8 TARGET MANOEUVRABILITY AND MISSILE GUIDANCE AND CONTROL MODELLING

8.1 Introduction

This chapter explains the mathematical modelling of the target manoeuvrability and the missile tracking, guidance and control algorithm. The inputs required for modelling are explained. During the target-missile engagement simulation, the target position and direction is calculated at every time interval or frame as per the selected mode or manoeuvre. The missile movement and rotation are calculated as per the relative position of the target and the aerodynamic limits of the missile. The missile tracking logic is implemented with a gimballed seeker head.

8.2 VR World Fields for Target and Missile Movement

As explained in Chapter-7, the virtual reality (VR) world scenario is built using several objects such as the missile, target aircraft, flares, background, sky and atmosphere etc. All these objects are placed in world coordinates at respective positions and directions as per the scenario. Out of these objects, the missile, the target (aircraft) and flares are the only objects that are dynamic in nature. During simulation these objects are moving and changing their directions. To control the movement of these objects in the virtual world, their "*translation*" and "*rotation*" fields are linked with the main algorithm in MATLAB. The target "*translation*" and "*rotation*" fields are given in Equation 8-1 and 8-2 respectively. Figure 8-1 illustrates the target location and direction in 3D world coordinates.

$$myworld.target.translation = [X_{tgt}, Y_{tgt}, Z_{tgt}] \quad \textbf{(8-1)}$$

$$myworld.target.rotation = [0, 1, 0, \phi_{tgt}] \quad \textbf{(8-2)}$$

The parameters of translation and rotation fields depend upon the target flight path or the manoeuvrability. These parameters are explained in detail in subsequent paragraphs.

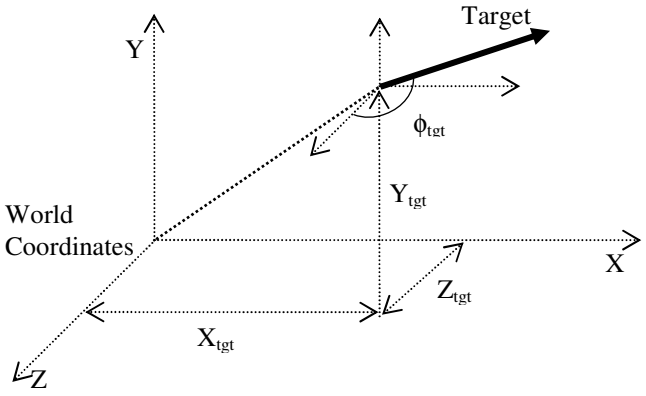


Figure 8-1 : Target translation and rotation Fields

Similarly, the missile movement is controlled by the "*translation*" and "*rotation*" fields which need to be upgraded regularly to chase the target. Equation 8-3 to 8-5 shows the translation and rotation fields related to the missile movement.

$$myworld.missile.translation = [X_{msl}, Y_{msl}, Z_{msl}] \quad \textbf{(8-3)}$$

$$myworld.missile\_LOS.rotation = [X_{m\_LOS}, Y_{m\_LOS}, Z_{m\_LOS}, \phi_{msl\_LOS}] \quad \textbf{(8-4)}$$

$$myworld.gimbal.rotation = [X_{m\_gimbal}, Y_{m\_gimbal}, Z_{m\_gimbal}, \phi_{gimbal}] \quad \textbf{(8-5)}$$

Unlike the target which has only one rotation field, the missile has two independent rotation fields. One for controlling the direction of the missile body or the "*missile-LOS*" as given in Equation 8-4 and the other is the missile gimballed seeker head which is denoted as "*gimbal*" in Equation 8-5. Figure 8-2 shows the "*missile_LOS*" and "*gimbal*" position and direction fields. The gimbal rotation is independent of the missile_LOS direction and may look away from missile direction of motion.

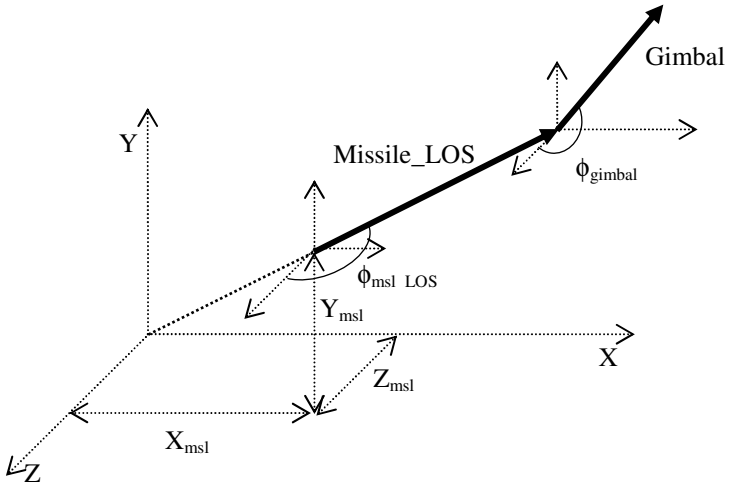


Figure 8-2 : Missile and gimbals translation and rotation Fields

8.3 Assumptions for Missile and Target Movement

For the missile-target engagement simulation, the following assumptions are made regarding the movement of the target and the missile in the 3D virtual world.

8.3.1 Target Degree-of-freedom

It is assumed that the target aircraft can move in four degrees-of-freedom (4-DOF). That means the target may move freely in three directions along the X-, Y- and Z-axes and can also perform "*yaw*" movement. The *yaw* is rotation around a vertical axis and therefore acts in a horizontal plane. However, the "*pitch*" and "*roll*" movement are not considered. Figure 8-3 illustrates the 6-DOF of an aircraft. Although, the 6-DOF is essentially required for modelling the aircraft movements in real scenarios, however, to keep the algorithm simple and due to the shortage of the time, the aircraft *pitch* and *roll* movements are not incorporated in the algorithm. The *pitch* and *roll* movement may be modelled in future by incorporating changes in the aircraft manoeuvrability algorithm.

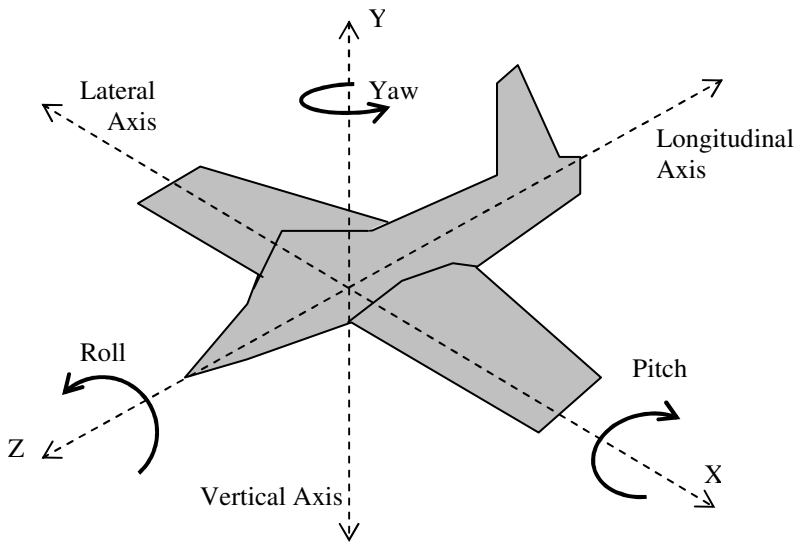


Figure 8-3 : Aircraft six degree-of-freedom

8.3.2 Missile Degree-of-freedom

It is assumed that the missile can perform 5-DOF, which means the missile may move along X, Y and Z axes and may also perform "*yaw*" and "*pitch*" movement along "*vertical*" and "*lateral*" axis respectively. However, the "*roll*" movement is not incorporated in the algorithm and is left on this assumption that it may not have any direct effect on the target movement and IR signature of the scene.

8.3.3 Targets Initial Position

The missile seeker relies on the target relative position (X_{tgt}, Y_{tgt}, Z_{tgt}) in the virtual world coordinates. This information is given to the missile launcher at the start of the simulation about the target aircraft. As it may be realistic to assume that in a real world scenario, the missile has access to similar information through some other radar, TV or visual search system.

8.3.4 Information about Missile Launch

The information about the time, location and direction of the missile launch is required by the target aircraft to initiate appropriate countermeasures such as dispensing flares and taking evasive manoeuvres. Although in realistic systems this information is generally provide by the Missile Approach Warning System (MAWS) or any other missile detection system. Although modelling MAWS is necessary for simulating true target missile engagement scenarios and the countermeasures analysis,

however, presently at this stage it is assumed that the only information regarding missile launch provided to the target aircraft is the time in seconds after the simulation has started and the direction of the missile approach so that the target aircraft may initiate countermeasures and manoeuvres accordingly.

8.4 User Inputs and Typical Data Ranges

To simulate any specific real world scenario of the missile-target engagement, several parameters are required as inputs. These inputs are fed in using either the "*menu*" command of MATLAB [MAT07], or Microsoft Excel spreadsheet or manually in the main m-file of MATLAB. Table 8-1 lists the inputs required for the simulation and also their typical ranges. The typical ranges are based on the open source data or interpolated from system brochures and do not represent classified data of any missile or aircraft.

Table 8-1 : Typical input data ranges

PARAMETER	SYMBOL	TYPICAL RANGE	REMARKS
Missile speed in Mach number	M_{msl}	Mach # 1 to 4	For aerial targets
Seeker refresh-rate	Ref_rate	100-125 Hz	
Seeker frame-rate	fps	30-70 frame/sec	May be same as refresh-rate
Seeker field-of-view	FOV_{gimbal}	1-2 degree	
Missile Load factor	$G's_{msl}$	< 40 G's	For lateral acceleration
Hit criteria (impact distance)	R_{impact}	<2 meters	Cylindrical region in front of missile
Seeker detection criterion	threshold	Based on detector type	for centriod go-no-go criterion
Seeker head gimbal turn limit	θ_{gimbal_max}	± 45 deg to ±90 deg	Gimbal seeker head rotate limit
Missile initial location	$(X_{msl}, Y_{msl}, Z_{msl})$	(0, 0, 0)	Initially keeping missile at origin of virtual world
Missile initial rotation	$(X_{m\_LOS}, Y_{m\_LOS}, Z_{m\_LOS}, \phi_{msl\_LOS})$	(0,1,0,0)	Initially missile looking along negative Z-axis
Gimbal initial rotation	$(X_{m\_gimbal}, Y_{m\_gimbal}, Z_{m\_gimbal}, \phi_{gimbal})$		Looking towards target aircraft location
Target Speed in Mach number	M_{tgt}	Mach # 0.4 to 2.2	
Target Load factor	$G's_{tgt}$	< 9 G's	
Target rate-of-descent	ROD	In x1000 of meters/sec	
Target initial position	$(X_{tgt\_start}, Y_{tgt\_start}, Z_{tgt\_start})$	Any point in front of missile $Z_{tgt} < Z_{msl}$	to be within seeker head FOV and detection range
Target initial direction	$(0, 1, 0, \phi_{tgt})$		As per selected mode
Target manoeuvre Mode		12 modes	Select one mode

8.5 An Aircraft in a Level Turn

The level flight of the aircraft is when it is flying with constant velocity along a straight line. However, in a realistic world the aircraft apply radial acceleration which leads to the movement of aircraft along a curved path. There are typically three cases of turned flight: a level-turn, a pull-up and a pull-down. For the purpose of the IR signature analysis, presently, the level-turn along the horizontal plane is considered for modelling, however, the pull-up and pull-down movement may be modelled in future by making additions in the algorithm. A level-turn is illustrated in Figure 8-4. Figure 8-4(a) shows the top view of the complete level-turn circle and Figure 8-4(b) shows the same from the front view. The distance from the centre of the circle to the aircraft is the turn-radius (*TR*) which is the tightness of the turn circle. The angular velocity (ω) which is the rate-of-change of the angle (θ) per unit time. The angular velocity (ω) is also called is the rate-of-turn (*ROT*). The ROT shows how fast the aircraft can get around the turn. From Figure 8-4(a) the ROT may be explained as given in Equation 8-6.

$$ROT = \frac{d\theta}{dt} = \frac{V}{TR} \qquad \textbf{(8-6)}$$

Where, V is the velocity of the aircraft along the direction of motion or LOS.

TR is the turn-radius

From Figure 8-4 (b) the resultant force (F_r) applying along the horizontal axis due to the radial acceleration may be given as in Equation 8-7 [AND00].

$$F_r = \frac{W \cdot V^2}{g \cdot TR} \qquad \textbf{(8-7)}$$

Where, W is the weight of the aircraft,

g is the acceleration due to gravity,

The resultant force (F_r) may also be expressed in terms of lift (L) and weight (W) as given in Equation 8-8.

$$F_r = \sqrt{L^2 - W^2} \qquad \textbf{(8-8)}$$

Let n = L/W, which is the load-factor of the aircraft or also know as G's force [AND00]. Then Equation 8-8 may be written in form of the load-factor as given in Equation 8-9.

$$F_r = W\sqrt{G's^2 - 1} \qquad \textbf{(8-9)}$$

Relating Equation 8-7 and 8-9 and simplifying for *TR*. The turn-radius may be calculated as given in Equation 8-10 [AND00].

$$TR = \frac{V^2}{g\sqrt{G's^2 - 1}} \qquad \textbf{(8-10)}$$

By putting the value of *TR* in Equation 8-6, the ROT may be expressed as given in Equation 8-11 [AND00].

$$ROT = \frac{g\sqrt{G's^2 - 1}}{V} \qquad \textbf{(8-11)}$$

In turn flight, the wing of the aircraft bank through an angle (ϕ) which is called the bank-angle [AND00]. From Figure 8-4(b) the forces applying along the vertical axis are given in Equation 8-12.

$$LCos\phi = W \qquad \textbf{(8-12)}$$

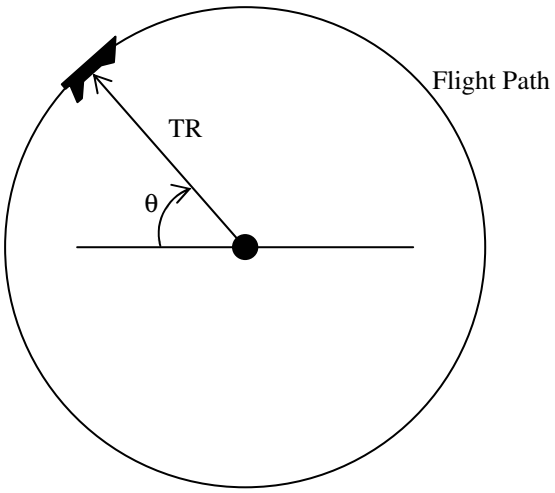
Therefore, from Equation 8-12 the load-factor (G's=L/W) may be given as in Equation 8-13.

$$G's = L/W = 1/Cos\phi \qquad \textbf{(8-13)}$$

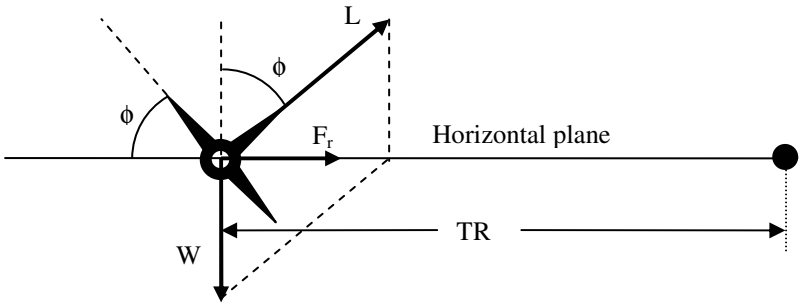
From Equation 8-13 the bank-angle may be expressed in terms of load-factor as given in Equation 8-14.

$$\phi = Cos^{-1}\left(1/G's\right) \qquad \textbf{(8-14)}$$

Therefore, to find the *TR* and the *ROT* of the aircraft we need to know the velocity of the aircraft (*V*) and either the load-factor (*G's*) or the bank-angle (ϕ).



(a) Top view of horizontal plane



(b) Front view of horizontal plane

Figure 8-4 : An aircraft in level turn

8.5.1 Effects of Turn-radius and Rate-of-turn on Aircraft Performance

The performance of an aircraft depends upon many factors, of which the turn-radius (*TR*) and the rate-of-turn (*ROT*) are two important characteristics. Considerations of the *TR* and *ROT* are particularly important for military aircraft and missiles. Keeping everything else constant, the aircraft or missile with the smallest *TR* and largest *ROT* will have an advantage in air combat. From Equation 8-10 and 8-11, to obtain the smallest *TR* and largest *ROT* the aircraft must operate with the highest possible load factor (*G's*) and the lowest possible velocity (*V*). High performance fighter aircraft are designed to operate at high G's (typically 3 to 10) [AND00]. In Equation 8-10 and 8-11, for higher values of *G's*, the term $\sqrt{G's^2 - 1}$ may be approximated to only "*G's*",

thus these equations may be rewritten as given in Equation 8-15 and 8-16 respectively [AND00].

$$TR \approx \frac{V^2}{g \cdot G's} \qquad \textbf{(8-15)}$$

$$ROT \approx \frac{g \cdot G's}{V} \qquad \textbf{(8-16)}$$

The combined effect of *V* and *G's* on *ROT* and *TR* is quite complex. To see the individual effects, the *ROT* and *TR* are calculated for different values of *V* and *G's* as shown in Table 8-2. When velocity is kept constant, the *TR* is inversely proportional to G's ($TR \propto 1/G's$) and the *ROT* is directly proportional to G's ($ROT \propto G's$). On the other hand, when G's is kept constant, the *TR* is directly proportional to the square of velocity ($TR \propto V^2$) and the *ROT* is inversely proportional to velocity ($ROT \propto 1/V$).

Table 8-2 : Rate-of-turn and Turn-radius Calculated Values

Mach #	Gs	ROT (deg/sec)	TR (meters)
1	2	2.86	6823
1	9	14.77	1321
1	20	32.98	592
1	40	66.03	296
2	2	1.43	27292
2	9	7.38	5285
2	20	16.49	2366
2	40	33.01	1182
3	2	0.95	61406
3	9	4.92	11891
3	20	10.99	5325
3	40	22.01	2660
4	2	0.71	109166
4	9	3.69	21140
4	20	8.25	9466
4	40	16.51	4729
6	2	0.48	245624
6	9	2.46	47565
6	20	5.50	21298
6	40	11.00	10639

The velocity has a much greater effect on the *TR* than the load-factor. When *G's* is constant and velocity is doubled the *TR* increases four times and *ROT* is reduced to half. When *G's* are kept constant and velocity is reduced to a half, the *TR* is a quarter and the *ROT* doubles. Whereas, when *V* is constant and *G's* doubled the *TR* is a half and the *ROT* doubles. Table 8-3 summarizes the effects of *G's* and *V* on *TR* and *ROT*. The fourth case shown in Table 8-3 shows the effects of changing *V* and *G's* both at the same time. By doubling the *G's* and reducing the *V* to half, the *TR* reduced to $1/8^{th}$ and the *ROT* increased by four times.

Table 8-3 : Summary of ROT and TR for different cases

Case	I	II	III	IV
G's	Constant	Constant	Double	Double
V	Double	Half	Constant	Half
TR	4 times	Quarter	Half	$1/8^{th}$
ROT	Half	Double	Double	4 times

Although having more G's force is better for the missiles performance, the low speed is dependent upon the type of missile and the speed of the target being chased. It may be that in a head-on scenario, a missile with less speed may hit a target aircraft flying faster than missile, but for most of the situations, the missile has to be faster than the target. Typically, for chasing an airborne target, the missile speed may be double that of the target and the missile G's force may be four times more than that of the target being chased.

8.6 Target and Missile Increment in one Time Frame

To augment the position of the target during the simulation, the distance the target and the missile can travel in one time interval (Δt) or between two frames is to be calculated. The target-step ($step_{tgt}$) may be calculated from the target velocity (V_{tgt}) and the time interval (Δt) which is equal to 1/ ref_rate.

$$step_{tgt} = V_{tgt} \cdot \Delta t \qquad \textbf{(8-17)}$$

Similarly, the missile step ($step_{msl}$) may be calculated from the missile velocity (V_{msl}) and the Δt as given in Equation 8-18.

$$step_{msl} = V_{msl} \cdot \Delta t \qquad \textbf{(8-18)}$$

8.7 Rate-of-Climb or Rate-of-Descent

In Figure 8-5 (a), the aircraft is flying along the horizontal axis and the thrust (T) is equal to the drag (D). This is the case of a straight and level flight. But if the thrust is greater than the drag, the aircraft may move up with a speed which is the vertical component of the aircraft velocity (V) due to the bank-angle (ϕ) as shown in Figure 8-5(b). The $VSin(\phi)$ gives the rate with which the aircraft is climbing up or may be called the rate-of-climb (ROC) [AND00].

$$ROC = V \sin\phi \qquad \textbf{(8-19)}$$

In Figure 8-5 (b) the forces applying along flight path may be written as.

$$T = D + W \sin\phi \qquad \textbf{(8-20)}$$

Multiplying both sides with velocity (V) gives us.

$$V \cdot T = V \cdot D + W \cdot V \sin\phi \qquad \textbf{(8-21)}$$

Rearranging the terms for $Vsin(\phi)$.

$$V \sin\phi = \frac{VT - VD}{W} = \frac{excess.power}{W} \qquad \textbf{(8-22)}$$

Where "*VT-VD*" is called the "*excess power*" [AND00]. Comparing Equation 8-19 and Equation 8-22 the *ROC* may be expressed in terms of the "*excess.power*" and the weight of the aircraft.

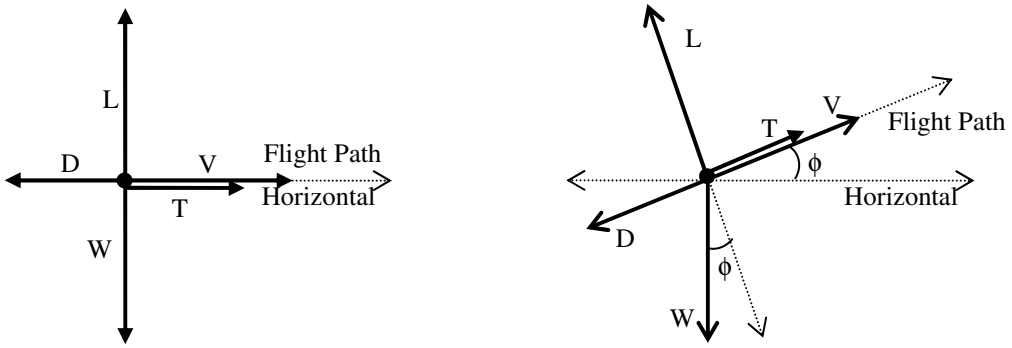
$$ROC = \frac{excess.power}{W} \qquad \textbf{(8-23)}$$

Typically, the *ROC* is expressed in thousands of meters per minute. The rate-of-descent (*ROD*) may be defined as the lost in altitude of the flying aircraft per unit time. Generally, *ROD* is also given as thousands of meters per minute. To use the *ROD* in the simulation for changing the altitude of the target aircraft, the *ROD* per time interval (Δt) in seconds needs to be calculated. This may be calculated as given in Equation 8-24.

$$ROD_{\text{int}} = ROD \cdot \Delta t / 60 \qquad \textbf{(8-24)}$$

Where, ROD_{int} is the *ROD* per time interval in seconds

ROD is the input *ROD* in meters per minute



(a) Steady level flight path

(b) Climb flight path

Figure 8-5 : Rate-of-climb explanation

8.8 Target Manoeuvrability

For simulating the missile-target engagement scenarios, the target aircraft is initially positioned at some desired location and aspect and then the aircraft performs some pre-defined manoeuvres. To cover typical manoeuvres which the target aircraft may perform in normal flight or to avoid a missile hit, the following modes of target manoeuvres are modelled by controlling the "*translation*" and "*rotation*" fields of the aircraft in the virtual world.

8.8.1 Straight-and-Level Modes

The aircraft may move straight-and-level in the following four directions.

(a) Straight and level head-on towards the missile,
(b) Straight and level tail-chase away from the missile,
(c) Straight and level tangential left to right,
(d) Straight and level tangential right to left.

Figure 8-6 illustrates a few possible scenarios for the target straight-and-level flight. By changing the relative position of the target and the missile any desired mode may be generated.

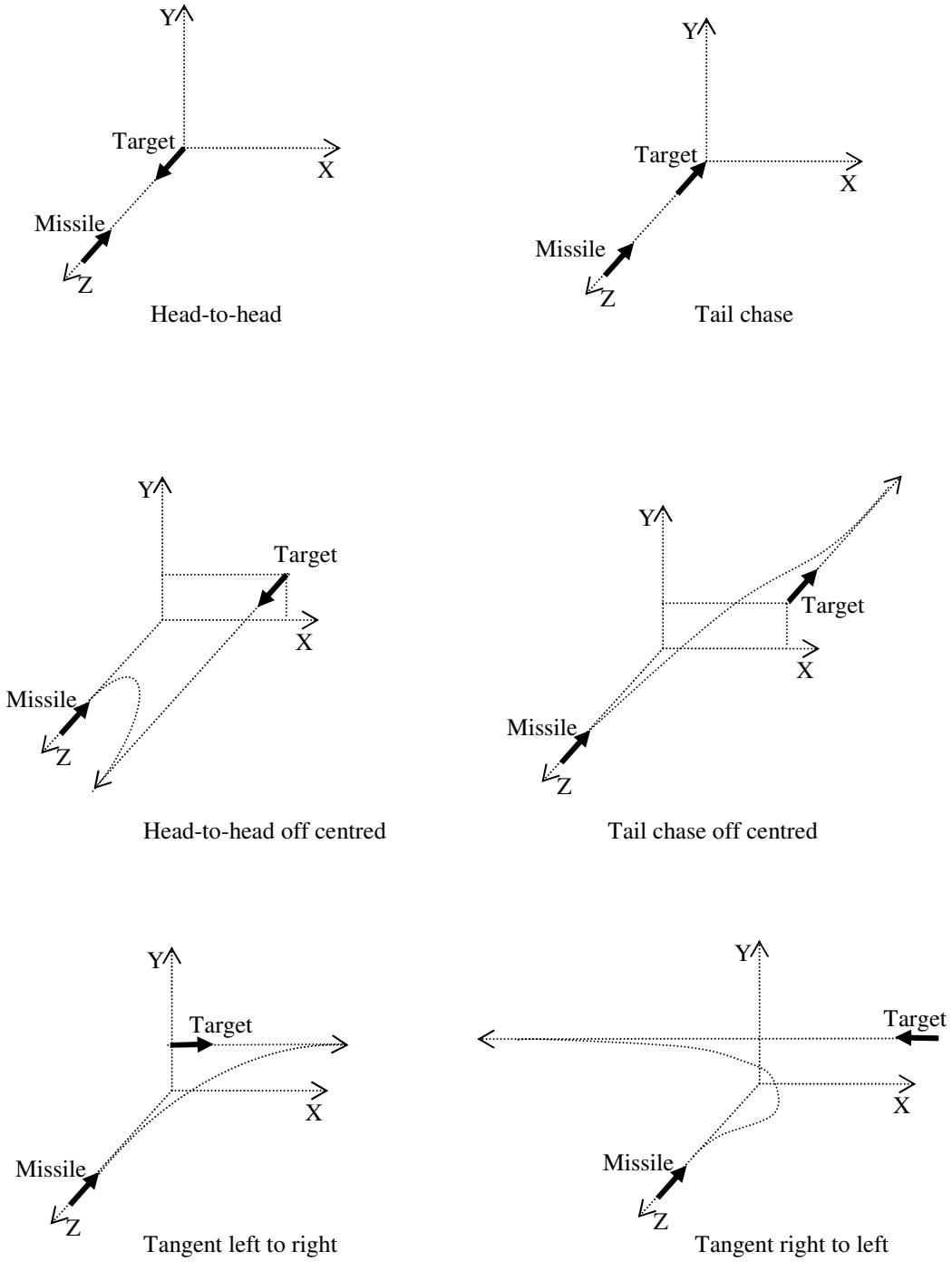


Figure 8-6 : Target Manoeuvres straight-and-level modes

8.8.2 Level-Turn Modes

The target aircraft may perform a level-turn in the following directions.

(a) Tangential right turning towards,
(b) Tangential right turning away,
(c) Tangential left turning towards,
(d) Tangential left turning away,
(e) Longitudinal right turning towards,
(f) Longitudinal right turning away,
(g) Longitudinal left turning towards,
(h) Longitudinal left turning away.

The directions are defined as looking from the missile position. Figure 8-7 illustrates several cases of level-turn.

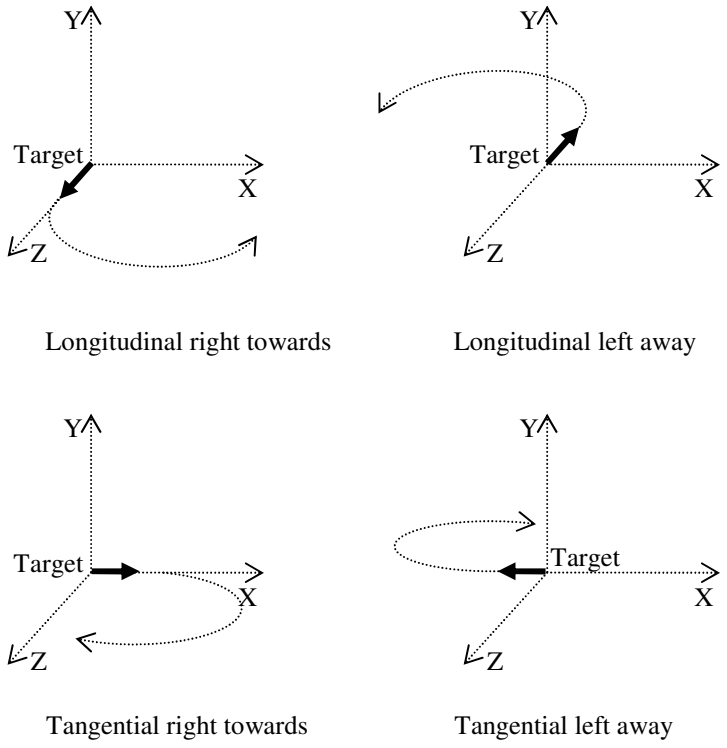


Figure 8-7 : Target manoeuvres level turn modes

8.8.3 Take-off and Landing Modes

The same cases of straight-and-level flight mentioned in paragraph 8-8-1 above may be converted into takeoff or landing by incrementing the aircraft altitude Y_{tgt} by ROD_{int} calculated in Equation 8-24. Whereas, for the straight and level flight modes the Y_{tgt} value is kept as constant. Figure 8-8 illustrates the target take-off and landing modes.

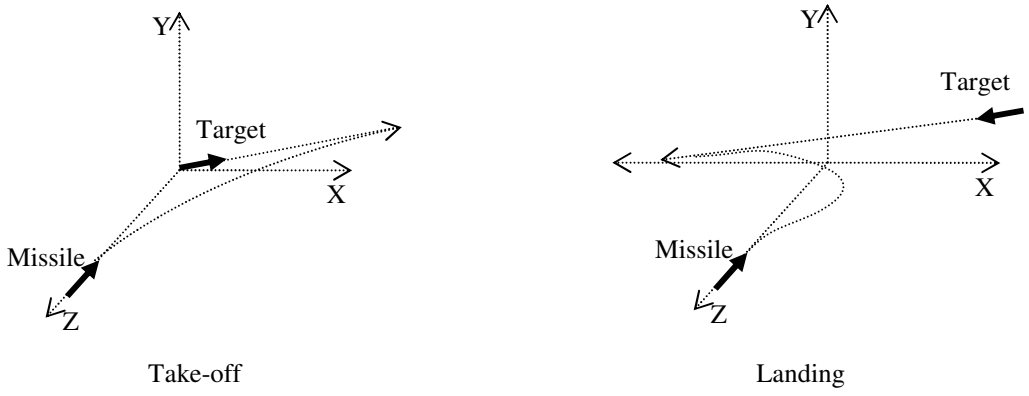


Figure 8-8 : Target take-off and landing modes

8.8.4 Spiral Descent Mode

The level turn cases discussed in paragraph 8-8-2 may be altered to a spiral descent by changing the altitude of the target aircraft during turn. The ROD_{int} calculated in Equation 8-24 may be used to increment the Y_{tgt} accordingly. Figure 8-9 shows one example of a spiral descent mode.

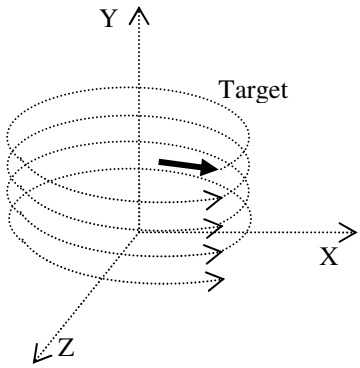


Figure 8-9 : Target manoeuvre spiral descent mode

8.8.5 Complex Manoeuvres

Small portions of the target flight such as the straight-and-level, level-turn, spiral descent and landing etc. may be joint together to model complex manoeuvres. For example the target aircraft coming straight-and-level may then make a turn and descend down and then land in a straight line. Any complex manoeuvre may be planned in advance before the simulation starts and may be implemented by selecting the modes at different time intervals. Presently, the complex manoeuvres are not incorporated in the algorithm. However, this may be added by making changes in the algorithm. Alternatively, B-spline curves may be used for generating complex paths for target manoeuvres [BUS03]. MATLAB Splines Toolbox deals with B-splines and may be used in future work for generating complex manoeuvres [MAT07].

8.8.6 Modelling Level Flight Modes

The changes required to be made in the "*translation*" and "*rotation*" field values for different cases of straight-and-level modes are explained and the summary of these is given in Table 8-4. The target translation field values given in Equation 8-1 may be altered to find the target new position for the straight-and-level modes. The $step_{tgt}$ calculated in Equation 8-17 is used to increment the target position in the corresponding direction. The translation fields for all four cases of straight flight are shown in Table 8-4. For the descent or take-off cases, the ROD_{int} calculated in Equation 8-24 is added to Y_{tgt}.

The initial direction of the target aircraft (θ_{tgt_ini}) is fed in as the angle in the rotation field of Equation 8-2. The θ_{tgt_ini} values for the different modes of straight-and-level are given in Table 8-4. In Equation 8-2, the values for the direction cosines are set as (0, 1, 0) which corresponds to the "*Yaw*" movement of the aircraft. The default direction of the rotation field is along the negative Z-axis. That means if θ_{tgt_int} is zero, the target aircraft heading is in the negative Z direction. Any positive value of θ_{tgt_int} corresponds to rotation in a counter-clock wise direction.

Table 8-4 : Target manoeuvrability different modes

Mode		Initial Direction	Initial Rotation (0 1 0 θ_{tgt_int})	Translation Field
Longitudinal	Head-on	Looking in +Z	(0 1 0 180^o)	(X_{tgt}, Y_{tgt} -ROD_{int}, Z_{tgt} + $step_{tgt}$)
	Tail-chase	Looking in -Z	(0 1 0 0^o)	(X_{tgt}, Y_{tgt} -ROD_{int}, Z_{tgt} - $step_{tgt}$)
Tangential	Right-to-left	Looking in -X	(0 1 0 90^o)	(X_{tgt}- $step_{tgt}$, Y_{tgt}-ROD_{int}, Z_{tgt})
	Left-to-right	Looking in +X	(0 1 0 -90^o)	(X_{tgt}+ $step_{tgt}$, Y_{tgt}-ROD_{int}, Z_{tgt})

8.8.7 Modelling Turn Modes

For the cases in which the aircraft is turning along a curved path, the location of the target at each time interval is calculated by converting the target *TR* and the ROT_{int} information into Cartesian coordinates. This is done by using the MATLAB function "*pol2cart*" which converts the point in polar coordinates into the corresponding Cartesian coordinates [MAT07].

$$\lfloor X_{tgt} \quad Z_{tgt} \rfloor = pol2cart(rot, TR) \qquad \textbf{(8-25)}$$

Where, *TR* is the turn-radius calculated in Equation 8-10 and *"rot"* is the angle the target is making from its initial direction θ_{tgt_int}. Depending upon the mode selected the "*rot*" may be incremented or decremented by ROT_{int} in each time-interval. The *rot* calculated for different modes are given in Table 8-5.

$$rot = rot \pm ROT_{\text{int}} \qquad \textbf{(8-26)}$$

Where, ROT_{int} represent the target angular displacement in one time interval. The ROT_{int} is calculated by multiplying the target aircraft *ROT* calculated in Equation 8-11 with the time interval (ROT_{int} =*ROT* x Δt). To turn the target from its initial position, the target location is shifted by *TR* along the Z-axis for tangential modes as shown in Table 8-5.

$$Z_{tgt} = Z_{tgt} \pm TR \qquad \textbf{(8-27)}$$

Similarly, for longitudinal modes shown in Table 8-5, the target location is moved by *TR* on X-axis.

$$X_{tgt} = X_{tgt} \pm TR \qquad \textbf{(8-28)}$$

For the cases of descent or climb, the altitude or the vertical component of the target position (Y_{tgt}) is changed accordingly. An increment (Y_{inc}) is calculated at each time interval as shown in Equation 8-29 and is subtracted from the target start altitude (Y_{tgt_start}). The value of ROD_{int} is calculated in Equation 8-24. However, in the case of level flight Y_{inc} remains zero.

$$Y_{inc} = Y_{inc} + ROD_{\text{int}} \qquad \textbf{(8-29)}$$

The target rotation angle (ϕ_{tgt}) as given in Equation 8-2 is calculated at each time interval by multiplying the target ROT_{int} with the number of frames (n). The direction of rotation depends upon the mode selected. Table 8-5 summarizes ϕ_{tgt} values for all modes.

$$\phi_{tgt} = \pm ROT_{\text{int}} \times n \qquad \textbf{(8-30)}$$

Table 8-5 : Aircraft translation and rotation fields for turned flight modes

Mode		Initial Direction	Rotation Field (Initial)	Rotation Field ($0\ 1\ 0\ \phi_{tgt}$)	rot (initial)	rot	Translation Field
Tangential	Right Towards	+X	($0\ 1\ 0\ -90^o$)	$-\ n \times ROT_{int}$	$-\pi/2$	$rot + ROT_{int}$	$X_{tgt\_start}+ X_{tgt}$, $Y_{tgt\_start}-Y_{inc}$, $Z_{tgt\_start}+Z_{tgt}+TR$
	Left Towards	- X	($0\ 1\ 0\ \ 90^o$)	$+ n \times ROT_{int}$	$-\pi/2$	$rot - ROT_{int}$	$X_{tgt\_start}+ X_{tgt}$, $Y_{tgt\_start}-Y_{inc}$, $Z_{tgt\_start}+Z_{tgt}+TR$
	Right Away	+X	($0\ 1\ 0\ -90^o$)	$+ n \times ROT_{int}$	$\pi/2$	$rot - ROT_{int}$	$X_{tgt\_start}+ X_{tgt}$, $Y_{tgt\_start}-Y_{inc}$, $Z_{tgt\_start}+Z_{tgt}-TR$
	Left Away	- X	($0\ 1\ 0\ \ 90^o$)	$-\ n \times ROT_{int}$	$\pi/2$	$rot + ROT_{int}$	$X_{tgt\_start}+ X_{tgt}$, $Y_{tgt\_start}-Y_{inc}$, $Z_{tgt\_start}+Z_{tgt}-TR$
Longitudinal	Right Towards	+Z	($0\ 1\ 0\ 180^o$)	$+ n \times ROT_{int}$	π	$rot - ROT_{int}$	$X_{tgt\_start}+ X_{tgt}+TR$, $Y_{tgt\_start}-Y_{inc}$, $Z_{tgt\_start}+Z_{tgt}$
	Left Towards	+Z	($0\ 1\ 0\ 180^o$)	$-\ n \times ROT_{int}$	0	$rot + ROT_{int}$	$X_{tgt\_start}+ X_{tgt}-TR$, $Y_{tgt\_start}-Y_{inc}$, $Z_{tgt\_start}+Z_{tgt}$
	Right Away	- Z	($0\ 1\ 0\ \ \ 0^o$)	$-\ n \times ROT_{int}$	π	$rot + ROT_{int}$	$X_{tgt\_start}+ X_{tgt}+TR$, $Y_{tgt\_start}-Y_{inc}$, $Z_{tgt\_start}+Z_{tgt}$
	Left Away	- Z	($0\ 1\ 0\ \ \ 0^o$)	$+ n \times ROT_{int}$	0	$rot - ROT_{int}$	$X_{tgt\_start}+ X_{tgt}-TR$, $Y_{tgt\_start}-Y_{inc}$, $Z_{tgt\_start}+Z_{tgt}$

8.9 Cartesian Coordinates and Axis Angle

In VRML, the rotation vector is defined using *axis-angle* [VRML97]. The rotation vector field holds four values (*X, Y, Z,*ϕ). The first three values specify a normalized rotation axis vector about which the rotation takes place. The fourth value specifies the amount of right-handed rotation about that axis in radians. Figure 8-10(a) illustrates the unit vector in Cartesian coordinates. The three axes are *X*, *Y* and *Z* and the corresponding angles which the unit vector OP makes with these axes are α, β and γ respectively. The lengths *a*, *b* and *c* are the corresponding projections of the unit vector along *X*-, *Y*- and *Z*-axis respectively.

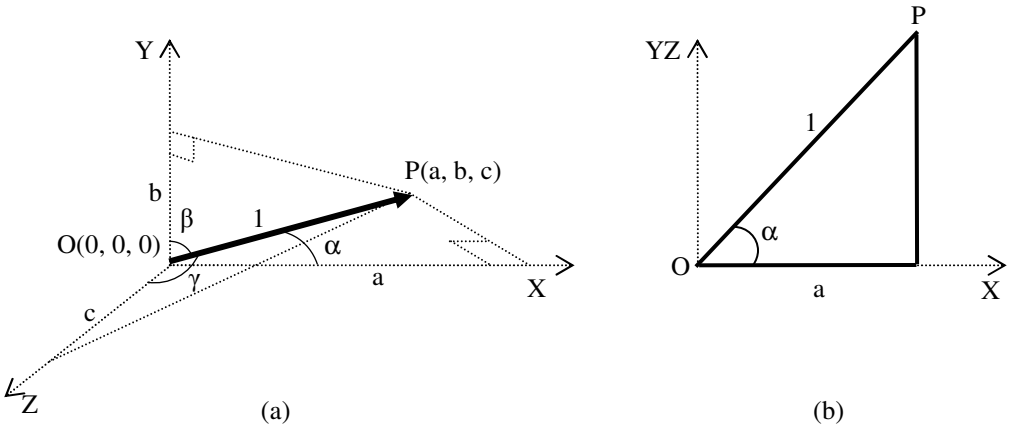


Figure 8-10 : Direction Cosines of a Unit vector

The projection of point *P* on the three axes makes three right angle triangles. From the triangle along *X*-axis as illustrated in Figure 8-10(b), the $Cos\alpha = a/1$ or $a = Cos\alpha$. Similarly, from the other two triangles $b = Cos\beta$ and $c = Cos\gamma$. These projections of unit vector are called "direction cosines" [PAT68]. Whereas, the magnitude of the unit vector is given in Equation 8-31.

$$1 = \sqrt{a^2 + b^2 + c^2} \qquad \textbf{(8-31)}$$

The projection of the same point *P* on the *XY*-plane is illustrated in Figure 8-11(a). The magnitude of the projection of the point *P* on the *XY*-plane is given as Equation 8-32.

$$XY_{projection} = \sqrt{a^2 + b^2} \quad \textbf{(8-32)}$$

The angle (ϕ) is the azimuth rotation angle which the unit vector is making with the *XY*-plane as shown in Figure 8-11(c) is given in Equation 8-33.

$$\phi = \tan^{-1}\left(\frac{c}{\sqrt{a^2 + b^2}}\right) \quad \textbf{(8-33)}$$

The angle (θ) is the elevation rotation angle which the projection of the point *P* on the *XY*-plane (*OP'*) is making with the X-axis as shown in Figure 8-11(b) and is given in Equation 8-34.

$$\theta = \tan^{-1}\left(\frac{b}{a}\right) \quad \textbf{(8-34)}$$

These two angles ϕ and θ are used for finding the rotation field values in VRML. In VRML, the angles are defined using the right-hand rule which means any object rotating along the counter-clockwise direction is considered as a positive rotation angle [VIR04].

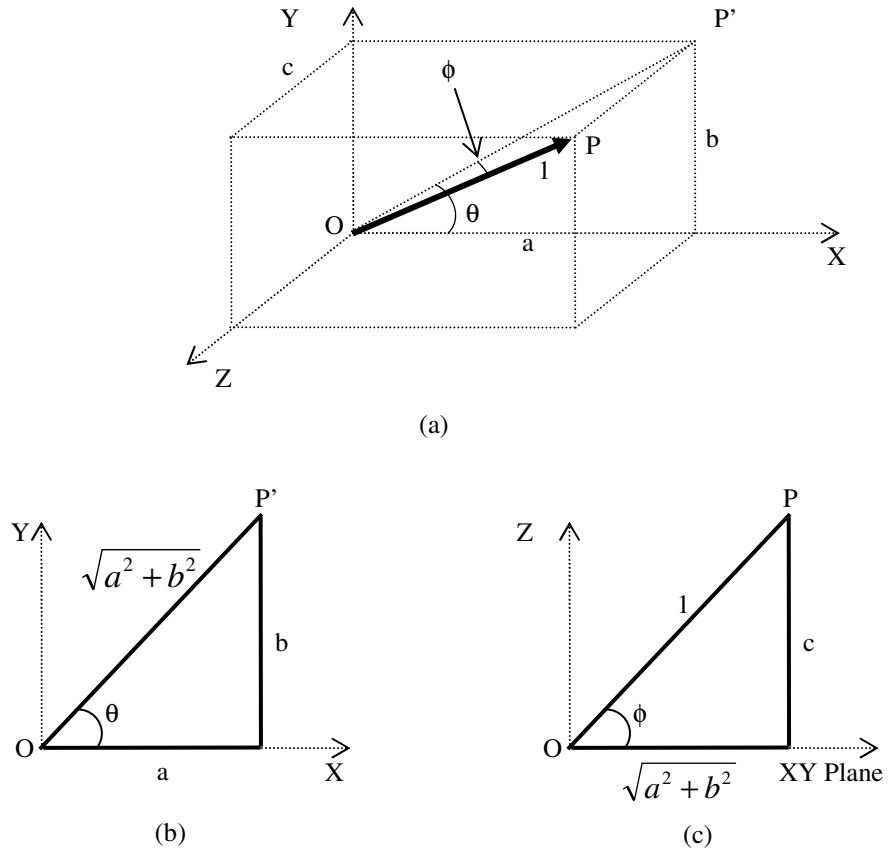


Figure 8-11 : Cartesian coordinates and direction cosines

8.10 Missile Guidance and Control Modelling

The missile movement at any one instance depends upon the missile and target relative position and direction. The missile gimballed seeker head tracks the target aircraft. Remaining within the aerodynamic limits, the missile steers towards the target. The movement of the missile depends upon the speed (V_{msl}) and the load factor ($G's_{msl}$) which in turn decides the *ROT*, *TR* and lateral acceleration capabilities. The following paragraphs explain the algorithm developed for the tracking and guidance of the missile in three dimensions. The missile movement in the 3D virtual world is controlled by changing the "*translation*" and "*rotation*" fields given in Equation 8-3 to 8-5.

8.10.1 Missile Gimbal Initial Direction Modelling

The gimballed seeker head can rotate independent of the missile body rotation or missile_LOS. This implies that the seeker head can look in any direction (within a cone in front of the missile) independent of the missile_LOS. The missile gimbal

rotation field is given in Equation 8-5. At the start of the simulation, irrespective of the missile_LOS, the seeker head must always look towards the target. Figure 8-12 shows the 3D view of the target and missile initial positions.

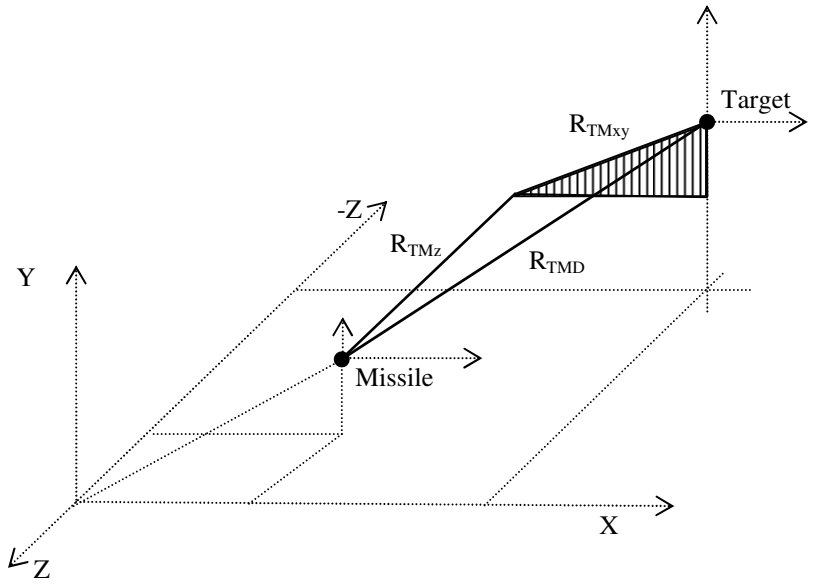


Figure 8-12 : Target and missile relative positions in 3D view

To understand the calculations involved in finding the gimbal direction, the two triangles in the 3D view shown in Figure 8-12 are separately redrawn in 2D in Figure 8-13(a) and (b). From Figure 8-13(a) the angle (θ) is calculated as in Equation 8-35.

$$\theta = \tan^{-1}\left(\frac{Y_{tgt} - Y_{msl}}{-(X_{tgt} - X_{msl})}\right) \qquad \textbf{(8-35)}$$

The negative sign in the denominator is to change the θ polarity in accordance with the VRML quadrant sign requirements. The differences in VRML and MATLAB sign conversions are discussed in paragraph 8-10-6. Therefore, if $(X_{tgt} - X_{msl}) > 0$ the θ calculated will be a negative value and needs to add π to make it positive. (i.e. $\theta = \theta + \pi$). The three direction cosines given in the gimbal rotation field of Equation 8-5 may be calculated as given in Equation 8-36 to 38.

$$X_{m\_gimbal} = \sin\theta \quad \textbf{(8-36)}$$

$$Y_{m\_gimbal} = \cos\theta \quad \textbf{(8-37)}$$

$$Z_{m\_gimbal} = 0 \quad \textbf{(8-38)}$$

The three direction cosines given in Equation 8-36 to 8-38 must satisfy the condition of unity magnitude given in Equation 8-31. From Figure 8-13(b), the gimbal angle ϕ_{gimbal} is calculated as in Equation 8-39.

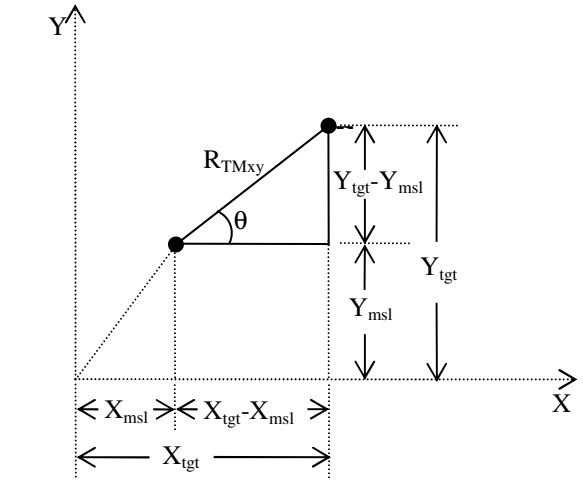
$$\phi_{gimbal} = \tan^{-1}\left(R_{TMxy} \Big/ R_{TMz} \right) \quad \textbf{(8-39)}$$

Where, R_{TMz} is the distance between the target and the missile along the Z-axis.

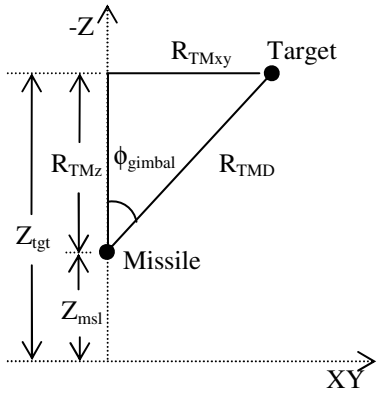
$$R_{TMz} = Z_{msl} - Z_{tgt} \quad \textbf{(8-40)}$$

and R_{TMxy} is calculated using Figure 8-13(a) as in Equation 8-41.

$$R_{TMxy} = \sqrt{\left(X_{tgt} - X_{msl}\right)^2 + \left(Y_{tgt} - Y_{msl}\right)^2} \quad \textbf{(8-41)}$$



(a)



(b)

Figure 8-13 : 2D views of target and missile relative positions

The distance R_{TMxy}, will always be a positive number as this is a square root of a sum of squares. However, R_{TMz} may become negative once the target crosses over the missile Z-axis (see Figure 8-13(b)). This negative value needs to be taken care of when calculating the angle ϕ_{gimbal} of the missile rotation field. Also at the instance the target crosses the missile on the Z-axis the R_{TMz} may become zero, this will make the output of Equation 8-39 undefined. To avoid this, when the R_{TMz} value becomes zero this is replaced with a very small value "*eps*" of MATLAB [MAT07].

The direct distance from the target to the missile (R_{TMD}) is calculated from the triangle as shown in Figure 8-13(b).

$$R_{TMD} = \sqrt{R_{TMxy}^2 + R_{TMz}^2} \qquad \textbf{(8-42)}$$

For the missile-target engagement simulation, the missile rotation field and distance R_{TMD} and R_{TMz} are calculated after every Δt (or for each frame) as the target and missile relative positions are changing constantly.

8.10.2 Capturing Frame from 3D VR Viewer

For the missile tracking and guidance algorithm, after every Δt seconds, the missile seeker needs to capture the 2D image from the 3D VR world viewer. To capture the 2D image, an algorithm is developed using several functions of MATLAB Virtual Reality toolbox. Table 8-6 explains the VR Toolbox functions used in this algorithm. The VRML file containing the 3D VR world is opened in MATLAB and a VR world object is associated with virtual world. The VR world object is given a name "*myworld*". Several VR viewer windows are opened to view the virtual world "*myworld*" from different aspects. The 2D image is captured in true RGB colours and stored in the variable "X_{RGB}".

Table 8-6 : VR Toolbox functions used for capturing 2D image

VR Toolbox function	Explanation
myworld = vrworld('file_name.wrl')	Creating a new world object associated with virtual world under name "*myworld*"
Open(myworld)	Opening virtual world object in MATLAB. The VRML fields can be controlled from MATLAB command
View(myworld)	Opening virtual world in a default VRML viewer. Virtual world can be viewed from different aspects by opening same virtual world in different windows
f = vrfigure(myworld)	Creating new VR figure and returning a VR figure object as variable "*f*"
Vrdrawnow	Updates virtual world field values. The view is updated when MATLAB is idle for some time.
X_{RGB} = capture(f)	Create 2D image from VR figure variable "*f*" and stores image as variable "*X*" in true RGB colours

8.10.2.1 Problem Faced in Capturing Image from Virtual World

During simulation, the virtual world fields' values need to be updated to augment the positions of the missile and the target. However, in MATLAB, the virtual world field values are updated when MATLAB is idle or no other function is running [MAT07]. For running the simulation, I am using the "*while*" function of MATLAB. This caused a problem in my algorithm as with the "*while*" loop running; the virtual world was not updating the field values correctly. To overcome this system limitation a pause of $1/30^{th}$ of a second is introduced in the "*while*" loop. This means that the maximum frame rate which can be achieved is 30 frames/sec. The maximum 30 frames/sec limit

is found by running the simulation several times with different "*pause*" values. For any value above 30 frames, the image was not refreshing correctly. The "*pause*" function of MATLAB is used to insert a time delay.

8.10.3 Converting RGB Image into Gray-scale Image and Binary Image

The 2D image captured from the 3D virtual world is in true RGB colours. Each pixel of the RGB image (X_{RGB}) holds the RGB colour values. For the "*intensity centroid*" tracker, each pixel of the image must represent its intensity over a gray-level. The X_{RGB} image is then converted into the intensity image with each pixel representing the intensity over the Gray-scale. The MATLAB function "*rgb2gray*" is used for this purpose [MAT07].

$$X_{gray} = rgb2gray\left(X_{RGB}\right) \quad \textbf{(8-43)}$$

To clip the image (X_{gray}) as per the missile detector criterion, a detector threshold ($threshold_{det}$) is used as the detection criterion for the missile seeker. There are different options for the detection criterion. The $threshold_{det}$ may be kept as an input parameter or be calculated separately from the detector *NEP* and the *SNR*. However, presently, the detection criterion is incorporated in the algorithm as a percentage level above minimum intensity present in the image ($threshold_{percent}$) as a user input and using this to calculate the $threshold_{det}$. Equation 8-44 calculates the detector threshold from the given threshold percentage.

$$threshold_{\det} = \left(1 + \frac{threshold_{percent}}{100}\right) X_{gray_{\min}} \quad \textbf{(8-44)}$$

Where, $threshold_{det}$ is the detector threshold level for the detection criterion

$threshold_{percent}$ is the percentage above the minimum intensity in the image

X_{gray_min} is the minimum intensity of any one pixel in the image. This may be found by $X_{gray_{\min}} = \min\left(\min\left(X_{gray}\right)\right)$

The image X_{gray} is tested for the detection criterion as given in Equation 8-45 and any pixel having intensity below $threshold_{det}$ calculated in Equation 8-44, is converted to a zero level by using a "*find*" function of MATLAB [MAT07].

$$X_{gray}\left(find\left(X_{gray} < threshold_{\det}\right)\right) = 0 \quad \textbf{(8-45)}$$

For the "*binary centroid*" tracker, each pixel of the image is represented as the binary value of "0" or "1" with zero as no target and one representing presence of the target irrespective of the intensities. The intensity image X_{gray} is converted into binary image data (X_{BW}) using the "*im2bw*" function of MATLAB. The "*im2bw*" uses a "*level*" between 0 and 1 to decide the go-no-go criterion [MAT07].

$$X_{BW} = im2bw\left(X_{gray}, level\right) \qquad \textbf{(8-46)}$$

This "*level*" value may be entered as any value between 0 and 1 to represent the detection criterion or the $threshold_{det}$ calculated in Equation 8-44 may be used as the "*level*". Another way of incorporating a threshold is to use the "*graythreshold*" function of MATLAB, which is based on Otsu's method of finding the threshold [MAT07]. The Otsu's method [OTS79] performs discriminant analysis to automatically select an optimal threshold to maximize the distribution of the resultant classes in gray levels. The method utilizes only the zeroth- and the first-order cumulative moments of the gray-level histograms which is normalized and regarded as a probability distribution. The "*graythreshold*" function given in Equation 8-47, generates a "*level*" between 0 and 1 which may be used in Equation 8-46 as a go-no-go criterion for converting the Gray-scale intensity image X_{gray} into binary image X_{BW}.

$$level = graythreshold\left(I_{gray}\right) \qquad \textbf{(8-47)}$$

8.10.4 Centroid Tracking as Target Location Estimation

Out of the different missile tracking techniques discussed in Chapter-3 paragraph 3-3, the most common and probably the best known of these is the centroid tracker [DUD93]. In the algorithm used in this simulation, the "*binary centroid*" tracker and the "intensity centroid" tracker are incorporated for target location estimation. The other methods discussed in Chapter-3 paragraph 3-3 may be added in the algorithm in future work.

In the binary centroid tracker, depending upon the threshold level, a binary image is generated which represents the presence of the target as pixels having level one and the rest all as level zero. Irrespective of their intensity the pixels having intensity more than the threshold are level one. Whereas, in the intensity centroid each pixel is given a different value on the Gray-scale corresponding to the intensity of each pixel. Their level depends upon their mass factor. To understand the centroid tracking algorithms, first we take an example of a binary image of a 10x10 pixel

matrix as illustrated in Figure 8-14. All the pixels other than the pixels shown as ones are having zero value. The centroid (or the centre of mass) may be found by summing the product of mass of each pixel and its distance from the origin. In Figure 8-14 the origin is considered to be at the left top corner of the image. The reason for considering origin at top left corner is that the image size in MATLAB is shown as numbers of rows and columns. The number of rows starts from the top and increase downward and the number of columns starts at the left and increases to the right [VIR04]. The X and Y components of the centroid may be found by using Equations 8-48 and 8-49 respectively.

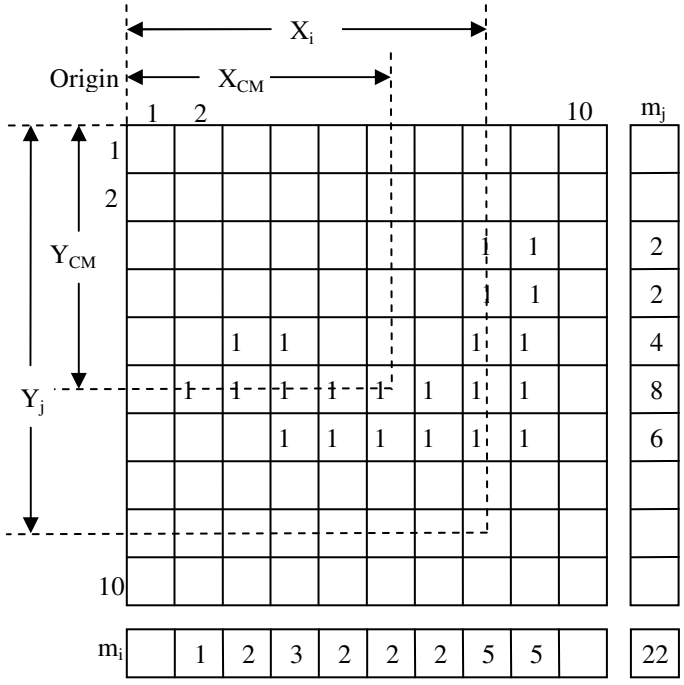


Figure 8-14 : Explaining binary centroid

$$X_{CM} = \frac{\sum_{i=1}^{q} m_i . X_i}{M} \qquad \textbf{(8-48)}$$

$$Y_{CM} = \frac{\sum_{j=1}^{p} m_j . Y_j}{M} \qquad \textbf{(8-49)}$$

where, $M = \sum_{j=1}^{p} \sum_{i=1}^{q} m_{ji}$ is the sum of all the pixels masses of the entire image

X_{CM} is the centre of mass on the X-axis in number of pixels,

Y_{CM} is the centre of mass on the Y-axis in number of pixels,

X_i is the distance (in number of pixels) of the ith element on the X-axis from the origin,

Y_j is the distance (in number of pixels) of the jth element on the Y-axis from the origin,

m_i is the sum of mass of all the pixels on the ith column,

m_j is the sum of mass of all the pixels on the jth row,

p is the number of rows of the image,

q is the number of columns of the image.

Therefore, the centroid of the complete image can be written as (X_{CM}, Y_{CM}). Although the binary and intensity centroid tracker uses the same Equations 8-48 and 8-49 and the only difference between these is the mass "m" of each pixel. In the case of binary centroid the mass of each pixel representing the target is considered as unity. Whereas, in the case of intensity centroid the intensity on the Gray-scale of each pixel is taken as the mass "m". In Figure 8-14 an example of a binary image is illustrated and Figure 8-15 illustrates the same image with the pixels intensities on the Gray-scale level.

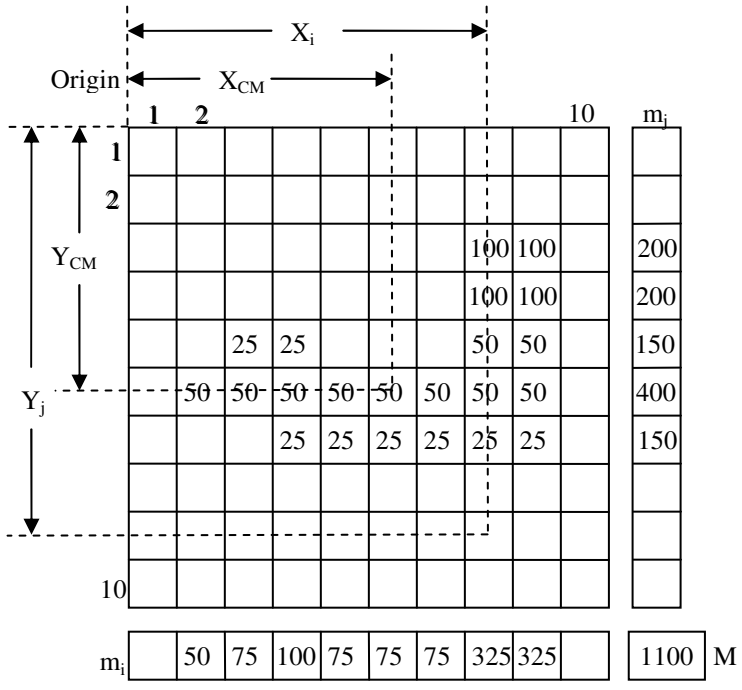


Figure 8-15 : Explaining intensity centroid

The MATLAB function "*regionprops*" may be used to find the centroid of an image [MAT07]. However, this function only uses the binary image (I_{BW}) for finding

the centroid thus it can only work for the binary centroid tracker. Initially, I used this function in my algorithm but to add the intensity centroid, I developed my own algorithm which finds centroids by using either the binary or the intensity centroid method. Figure 8-16 shows the flow diagram of the centroid tracker algorithm using the binary image X_{BW} of Equation 8-46 for the binary centroid and the intensity image X_{gray} of Equation 8-43 for the intensity centroid.

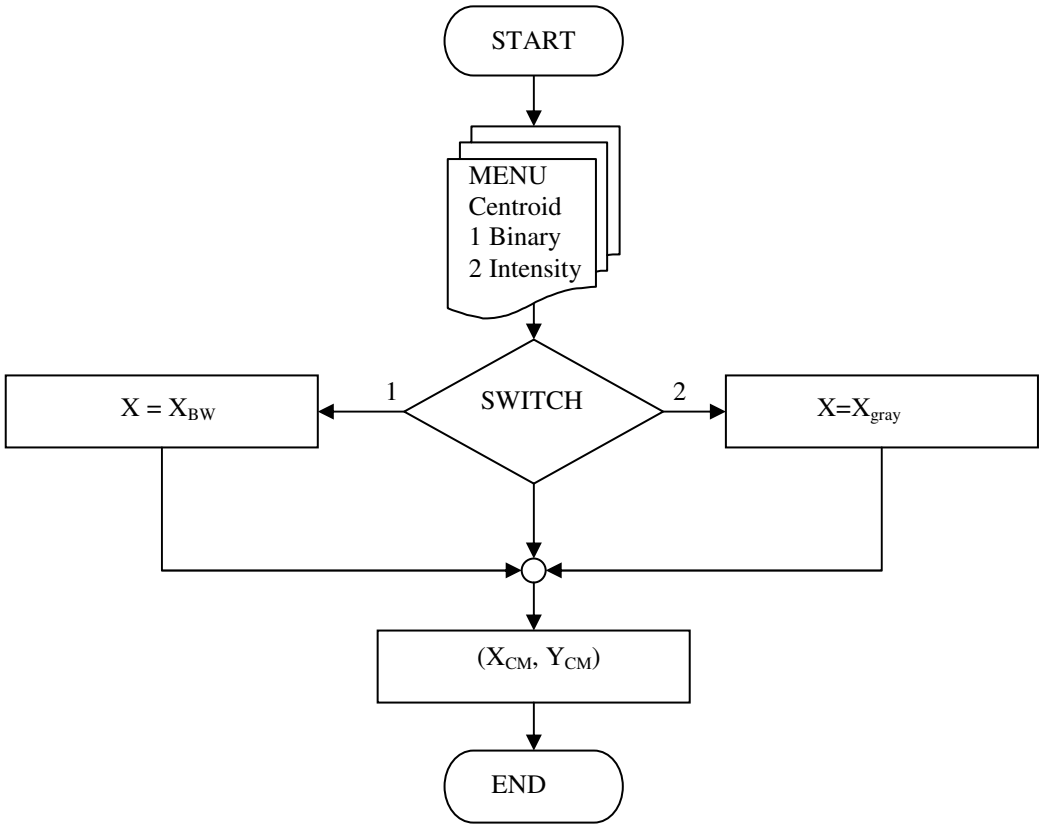


Figure 8-16 : Flow diagram of centroid tracking algorithm

8.10.5 Calculating Target Error from Bore-sight

The centroid (X_{CM}, Y_{CM}) shown in Figure 8-14 and Figure 8-15 are from the origin of the image which is at the top-left corner. To calculate the error vector from the bore-sight, the centroids are subtracted from the centre of the image (i.e. the bore-sight of the missile). Figure 8-17 illustrates the bore-sight and the error vector represented as X_{err} and Y_{err}. The values of the X- and Y-components of the error vector are calculated using Equation 8-50 and 8-51 respectively.

$$X_{err}^{pix} = col/2 - X_{CM} \qquad \textbf{(8-50)}$$

$$Y_{err}^{pix} = row/2 - Y_{CM} \quad \textbf{(8-51)}$$

Where, X_{err}^{pix} and Y_{err}^{pix} are the X- and Y-components of the error vector in number of pixels. The "*col*" is the number of columns in the image representing the number of pixels in the horizontal direction. The "*row*" is the number of rows in the image representing the number of pixels in the vertical direction. The "*col"* and "*row"* also correspond to the horizontal FOV and the vertical FOV respectively.

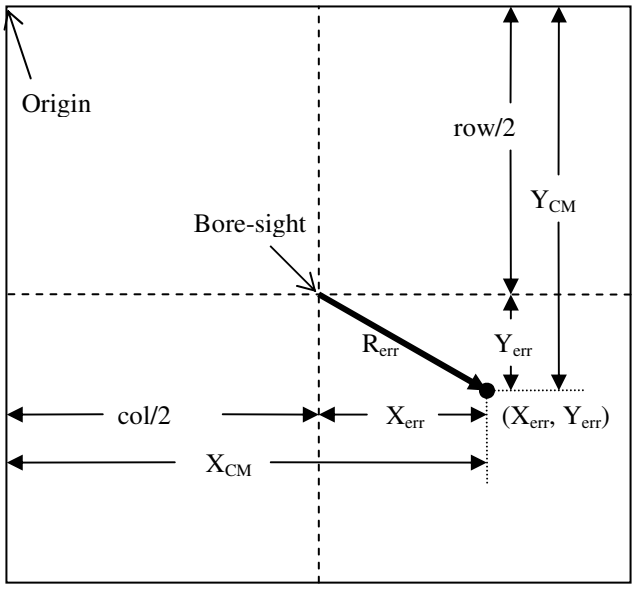


Figure 8-17 : Error signals from bore sight

For estimating the missiles new location, the error values in the pixels are to be converted into meters. This is done by converting number of pixels in the FOV into meters. The FOV is an angle which depends upon the distance between the target and the missile (R_{TMD}). In Figure 8-18, the height (h) of the image in meters at the distance R_{TMD} from the detector may be calculated using Equation 8-52.

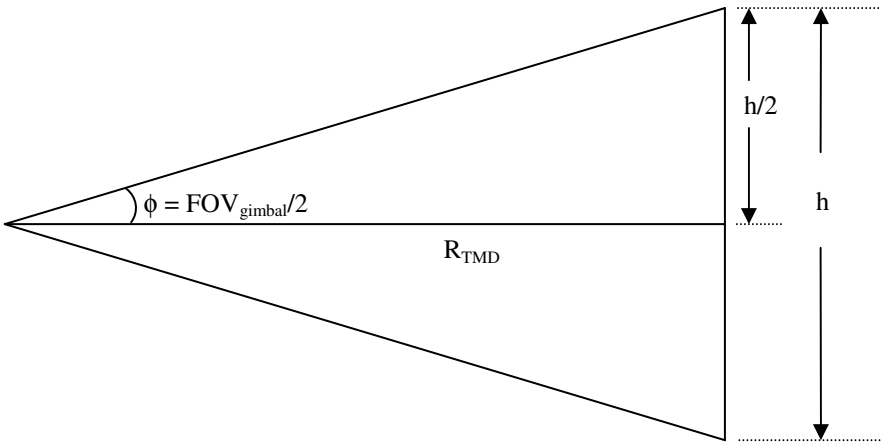


Figure 8-18 : Geometry explaining size of one pixel in meters

$$h = 2 \cdot R_{TMD} \cdot \tan\left(FOV_{gimbal} / 2\right) \quad \textbf{(8-52)}$$

In VRML, the FOV given in the "*navigationInfo*" node is the vertical FOV. The vertical FOV corresponds to the number of rows "*row*" and it is the Y-component of the error signal. Therefore, the size of one pixel (*pix_size*) in meters can be calculated as in Equation 8-53.

$$pix\_size = {}^{h}\!/_{row} \quad \textbf{(8-53)}$$

Where, the "*row*" is the number of rows of the image. For the square shape pixel, the same pixel size can also be used for converting the X-component of the error signal. Therefore, the error components in meters are calculated by multiplying the error components given in Equation 8-50 and 8-51 with *pix_size* given in Equation 8-53.

$$X_{err} = pix\_size \cdot X_{err}^{pix} \quad \textbf{(8-54)}$$

$$Y_{err} = pix\_size \cdot Y_{err}^{pix} \quad \textbf{(8-55)}$$

And the magnitude of the error vector (R_{err}) is given in Equation 8-56

$$R_{err} = \sqrt{X_{err}^2 + Y_{err}^2} \quad \textbf{(8-56)}$$

8.10.6 Sign Conversions in VRML and MATLAB

In VRML, the polarity standards in the four quadrants followed for the "*translation*" and "*rotation*" fields are different. Considering the four quadrants shown in Figure 8-19, the X-components of the "*translation*" fields of the missile (X_{msl}) and the target (X_{tgt}) are positive in the First and Fourth quadrants and the Y-components Y_{msl}

and Y_{tgt} are positive in the First and Second quadrants. Whereas, for the "*rotation*" field the X_m is positive in the First and Second quadrants and Y_m is positive in the Second and Third quadrants. The X_{err} is considered positive on the left of the Y-axis (i.e. Second and Third quadrants) and the Y_{err} is considered positive in the First and Second quadrants. As the MATLAB and VRML sign conversion are different [VIR04], therefore, for controlling the movement of objects in the 3D virtual world from MATLAB, the sign conversions, summarized in Figure 8-19, are finalized after repeatedly making changes in the algorithm and trying different options and finally deciding the logic for each field separately.

Second Quadrant X_{err} (+ve) Y_{err} (+ve) X_m (+ve) Y_m (+ve) X_{msl} (- ve) Y_{msl} (+ve) X_{tgt} (- ve) Y_{tgt} (+ve)	First Quadrant X_{err} (- ve) Y_{err} (+ve) X_m (+ve) Y_m (- ve) X_{msl} (+ve) Y_{msl} (+ve) X_{tgt} (+ve) Y_{tgt} (+ve)
Third Quadrant X_{err} (+ve) Y_{err} (- ve) X_m (- ve) Y_m (+ve) X_{msl} (- ve) Y_{msl} (- ve) X_{tgt} (- ve) Y_{tgt} (- ve)	Fourth Quadrant X_{err} (- ve) Y_{err} (- ve) X_m (- ve) Y_m (- ve) X_{msl} (+ve) Y_{msl} (- ve) X_{tgt} (+ve) Y_{tgt} (- ve)

Figure 8-19 : Sign for translation and rotation fields as per four quadrants

8.10.7 Updating Gimbal Rotation Field

At the start of the simulation, the missile gimbal direction is locked on towards the target. Paragraph 8-10-1 explains the steps involved in calculating the missile initial direction. During the simulation for tracking the target, the gimbal should remain looking towards the target. However, due to the movement of the missile and the target, an error is induced. This error is used in the missile control algorithm for steering the missile towards the target. The error components calculated in Equation 8-54 and 8-55 are used to calculate the total error at every frame using Equation 8-57 and 8-58 respectively.

$$X_{errT} = X_{err} + X_{errT} \qquad \textbf{(8-57)}$$

$$Y_{errT} = Y_{err} + Y_{errT} \quad \textbf{(8-58)}$$

In Figure 8-20, the total magnitude (R_{errT}) and the total direction (θ_T) of the error vector are calculated as given in Equation 8-59 and 8-60 respectively.

$$R_{errT} = \sqrt{X_{errT}^2 + Y_{errT}^2} \quad \textbf{(8-59)}$$

$$\theta_T = \tan^{-1}\left(Y_{err} \Big/ X_{err} \right) \quad \textbf{(8-60)}$$

If X_{errT} is less than zero (X_{errT} <0), the θ_T calculated in Equation 8-60 is negative and needs to be added with "π" to make this a positive angle. In Figure 8-20(a) X_{errT} is negative if it is in the First or Fourth quadrants.

$$\theta_T = \theta_T + \pi \quad \textbf{(8-61)}$$

To rotate the gimbal towards the target new position, the direction cosines for the missile gimbal rotation field are calculated as per the new position of the missile and the target by putting total angle θ_T in the Equation 8-36 to 8-38.

$$X_{m\_gimbal} = Sin\theta_T \quad \textbf{(8-62)}$$

$$Y_{m\_gimbal} = Cos\theta_T \quad \textbf{(8-63)}$$

$$Z_{m\_gimbal} = 0 \quad \textbf{(8-64)}$$

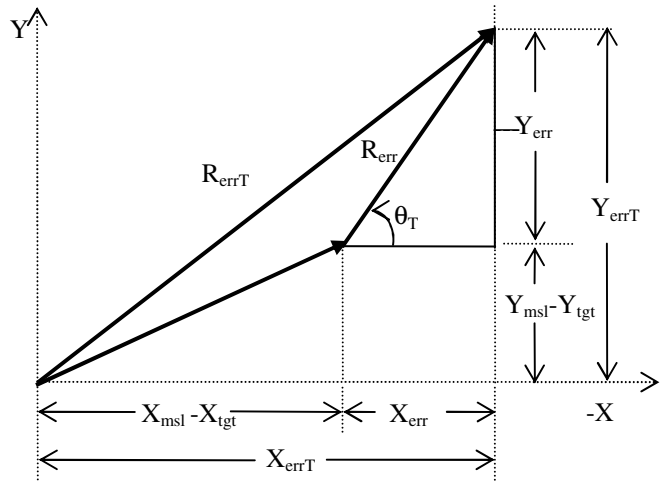
The total rotation angle (ϕ_{gimbal_T}) shown in Figure 8-20(b) is calculated as given in Equation 8-65.

$$\phi_{gimbla\_T} = \tan^{-1}\left(R_{errT} \Big/ R_{TMz} \right) \quad \textbf{(8-65)}$$

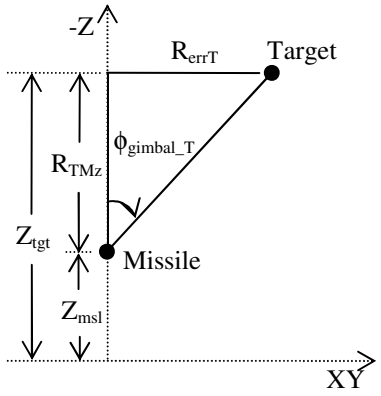
In Equation 8-65, if R_{TMz} is less than zero (R_{TMz} <0), then ϕ_{gimbal_T} is a negative angle and needs to be converted to a positive angle by adding π.

$$\phi_{gimbal\_T} = \phi_{gimbal\_T} + \pi \quad \textbf{(8-66)}$$

Finally the gimbal rotation field given in Equation 8-5 is updated as per the new direction cosines and the rotation angle calculated in Equation 8-62 to 8-64 and 8-66 respectively.



(a)



(b)

Figure 8-20 : Total error vector magnitude and direction

8.10.8 Gimballed Seeker Head Maximum Angle Limit

The gimballed seeker head is modelled to rotate in a spherical region around missile_LOS. The gimballed seeker head rotation is limited by an angle (θ_{gimbal_max}) which holds a value based upon the missile input parameters (typical value between 45^o to 90^o). This limit is incorporated in the algorithm using the logic explained in the flow diagram of Figure 8-21.

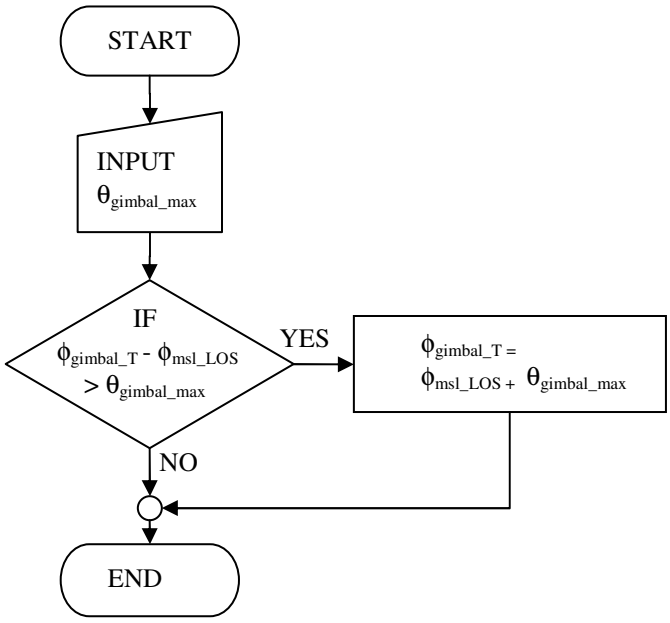


Figure 8-21 : Flow diagram explaining gambal maximum angle limit

8.10.9 Updating Missile Translation Fields

During the simulation, the missile new position needs to be calculated for each frame. The new position of the missile depends upon the target new location and the missile LATAX limit. Considering the missile target locations given in Figure 8-12, and using the θ and ϕ_{gimbal} calculated in Equation 8-35 and 8-39, the angle ζ_T and angle $\phi(n)$ are written as Equation 8-67 and 8-68 respectively. The angle θ of Equation 8-35 is given a new name ζ_T, just to differentiate it from the initial value of θ as during the simulation ζ_T is used for the LATAX implementation.

$$\zeta_T = \tan^{-1}\left((Y_{tgt} - Y_{msl}) \Big/ -\left(X_{tgt} - X_{msl}\right) \right) \qquad \textbf{(8-67)}$$

$$\phi(n) = \tan^{-1}\left(R_{TMxy} \Big/ R_{TMz} \right) \qquad \textbf{(8-68)}$$

The "n" in Equation 8-68 depicts the number of the frame during simulation. If $R_{TMz} < 0$ that means the target has gone behind the missile and the negative angle $\phi(n)$ needs to be corrected by adding π. Figure 8-22 explains the change of sign for the negative $\phi(n)$.

$$\phi(n) = \phi(n) + \pi \qquad \textbf{(8-69)}$$

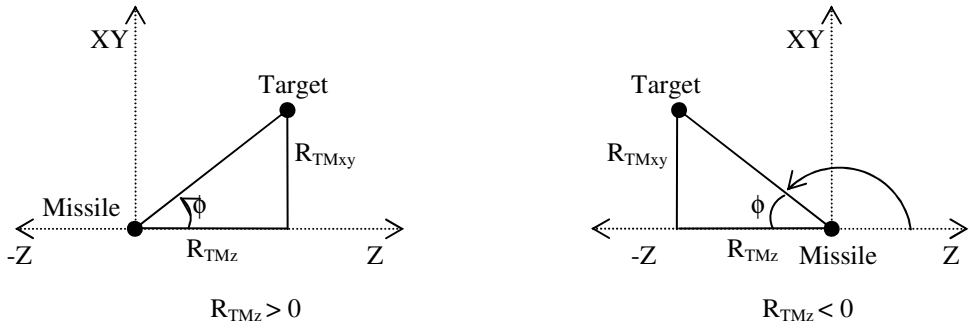


Figure 8-22 : Calculating missile angle

8.10.10 Modelling Missile Lateral Acceleration Limit

To incorporate the missile lateral acceleration (LATAX) limit, the missile ROT_{msl} is limited as per the missile load factor (*G's*) and the speed (V_{msl}). From the *ROT* calculated in Equation 8-11, the maximum angle that the missile can turn in one frame or time interval (Δt) is calculated by multiplying ROT_{msl} with Δt.

$$ROT_{msl\_int} = ROT_{msl} \cdot \Delta t \qquad \textbf{(8-70)}$$

Initially, LATAX was implemented by comparing the missile angle $\phi(n)$ as given in Equation 8-69 with $ROT_{msl\_int} \cdot n$ at every frame. Although, this logic works satisfactorily for situations when the missile is constantly turning in one direction, however, once the target crosses the missile-LOS and the missile starts to move in the opposite direction, this simple logic was not limiting the LATAX. This problem is resolved by comparing the difference between the missile rotation angle (ϕ) at two consecutive time intervals. For the missile to operate within the LATAX limit, this difference must be less than the ROT_{msl_int}.

$$\left|\phi(n-1) - \phi(n)\right| < ROT_{msl\_int} \qquad \textbf{(8-71)}$$

If the logic of Equation 8-71 is true, the missile new angle $\phi(n)$ is the same as that calculated in Equation 8-69, otherwise, the $\phi(n)$ is limited. If angle $\phi(n)$ is increasing then ROT_{msl_int} is added to the angle $\phi(n)$.

$$\phi(n) = \phi(n-1) + ROT_{msl\_int} \qquad \textbf{(8-72)}$$

Otherwise, if $\phi(n)$ is decreasing then ROT_{msl_int} is taken away from the angle $\phi(n)$.

$$\phi(n) = \phi(n-1) - ROT_{msl\_int} \qquad \textbf{(8-73)}$$

The flow chart shown in Figure 8-23 explains the logic for limiting the missile ROT as per the LATAX limit. However, for the first frame (i.e. n = 1), in Equation 8-71 the "n-1" term is not defined. To overcome this problem, for the first two frames the angle $\phi(n)$ is calculated by multiplying ROT_{msl_int} with "n".

$$\phi(n) = ROT_{msl\,\mathrm{int}} \times n \qquad \textbf{(8-74)}$$

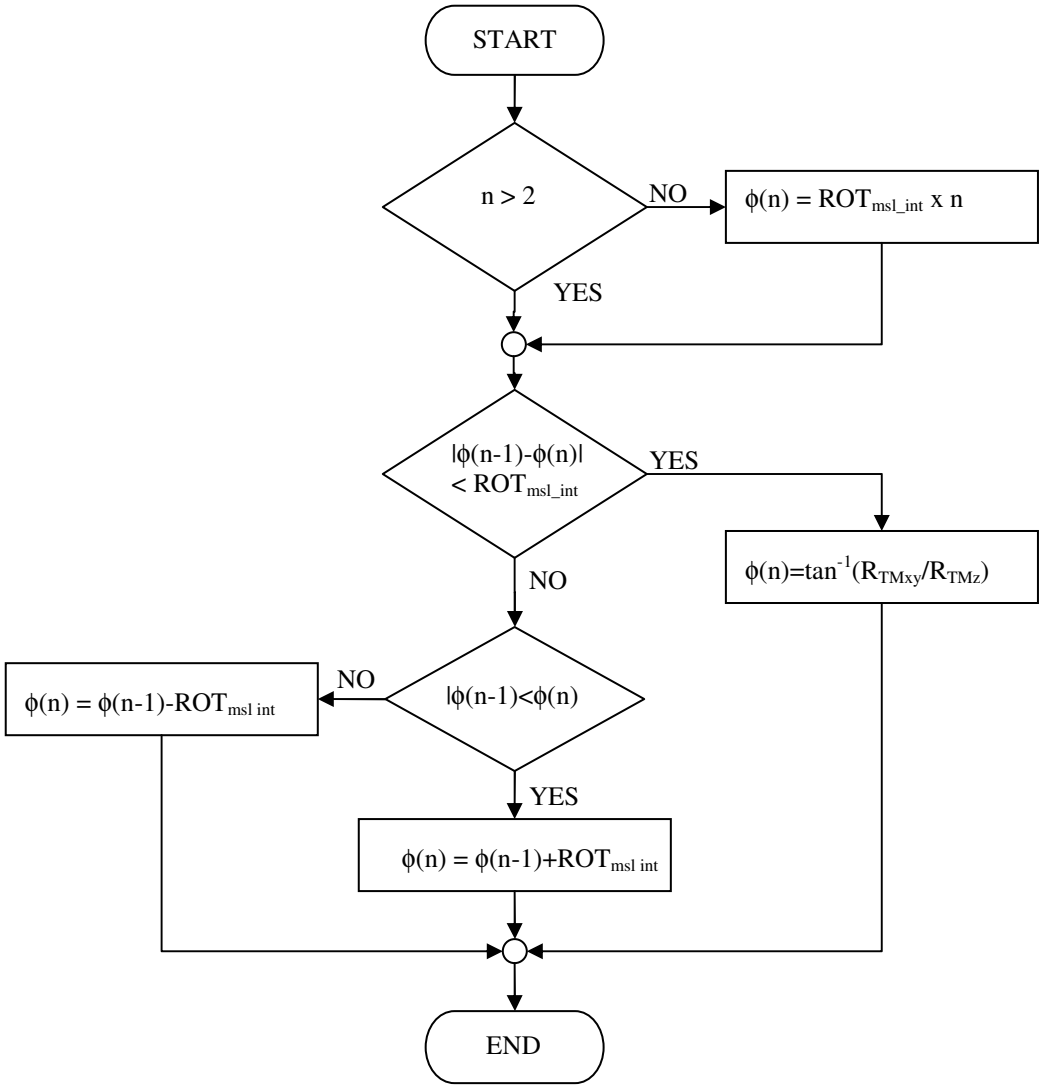


Figure 8-23 : Flow chart explaining missile ROT limit logic

After limiting the missile angle ϕ as per the LATAX, the missile new position is calculated as per the new ϕ value. The R_{TMxy} is calculated from R_{TMz} as given in Equation 8-75 and illustrated in Figure 8-13(b).

$$R_{TMxy} = \left| R_{TMz} \tan \phi(n) \right| \qquad \textbf{(8-75)}$$

If the target is on the left of the missile (ie $X_{tgt} < X_{msl}$) then the X-component of the increment in the target-missile distance (ΔX_{TM}) is calculated as given in Equation 8-76

$$\Delta X_{TM} = -R_{TMxy} Cos(|\zeta_T|) \quad \textbf{(8-76)}$$

Otherwise, if the target in on the right of the missile (ie $X_{tgt} > X_{msl}$) then ΔX_{TM} is positive as given in Equation 8-77.

$$\Delta X_{TM} = R_{TMxy} Cos(|\zeta_T|) \quad \textbf{(8-77)}$$

Similarly, for the Y-component, if the target is below the missile (ie $Y_{tgt} < Y_{msl}$) then,

$$\Delta Y_{TM} = -R_{TMxy} Sin(|\zeta_T|) \quad \textbf{(8-78)}$$

And for the target above the missile (ie $Y_{tgt} > Y_{msl}$) then ΔY_{TM} is a positive value.

$$\Delta Y_{TM} = R_{TMxy} Sin(|\zeta_T|) \quad \textbf{(8-79)}$$

The missile increment along the Z-axis is calculated by multiplying the missile-step ($step_{msl}$) calculated in Equation 8-18 with the Cosine of $\phi(n)$.

$$step_{msl\_z} = step_{msl} \times Cos\phi(n) \quad \textbf{(8-80)}$$

The $step_{msl\_z}$ calculated in Equation 8-80 is taken as a negative value when R_{TMz} is decreasing and a positive value for increasing R_{TMz}.

To steer the missile towards the target, the 3D location of the missile in the next frame (n+1) is calculated. The $X_{msl}(n+1)$ and $Z_{msl}(n+1)$ given in Equation 8-81 and 8-83 are calculated using the two similar triangles shown in Figure 8-24. Similarly, the $Y_{msl}(n+1)$ given in Equation 8-82 is calculated by solving the similar triangles along the Y-axis.

$$X_{msl}(n+1) = \Delta X_{TM} . \frac{|step_{msl\_z}|}{|R_{TMz}|} + X_{msl}(n) \quad \textbf{(8-81)}$$

$$Y_{msl}(n+1) = \Delta Y_{TM} . \frac{|step_{msl\_z}|}{|R_{TMz}|} + Y_{msl}(n) \quad \textbf{(8-82)}$$

$$Z_{msl}(n+1) = step_{msl\_z} + Z_{msl}(n) \quad \textbf{(8-83)}$$

Using these values the "*translation*" field of the missile-LOS given in Equation 8-3 is updated at each frame.

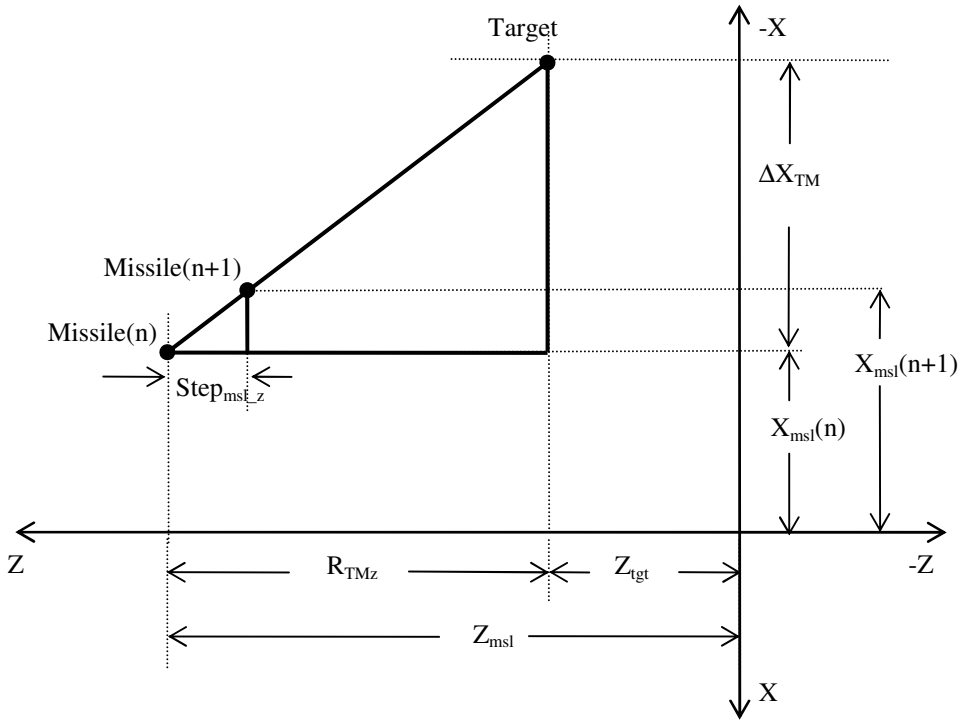


Figure 8-24 : Explaining missile new position in next frame

8.10.11 Updating Missile_LOS Rotation Field

The missile rotation field given in Equation 8-4 is updated at every frame by calculating the direction-cosines and the rotation angle of the missile_LOS. The direction-cosines are calculated using ζ_T calculated in Equation 8-67. For, $X_{msl} < X_{tgt}$, the ζ_T is a negative angle and π is added to make ζ_T a positive value. The direction-cosines calculated in Equation 8-85 to 8-87 are used to update the rotation field values in Equation 8-4. The missile_LOS rotation angle $\phi_{msl\_LOS}(n)$ is the same angle as calculated after the LATAX limit and explained in flow diagram of Figure 8-23.

$$\zeta_T = \zeta_T + \pi \qquad \textbf{(8-84)}$$

$$X_{m\_LOS} = Sin(\zeta_T) \qquad \textbf{(8-85)}$$

$$Y_{m\_LOS} = Cos(\zeta_T) \qquad \textbf{(8-86)}$$

$$Z_{m\_LOS} = 0 \qquad \textbf{(8-87)}$$

8.11 Missile Miss-distance and Hit-Criterion

In homing missiles, the contributors to missile miss-distance are the seeker errors, autopilot lag, target manoeuvres, target state estimation lag etc. The hit-criterion is based on the type of fuse and effective range of the warhead used. The fuse may be a simple impact device or a complex "*fisheye*" lens proximity system. Typically, the missile reaching less than half the minimum dimensions of the target is considered as a hit [MEN03]. If the missile satisfies the hit-criterion then it is considered as a direct hit. Otherwise, the missile has missed the target. The missile miss-distance is the minimum distance between the missile and the target aircraft at any time during the engagement. The miss-distance is calculated between the geometrical origins of the missile and the target or in other words for miss-distance the missile and target are considered as point objects.

The hit-criterion is incorporated in the algorithm by considering that if the target aircraft enters the cylindrical region ahead of the missile as shown in Figure 8-25 then it is a hit, otherwise the missile has not hit the target. The radius of the cylinder is entered as R_{impact} as an input parameter and the length is taken as the $step_{msl}$ calculated in Equation 8-18. During the simulation, at every frame the R_{err} calculated in Equation 8-56 is compared with R_{impact} and the R_{TMD} calculated in Equation 8-42 is compared with $step_{msl}$. If the logic given in Equation 8-88 is satisfied, then missile hits the target.

$$(R_{err} < R_{impact}) \,\&\, (R_{TMD} < msl_{step}) \qquad \textbf{(8-88)}$$

However, in the case where the missile misses the target aircraft, the miss-distance is calculated as the largest out of the R_{TMD} and R_{err} used in the hit-criterion logic in Equation 8-88.

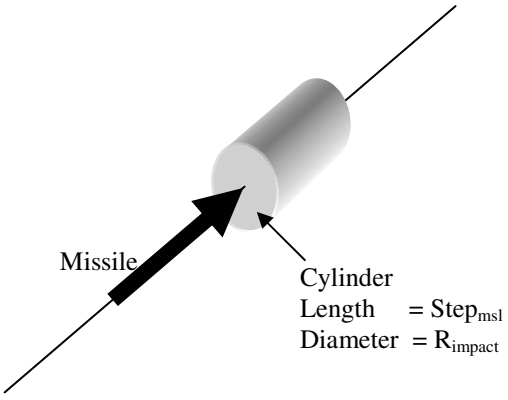


Figure 8-25 : Considering Hit-criterion as a cylindrical region in front of missile

In the virtual world, the R_{TMD} represents the distance from the centre of the target body to the head of the missile and the R_{err} denotes the magnitude of the error from the centre of the target aircraft. However, to increase the fidelity of the model, the hit-criterion may be improved by adding the "*collision*" node and integrating this with "*avatarSize*" fields in the virtual world. Presently, due to shortage of time, this feature is not incorporated in the algorithm.

8.12 Conclusion

The missile-target engagement simulation is modelled by controlling the "*translation*" and "*rotation*" fields of the missile and target aircraft by the algorithm implemented in MATLAB. The target aircraft may fly with 4-DOF and can perform different manoeuvres in straight-and-level, descent and turn flight paths. The missile movement is modelled in 5-DOF. The missile is modelled with a gimballed seeker head. The binary centroid and intensity centroid trackers are implemented in the algorithms. The missile seeker/detector captures the 2D image from 3D virtual world. This is implemented using MATLAB Virtual-reality Toolbox. The Pursuit-course guidance is modelled for steering the missile towards the target aircraft. The lateral acceleration (LATAX) is modelled by limiting the rate-of-turn as per the missile speed and the load-factor. The missile hit-criterion is implemented as a cylindrical region in front of the missile. The target manoeuvrability and the missile guidance and control in 3D are modelled to simulate the missile-target engagement sequences for the analysis of the IR signature of the targets and the countermeasure flares.