type	:	`bool`
Output fields	:	`conflict`, `time2conf`, `duration`

Returns `conflict`=true if the aircraft are in conflict, `time2conf` less than `T`, and `duration` is greater than `filter`. If the aircraft are in violation, `time2conf` is 0 and `duration` is the time remaining to the end of violation.

Remark: Conflicts that last less than `filter` are disregarded.

2.3 KB3D

The class `KB3D` implements the conflict resolution algorithm. Figure 2.2 illustrates the interface for coordinated horizontal and vertical resolutions. Figures 2.3 and 2.4 illustrate the interface for independent horizontal and vertical resolutions, respectively.

Input fields

Field	Type	Description	Initial value
`D`	`double` (> 0)	Diameter of protected zone	Given at object creation
`H`	`double` (> 0)	Height of protected zone	Given at object creation
`T`	`double` (> 0)	Lookahead time	Given at object creation
`coord`	`int` $(\pm\, 1)$	Coordination strategy	1

Remark: KB3D implements two coordination strategies corresponding to values of `coord` of 1 and -1. The value 1 is the default and it corresponds to the coordination strategy described in detail in [4]. All aircraft using KB3D's CD&R logic must set this parameter to the same value.

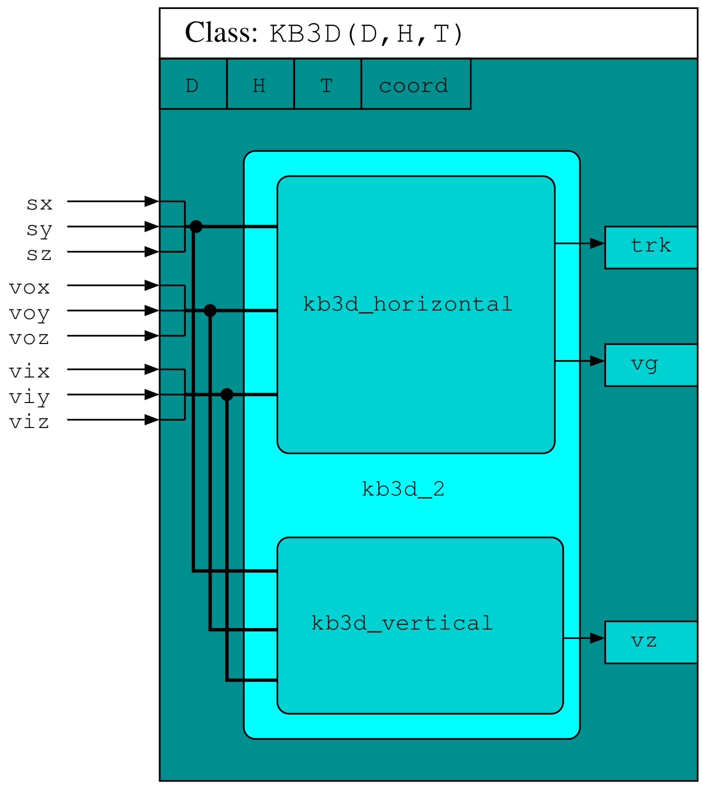


Figure 2.2: Class `KB3D`: Coordinated resolutions

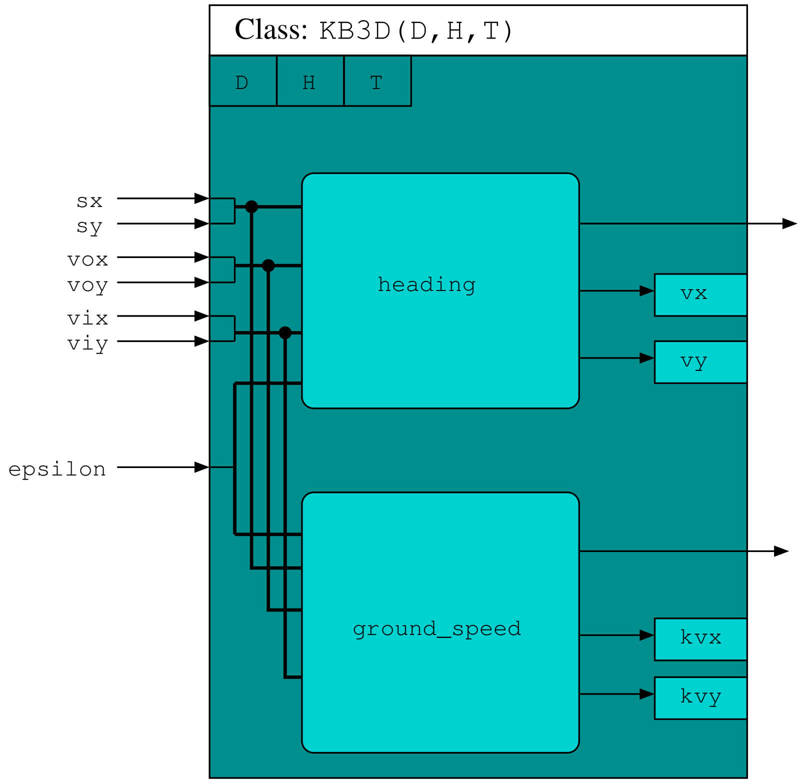


Figure 2.3: Class KB3D: Independent horizontal resolutions

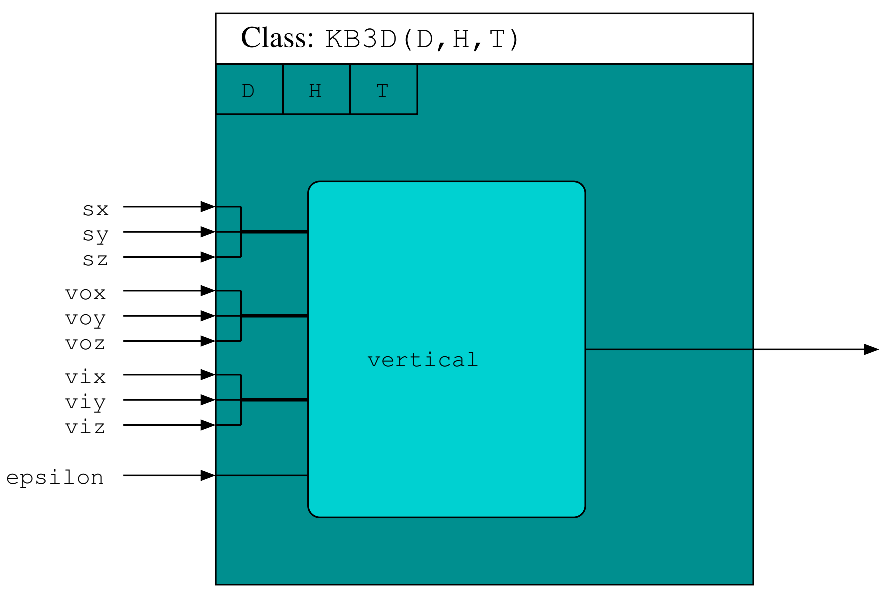


Figure 2.4: Class KB3D: Independent vertical resolutions

Input parameters

Parameter	Type	Description	Used by
sx	double	The x-component of $\mathbf{s}$	kb3d_horizontal, kb3d_vertical, heading, ground_speed, vertical
sy	double	The y-component of $\mathbf{s}$	kb3d_horizontal, kb3d_vertical, heading, ground_speed, vertical
sz	double	The z-component of $\mathbf{s}$	kb3d_horizontal, kb3d_vertical, vertical
vox	double	The x-component of $\mathbf{v}_o$	kb3d_horizontal, kb3d_vertical, heading, ground_speed, vertical
voy	double	The y-component of $\mathbf{v}_o$	kb3d_horizontal, kb3d_vertical, heading, ground_speed, vertical
voz	double	The z-component of $\mathbf{v}_o$	kb3d_horizontal, kb3d_vertical, vertical
vix	double	The x-component of $\mathbf{v}_i$	kb3d_horizontal, kb3d_vertical, heading, ground_speed, vertical
viy	double	The y-component of $\mathbf{v}_i$	kb3d_horizontal, kb3d_vertical, heading, ground_speed, vertical
viz	double	The z-component of $\mathbf{v}_i$	kb3d_horizontal, kb3d_vertical, vertical
epsilon	int (± 1)	Parameter for independent resolutions	heading, ground_speed,vertical

Output fields

Field	Type	Description	Written by
trk	double	Heading only resolution	kb3d_horizontal
vg	double (> 0)	Ground speed only resolution	kb3d_horizontal
vz	double	Vertical speed only resolution	kb3d_vertical
vx	double	The x-component of velocity vector of a heading only resolution	heading, kb3d_horizontal
vy	double	The y-component of velocity vector of a heading only resolution	heading, kb3d_horizontal
kvx	double	The x-component of velocity vector of a ground speed only resolution	ground_speed, kb3d_horizontal
kvy	double	The y-component of velocity vector of a ground speed only resolution	ground_speed, kb3d_horizontal

Methods

Method	:	`kb3d_2(sx,sy,sz,vox,voy,voz,vix,viy,viz)`
Output type	:	`void`
Output fields	:	`trk`, `vg`, `vz`, `vx`, `vy`, `kvx`, `kvy`

Provides *coordinated* and *independent* horizontal and vertical resolution maneuvers for the ownship (via `kb3d_horizontal` and `kb3d_vertical`).

Method	:	`kb3d_horizontal(sx,sy,sz,vox,voy,voz,vix,viy,viz)`
Output type	:	`void`
Output fields	:	`trk`, `vg`, `vx`, `vy`, `kvx`, `kvy`

If `trk` $\neq$ `NaN`, `trk` is a coordinated heading only resolution for the ownship. If `vg` $\neq$ `NaN`, `vg` is a coordinated ground speed only resolution for the ownship. The horizontal components of the velocity vectors for the heading only maneuver and ground speed only maneuvers are `(vx,vy)` and `(kvx,kvy)`, respectively.

Method	:	`kb3d_vertical(sx,sy,sz,vox,voy,voz,vix,viy,viz)`
Output type	:	`void`
Output fields	:	`vz`

If `vz` $\neq$ `NaN`, `vz` is a coordinated vertical only resolution for the ownship.

Method	:	`heading(sx,sy,vox,voy,vix,viy,epsilon)`
Output type	:	`double`
Output fields	:	`trk`, `vx`, `vy`

Returns either `NaN` or an independent heading resolution for the ownship. The horizontal components of the velocity vector for the resolution maneuver are `(vx,vy)`.

Method	:	`ground_speed(sx,sy,vox,voy,vix,viy,epsilon)`
Output type	:	`double`
Output fields	:	`vg`, `kvx`, `kvy`

Returns either `NaN` or an independent ground speed resolution for the ownship. The horizontal components of the velocity vector for the resolution maneuver are `(kvx,kvy)`.

Remark: `heading` and `ground_speed` return *independent* horizontal resolutions for each value of `epsilon`, i.e., ± 1. These resolutions are *coordinated* only if the ownship and the intruder use the same value of `epsilon` (see [4]).

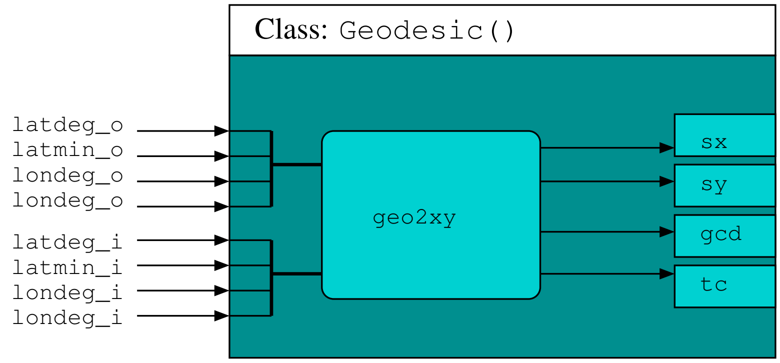


Figure 2.5: Class `Geodesic`

Method	:	`vertical(sx,sy,sz,vox,voy,voz,vix,viy,viz,epsilon)`
Output type	:	`double`

Returns either `NaN` or an independent vertical speed resolution for the ownship.

Remark: `vertical` returns *independent* vertical resolutions for each value of `epsilon`, i.e., ± 1. These resolutions are *coordinated* only if the ownship and the intruder use different values of `epsilon` (see [4]).

2.4 Geodesic

The class `Geodesic` implements coordinate transformations from a geodesic to rectangular coordinates using formulas from [10]. These transformations assume a flat earth. Therefore, they fail in the vicinity of either pole and at large distances. Figure 2.5 illustrates the interface of the class.

Remark: Longitudes are US-centric, i.e., West longitudes are positive.

Input parameters

Parameter	Type	Description	Used by
`latdeg_o`	`int` $(-80\ldots 80)$	Ownship latitude (degrees)	`geo2xy`
`latmin_o`	`double` $(0\ldots 60)$	Ownship latitude (minutes)	`geo2xy`
`londeg_o`	`int` $(0\ldots 360)$	Ownship longitude (degrees)	`geo2xy`
`lonmin_o`	`double` $(0\ldots 60)$	Ownship longitude (minutes)	`geo2xy`
`latdeg_i`	`int` $(-80\ldots 80)$	Intruder latitude (degrees)	`geo2xy`
`latmin_i`	`double` $(0\ldots 60)$	Intruder latitude (minutes)	`geo2xy`
`londeg_i`	`int` $(0\ldots 360)$	Intruder longitude (degrees)	`geo2xy`
`lonmin_i`	`double` $(0\ldots 60)$	Intruder longitude (minutes)	`geo2xy`

Output fields

Field	Type	Description	Written by
`sx`	`double`	The x-component of $\mathbf{s}$ (meters)	`geo2xy`
`sy`	`double`	The y-component of $\mathbf{s}$ (meters)	`geo2xy`
`gcd`	`double`	The great circle distance between the aircraft (meters)	`geo2xy`
`tc`	`double`	The relative course at current position (radians)	`geo2xy`

Methods

Method : `geo2xyz(latdeg_o,latmin_o,londeg_o,lonmin_o,`
`latdeg_i,latmin_i,londeg_i,lonmin_i)`
Output type : `void`
Output fields : `sx`, `sy`, `gcd`, `tc`

Transforms the geodesic coordinates of the ownship and intruder to a relative Cartesian system (assuming flat earth).

2.5 Utility and Conversion Functions

Constant : `NaN`
Type : `double`

An arbitrary large number used for exceptional cases.

Function : `rad2deg(double x)`
Type : `double`

Converts radians to degrees.

Function : `deg2rad(double x)`
Type : `double`

Converts degrees to radians.

Function : `degmin2rad(int d, double m)`
Type : `double`

Converts degrees and minutes to radians.

Function : `nm2m(double x)`
Type : `double`

Converts nautical miles to meters.

Function : `m2nm(double x)`
Type : `double`

Converts meters to nautical miles.

Function : `knots2msec(double x)`
Type : `double`

Converts knots to meters/second.

Function : `msec2knots(double x)`
Type : `double`

Converts meters/second to knots.

Function : `kft2m(double x)`
Type : `double`

Converts kfeet to meters.

Function : `m2kft(double x)`
Type : `double`

Converts meters to kfeet.

Function : `kftmin2msec(double x)`
Type : `double`

Converts kfeet/minute to meters/second.

Function : `msec2kftmin(double x)`
Type : `double`

Converts meters/second to kfeet/minute.

Function	:	`vg2vx(double vg,double trk)`
Type	:	`double`

Returns the x-component of the ground speed `vg` and heading `trk`. Note that `trk` is in radians and True North clockwise convention.

Function	:	`vg2vy(double vg,double trk)`
Type	:	`double`

Returns the y-component of the ground speed `vg` and heading `trk`. Note that `trk` is in radians and True North clockwise convention.

2.6 Example

Assume the following variables representing ownship and intruder aircraft information (West logitudes are positive and angles are in True North clockwise convention).

Variable	**Value**	**Units**	**Description**
Parameters			
`D`	0.5	nautical miles	Diameter of protected zone
`H`	0.5	kfeet	Heigth of protected zone
`T`	2	minutes	Lookahead time
Ownship			
`latdeg_o`	36	degrees	Latitude
`latmin_o`	59.87	minutes	Latitude
`londeg_o`	76	degrees	Longitude
`lonmin_o`	28.74	minutes	Longitude
`alt_o`	42	kfeet	Flight level
`trk_o`	91.22	degrees	Heading
`vg_o`	237	knots	Ground speed
`v_o`	0.0020	kfeet/minute	Vertical speed
Intruder			
`latdeg_i`	36	degrees	Latitude
`latmin_i`	52.5	minutes	Latitude
`londeg_i`	76	degrees	Longitude
`lonmin_i`	20.1	minutes	Longitude
`alt_i`	40.9	kfeet	Flight level
`trk_i`	-0.6	degrees	Heading
`vg_i`	252	knots	Ground speed
`v_i`	0.531	kfeet/minute	Vertical speed

Inputs are translated to values suitable for KB3D using the following code (in C++):

```
double Dm = nm2m(D);
double Hm = kft2m(H);
double Ts = T*60;

Geodesic geo = new Geodesic();
geo->geo2xy(latd_o,latm_o,lond_o,lonm_o,
            latd_i,latm_i,lond_i,lonm_i);

double sx = geo->sx;
double sy = geo->sy;
double sz = kft2m(alt_o-alt_i);

double vg,trk;

vg   = knots2msec(vg_o);
trk  = deg2rad(trk_o);

double vx_o = vg2vx(vg,trk);
double vy_o = vg2vy(vg,trk);
double vz_o = kftmin2msec(v_o);

vg   = knots2msec(vg_i);
trk  = deg2rad(trk_i);

double vx_i = vg2vx(vg,trk);
double vy_i = vg2vy(vg,trk);
double vz_i = kftmin2msec(v_i);
```

At this point, we have the following values (the axis x points to the East, the axis y points to the North):

Variable	**Value**	**Units**	**Description**
`Dm`	926	meters	Diameter of protected zone
`Hm`	152.367	meters	Height of protected zone
`Ts`	120	seconds	Lookahead time
`sx`	-12838.6	meters	The x-component of $\mathbf{s}$
`sy`	13631.5	meters	The y-component of $\mathbf{s}$
`sz`	335.208	meters	The z-component of $\mathbf{s}$
`vx_o`	121.896	meters/second	The x-component of $\mathbf{v}_o$
`vy_o`	-2.59592	meters/second	The y-component of $\mathbf{v}_o$
`vz_o`	0.0101578	meters/second	The z-component of $\mathbf{v}_o$
`vx_i`	-1.35756	meters/second	The x-component of $\mathbf{v}_i$
`vy_i`	129.633	meters/second	The y-component of $\mathbf{v}_i$
`vz_i`	2.6969	meters/second	The z-component of $\mathbf{v}_i$

Current loss of separation, conflict detection, and conflict resolutions (for the ownship) are computed with the following code (in C++). In this case, conflicts that last less than one second will be ignored.

```
bool loss  = CD3D.violation(sz,sy,sz);

CD3D* cd3d = new CD3D(Dm,Hm,Ts);
cd3d->set_filter(1);
cd3d->cd3d(sx,sy,sz,vx_o-vx_i,vy_o-vy_i,vz_o-vz_i);

KB3D* kb3d = new KB3D(Dm,Hm,Ts);
kb3d->kb3d_2(sx,sy,sz,vx_o,vy_o,vz_o,vx_i,vy_i,vz_i);
```

At this point, we have the following results:

Variable	**Value**	**Units**	**Description**
Conflict Detection			
`loss`	false		Aircraft are not in violation
`cd3d->conflict`	true		Aircraft are in predicted conflict
`cd3d->time2conf`	98.495	seconds	Time to conflict
`cd3d->duration`	10.1891	seconds	Conflict duration
Conflict Resolution (for ownship)			
`kb3d->vz`	1.01459	meters/second	Vertical speed resolution
`kb3d->vg`	111.207	meters/second	Ground speed resolution
`kb3d->kvx`	111.182	meters/second	The x-component of ground speed resolution
`kb3d->kvy`	-2.36776	meters/second	The y-component of ground speed resolution
`kb3d->trk`	1.68603	radians	Heading resolution
`kb3d->vx`	121.115	meters/second	The x-component of heading resolution
`kb3d->vy`	-14.018	meters/second	The y-component of heading resolution

These results can be translated back to the original units (in C++):

```
double vz_sol  = msec2kftmin(kb3d->vz);
double vg_sol  = msec2knots(kb3d->vg);
double trk_sol = rad2deg(kb3d->trk);
```

Finally, we have the following results:

Variable	**Value**	**Units**	**Description**
`vz_sol`	0.199765	kfeet/minute	Independent and cooperative vertical resolution
`vg_sol`	216.17	knots	Independent and cooperative ground speed resolution
`trk_sol`	96.6021	degrees	Independent and cooperative heading resolution

Chapter 3

User Interface

The command lines of both `KB3D++` and `KB3Dj` have the following options:

```
kb3d [-D d] [-H h] [-T t] [-version] [-check|-nocheck]
     [-opt|-nonopt] [-geo|-xyz] [-help] [file.geo|.xyz] ...
```

Diameter of protected zone.

`-D d`

Unit: nautical miles
Legal values: positive
Default value: 5 nm

Height of protected zone.

`-H h`

Unit: kfeet
Legal values: positive
Default value: 1 kfeet

Lookahead time.

`-T t`

Unit: minutes
Legal values: positive
Default value: 5 min

Display current version and exit.
`-version`
Default:

Perform runtime validation of outputs.
`-check | -nocheck`
Default: no

Use optimal coordinated strategy.
`-opt | -nonopt`
Default: yes

Assume geodesic coordinates.
`-geo`
Default: yes

Assume rectangular coordinates.
`-xyz`
Default: no

Display help screen.
`-help`
Default: no

KB3D processes, in sequential order, the data files given in the command line. Files with extension `.xyz` contains aircraft information in rectangular coordinates, whereas files with extension `.geo` contains aircraft information in geodesic coordinates. Each data file has two lines, one per aircraft information.

3.1 Rectangular coordinates

Aircraft information in rectangular coordinates has the form:

```
id x y z trk vg vz
```

Identification.

○ `id`

Unit: string
Legal values: nonempty string

x-component of aircraft position (East is positive).

○ `x`

Unit: nautical miles
Legal values: any

y-component of aircraft position (North is positive).

○ `y`

Unit: nautical miles
Legal values: any

z-component of aircraft position.

○ `z`

Unit: kfeet
Legal values: positive

True North track.

○ `trk`

Unit: degrees
Legal values: any

Ground speed.

○ `vg`

Unit: knots
Legal values: positive

Vertical speed.

○ `vz`

Unit: kfeet/minute
Legal values: any

3.2 Geodesic coordinates

Aircraft information in geodesic coordinates has the form:

```
id latd latm lond lonm alt trk vg vz
```

Remark: The values `id`, `trk`, `vg`, and `vz`, are the same as in rectangular coordinates.

Latitude degrees.

○ `latd`

Unit: degrees
Legal values: integer number

Latitude minutes.

○ `latm`

Unit: minutes
Legal values: any

Longitude degrees (West is positive).

○ `lond`

Unit: degrees
Legal values: integer number

Longitude minutes.

○ `lonm`

Unit: minutes
Legal values: any

Altitude.

○ `alt`

Unit: kfeet
Legal values: positive

Chapter 4

Examples

This chapter shows examples of the detection and resolution of KB3D for various encounter scenarios. The parameters used are

- D - protected zone diameter of 0.5 nautical miles horizontally.
- H - protected zone of 500 feet vertically.
- T - 2 minutes lookahead time.

The interactive command for this execution is:

```
kb3d -D 0.5 -H 0.5 -T 2
```

4.1 Head-on encounter

The first example is a head on configuration encounter. Aircraft N123QL has a ground speed of 200 knots, at flight level FL410, on level flight, and due south (bearing 180.13 degrees). Aircraft N456FT has a ground speed of 453 knots, FL410, at level flight, and due north (bearing 0.27 degrees). The aircraft are at approximately 15 nautical miles apart. Figure 4.1 shows the top and side views of the encounter. The conflict detection part of the algorithm detects a conflict 1.35 minutes from the current state. Assuming that the aircraft do not maneuver, the conflict detection predicts that the aircraft will be closer than the minimum separation required for a duration of 0.08 minutes.

The input:

```
N123QL 37 05.41 76 20.11 41 180.13 200 0.001
N456FT 36 50.28 76 19.89 41 0.27 453 0.003
```

produces the following output:

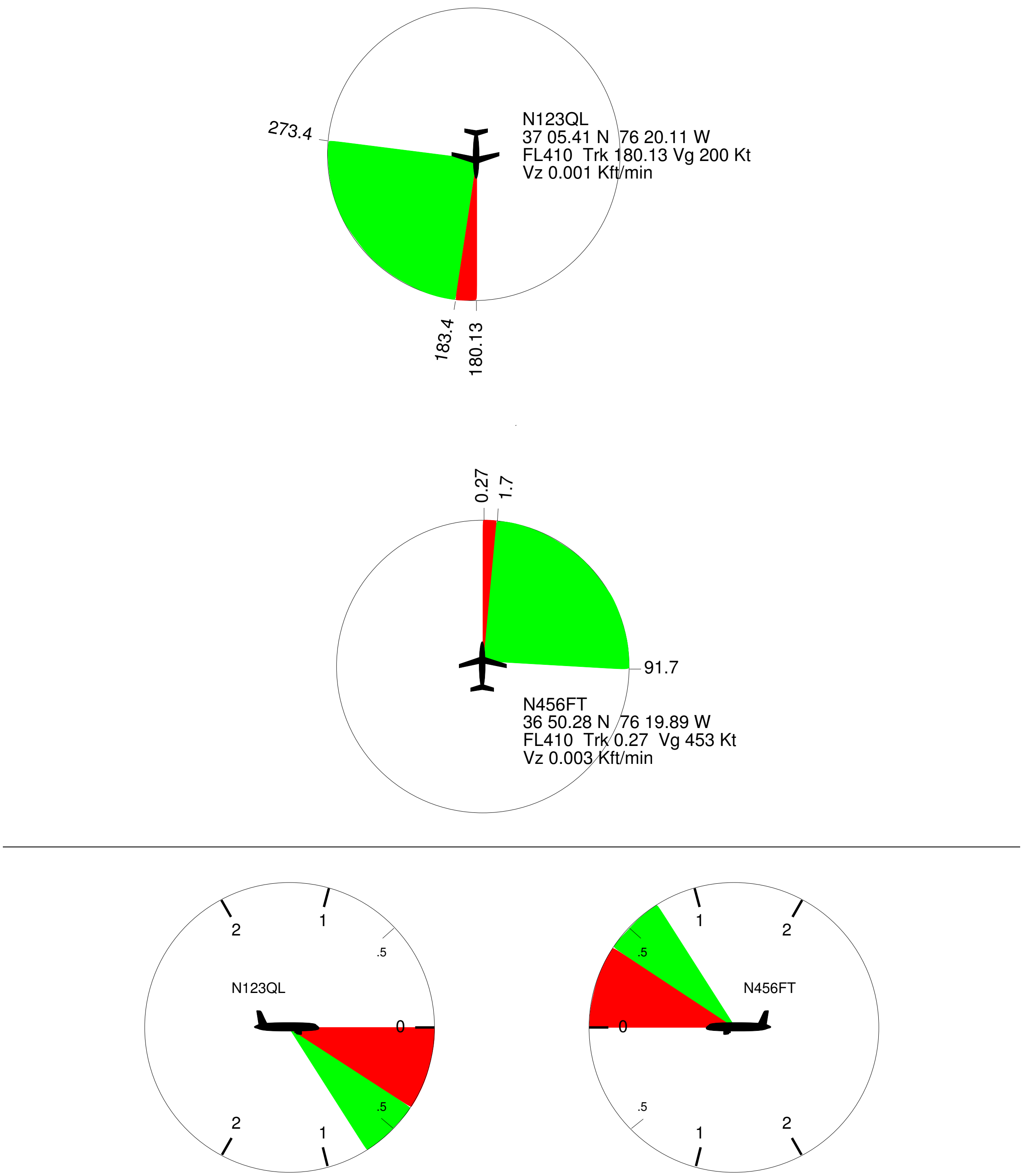


Figure 4.1: Top and side views of head-on encounter

```
CONSTANTS
---------
D = 0.5 [nm], H = 0.5 [kft], T = 2 [min]

INPUTS
------
N123QL
Lat: 37 [deg] 5.41 [min], Lon: 76 [deg] 20.11 [min], Alt: 41 [kft]
Trk: -179.87 [deg], Vg: 200 [knots], Vz: 0.001 [kft/min]

N456FT
Lat: 36 [deg] 50.28 [min], Lon: 76 [deg] 19.89 [min], Alt: 41 [kft]
Trk: 0.27 [deg], Vg: 453 [knots], Vz: 0.003 [kft/min]

Distance Great Circle : 15.131 [nm]
Relative Course : 179.333 [deg]

CD3D
----
Time to conflict: 1.34783 [min]
Duration of conflict: 0.0809555 [min]

KB3D Resolutions: N123QL
-----------------
Vertical Speed Only <= -0.367967 [kft/min]
Track Only = -176.607 [deg] (clockwise for coordinated resolution)

KB3D Resolutions: N456FT
-----------------
Vertical Speed Only >= 0.371967 [kft/min]
Track Only = 1.71062 [deg] (clockwise for coordinated resolution)
```

The vertical resolution maneuvers call for N123QL to descend at a rate of 368 feet per minute and for N456FT to climb at a rate of 372 feet per minute. The vertical resolution maneuvers are shown on a Vertical Speed Indicator graph in Figure 4.1.

The heading resolution maneuvers call for N123QL to turn right to a course of 183.4 degrees (-176.6 degrees) and for N456FT to turn right to a course of 1.7 degrees. This is shown in Figure 4.1 as the red sectors for conflict and the green sectors for resolution.

Remark: The conflict will be avoided if one of the aircraft implements a maneuvers or if both implement a maneuver.

For this encounter scenario, there is not a resolution maneuver for change in speed and the program gives non. That is, a change in the speed of the aircraft will not solve the conflict without heading or vertical change.

4.2 Crossing paths, one aircraft climbing

Aircraft N321QL and N654FT are approaching on perpendicular paths, one at level flight and one climbing. Aircraft N321QL has a ground speed of 237 knots, at flight level FL420, at level flight, and due east (bearing 91.22 degrees). Aircraft N654FT has a ground speed of 252 knots, at FL409, climbing at 531 feet per minute, and due north (bearing 359.40 degrees). A conflict is detected 1.64 minutes, for a duration of 0.17 minutes, from the current state. The aircraft are approximately 10 nautical miles apart and 7 miles form the collision point. Figure 4.2 shows the top and side views of the encounter.

The input:

```
N321QL 36 59.87 76 28.74 42 91.22 237 0.002
N654FT 36 52.50 76 20.10 40.9 359.40 252 0.531
```

produces the following output:

```
CONSTANTS
---------
D = 0.5 [nm], H = 0.5 [kft], T = 2 [min]

INPUTS
------
N321QL
Lat: 36 [deg] 59.87 [min], Lon: 76 [deg] 28.74 [min], Alt: 42 [kft]
Trk: 91.22 [deg], Vg: 237 [knots], Vz: 0.002 [kft/min]

N654FT
Lat: 36 [deg] 52.5 [min], Lon: 76 [deg] 20.1 [min], Alt: 40.9 [kft]
Trk: -0.6 [deg], Vg: 252 [knots], Vz: 0.531 [kft/min]

Distance Great Circle : 10.1 [nm]
Relative Course : 136.818 [deg]

CD3D
----
Time to conflict: 1.64158 [min]
Duration of conflict: 0.169819 [min]

KB3D Resolutions: N321QL
-----------------
Vertical Speed Only >= 0.199765 [kft/min]
Ground Speed Only <= 216.17 [knots]
Track Only = 96.6021 [deg] (clockwise for coordinated resolution)
```

```
KB3D Resolutions: N654FT
-----------------
Vertical Speed Only <= 0.333235 [kft/min]
Ground Speed Only >= 276.283 [knots]
Track Only = 4.17997 [deg] (clockwise for coordinated resolution)
```

There are three resolution maneuvers for this encounter: vertical speed, ground speed, and track. The vertical speed maneuver is for aircraft N321QL to climb at 200 feet per minute and for aircraft N654FT to reduce its climb rate to less than 333 feet per minute.

The ground speed maneuver requires aircraft N321QL to reduce its speed from 237 knots to 216 knots or less and for N654FT to increase its speed from 252 to 277 knots or more.

The heading resolution is for aircraft N321QL to change its course to a bearing of 96.6 degrees and for aircraft N654FT to change its course to a bearing of 4.2 degrees. The resolutions are illustrated in Figure 4.2.

Remark: The conflict will be avoided if one of the aircraft implements a maneuvers or if both implement a maneuver.

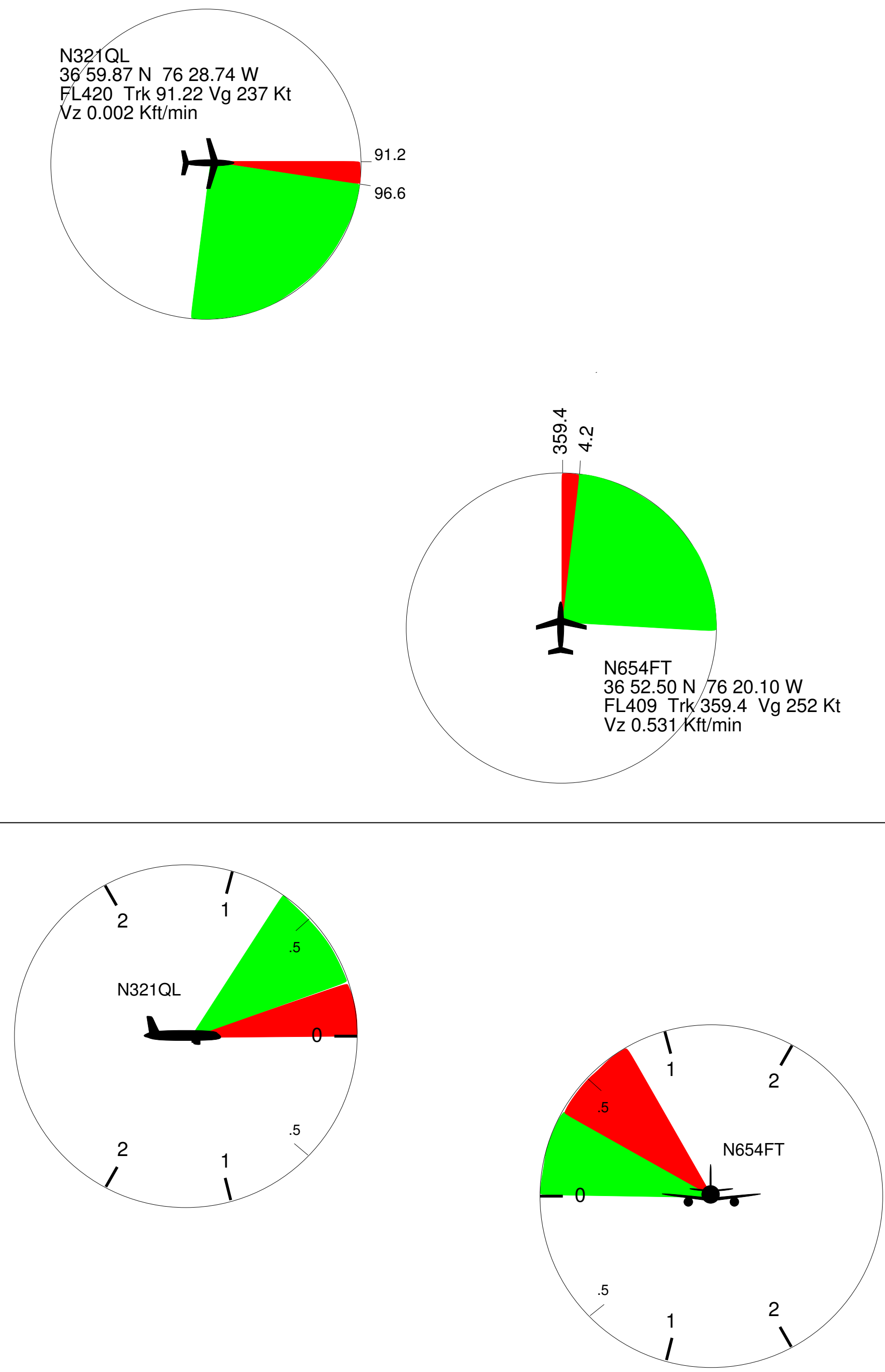


Figure 4.2: Top and side views of crossing paths, one aircraft climbing encounter

Index

Bibliography

[1] K. Bilimoria. A geometric optimization approach to aircraft conflict resolution. In *Guidance, Navigation, and Control Conference*, volume AIAA 2000-4265, Denver, CO, August 2000.

[2] R. Butler, A. Geser, J. Maddalon, and C. Muñoz. Formal analysis of air traffic management systems: The case of conflict resolution and recovery. In *Proceedings of the Winter Simulation Conference WSC'03*, pages 906–914, New Orleans, LA, December 2003. A long vesion appears as NASA/TP-2004-213015.

[3] G. Dowek, A. Geser, and C. Muñoz. Tactical conflict detection and resolution in 3-D airspace. In *4th USA/Europe Air Traffic Management R&D Seminar (ATM-2001)*, Santa Fe, New Mexico, 2001. Extended version available as ICASE Report No. 2002-12 NASA/CR-2002-211637.

[4] G. Dowek, C. Muñoz, and V. Carreño. Provably safe coordinated strategy for distributed conflict resolution. In *Proceedings of the AIAA Guidance Navigation, and Control Conference and Exhibit 2005, AIAA-2005-6047*, San Francisco, California, 2005.

[5] A. Geser and C. Muñoz. A geometric approach to strategic conflict detection and resolution. In *Proceedings of the 21st Digital Avionics Systems Conference*, Irvine, CA, 2002.

[6] A. Geser, C. Muñoz, G. Dowek, and F. Kirchner. Air traffic conflict resolution and recovery. Technical Report ICASE Report No. 2002-12 NASA/CR-2002-211637, ICASE-NASA Langley, ICASE Mail Stop 132C, NASA Langley Research Center, Hampton VA 23681-2199, USA, May 2002.

[7] J. Hoekstra, R. Ruigrok, R. van Gent, J. Visser, B. Gijsbers, M. Valenti, W. Heesbeen, B. Hilburn, J. Groeneweg, and F. Bussink. Overview of NLR free flight project 1997-1999. Technical Report NLR-CR-2000-227, National Aerospace Laboratory (NLR), May 2000.

[8] J. Maddalon, R. Butler, A. Geser, and C. Muñoz. Formal verification of a conflict resolution and recovery algorithm. Technical Report NASA/TP-2004-213015, NASA Langley Research Center, NASA LaRC,Hampton VA 23681-2199, USA, April 2004.

[9] S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, June 1992. Springer-Verlag.

[10] E. Williams. Aviation formulary V1.42. Available from `http://williams.best.vwh.net/avform.htm`.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*
01- 06 - 2005

2. REPORT TYPE
Technical Memorandum

3. DATES COVERED *(From - To)*

4. TITLE AND SUBTITLE
KB3D Reference Manual – Version 1.a

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Munoz, Cesar; Siminiceanu, Radu; Carreno, Victor A.; and Dowek, Gilles

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
23-137-10-10

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER
L-19132

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)
NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
NASA/TM-2005-213769

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 03
Availability: NASA CASI (301) 621-0390

13. SUPPLEMENTARY NOTES
An electronic version can be found at http://ntrs.nasa.gov

14. ABSTRACT
This paper is a reference manual describing the implementation of the KB3D conflict detection and resolution algorithm. The algorithm has been implemented in the Java and C++ programming languages. The reference manual gives a short overview of the detection and resolution functions, the structural implementation of the program, inputs and outputs to the program, and describes how the program is used. Inputs to the program can be rectangular coordinates or geodesic coordinates. The reference manual also gives examples of conflict scenarios and the resolution outputs the program produces.

15. SUBJECT TERMS
Conflict detection; Coordinated resolution; Distributed CD&R; Air Traffic Management

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON / 19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	39	STI Help Desk (email: help@sti.nasa.gov) / (301) 621-0390