Coordinate Systems Functionality Description And Applications

By

Hilaire M. Kadjo

A MASTER OF ENGINEERING REPORT

Submitted to the College of Engineering at

Texas Tech University in

Partial Fulfillment of

The Requirements for the

Degree of

MASTER OF ENGINEERING

Approved

Dr. A. Ertas

Dr. T. Maxwell

Dr. M. Tanik

Dr. J. Woldstad

October 27, 2007

ACKNOWLEDGEMENTS

This report would not have been possible without the support of my family, friends and colleagues and I would like to thank everyone who has helped and inspired me during this master's program.

I thank Mr. Johnny Barrett for hiring me to his team. He opened a door for me and I needed to know how to enter myself. The tremendous experience of Max Lee inspired me to pursue this master's degree in engineering. Max Lee has been my mentor through my career at Raytheon. He always offers advice and suggestions whenever I need them and I would like to thank him for helping me through this process and also for subjecting my text to rigorous review.

I would like to express my thanks to the entire Texas Tech staff and guest speakers. Dr. Ertas has been wonderful, devoting his weekend for Raytheon employees to get the knowledge in order to serve our country better. Also I want to let everyone know that Dr. Ertas invited all the students for lunch at his cost every Saturday since the beginning of our program. Thanks Dr. Ertas.

My life has not been easy, but believing in God has helped me go through all the tough and difficult moments. The strength that the Almighty gives me every day helps me move forward and I want to thank him for all he has done for me and my family. May his name be exalted, honored and glorified.

And finally never enough thanks to one who doesn't want to be named but she knows who she is and so do I. Working full time and being a full time student has taken away most of my family time. But my wife understands and has been very supportive during this difficult moment. Thanks sweetie for all your support during this time.

TABLE OF CONTENTS

DISCLAIMER

ABSTRACT

Today's systems contain multiple, independent sensors. The information from each sensor is easier to calculate and understand in a coordinate system that is related to a specific sensor measurement (e.g., radar, infrared, etc) instead of the reference frame that is desired by the user.

Coordinate transformations are of great importance to convert information between the various reference frames. For example when a Laser Range Finder is used to estimate the geographical location of an object, we convert the range vector into the local coordinate system and add it to our geographical position. There are many different coordinate systems, based on a variety of geodetic datums (geodetic datums defines the size and shape of the earth and the origin and orientation of the coordinate systems used to map the earth), units, projections, and reference systems in use today. The common coordinate frame used is the geodetic system which gives the latitude (North or South angle between any point and the equator), longitude (East or West angle of any point and the Greenwich meridian) and the altitude (elevation).

This presentation is educational and defines some basic coordinate systems and the mathematical processes used to show transformations between them. It provides a brief description of local and global systems for use in precise positioning, navigation, and geographic information systems for the location of points in space [1]. First we focus on several different coordinate systems and their application for the user. Finally we present some test cases, verification and validation of several algorithms in this document.

LIST OF FIGURES

CHAPTER I
INTRODUCTION

A coordinate system is a reference frame which is used for measurements of angles, directions, locations and distances. A coordinate system in a plane or in n-dimensional space is a systematic method of assigning a pair or an n-tuple of numbers to a point in the plane or in n-dimensional space (respectively) which describe its position uniquely. Therefore a coordinate is a component of a vector in a vector space and so a coordinate system is used to represent a vector space. For example a point in space with the coordinates (x, y, z) as (-1, 2, 1) means that this 3-tuple point is -1 unit away from the origin on the x-axis, +2 units away from the origin on the y-axis, and -1 unit away from the origin on the z-axis. The origin is the point where the x-axis, the y-axis and the z-axis intercept. Therefore any representation or plot of a point always uses a coordinate system so that the representation has a meaning. Although a specific coordinate system is useful for numerical calculations in a given space, the space itself is considered to exist independently of any particular choice of coordinates [2].

Coordinate Systems have always been of great importance for use in precise positioning, navigation, and geographic information systems for the location of points in space. The need to use more than one coordinate system arises from the fact that many different physical phenomena are easier calculated or understood in a system that is appropriate the phenomenon. The choice on the coordinate system used to solve a problem depends on the objectives that one wants to obtain as even though you change the coordinate system to represent a point, that point itself does not changes. For example a geographical coordinate system will express every point on earth in the spherical using the latitude, the longitude and the altitude. In this case any point on earth can be specified. Sometimes this information may not be available; instead, other parameters are given. Therefore the need to transform from one coordinate system to another is necessary. Note that even though reference system and reference frame are most of the time interchanged, it is important to know the difference between these two concepts. A reference system is the conceptual idea of a particular coordinate system, while a reference frame is the practical realization of a reference system by observations and measurements.

This document is for educational purpose and defines some basic coordinate systems and the transformations or algorithms between them. There are many different coordinate systems, based on a variety of geodetic datums (geodetic datums defines the size and shape of the earth and the origin and orientation of the coordinate systems used to map the Earth), units, projections, and reference system in use today. Coordinate systems will help Raytheon engineers to understand better how Raytheon sensors information can be useful by converting this information to for example a North East Down (NED) coordinate frame.

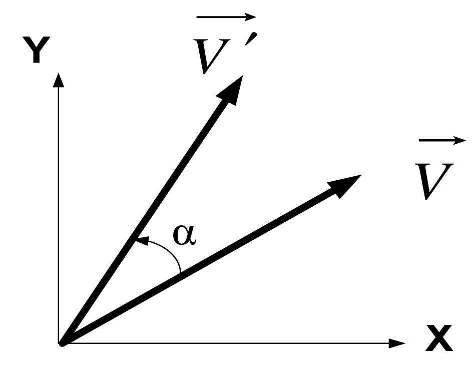
CHAPTER II
BACKGROUND

We will define some basic coordinate systems which will include the Cartesian Coordinates, the Polar Coordinates, the spherical Coordinates and the transformations between them.

2.1 Rotation vs. Transformation

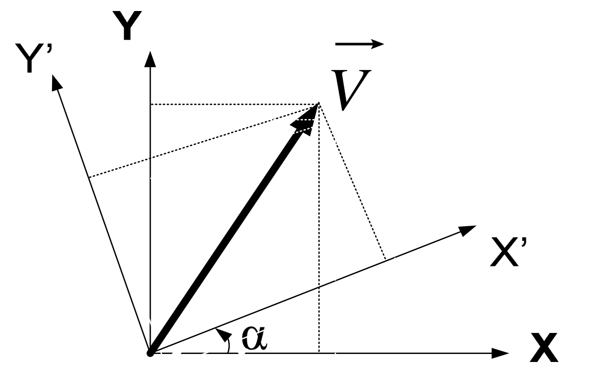
We define an operational concept for a solution by describing the different parameters used in the coordinate systems definition. The angles described above are rotation or transformation angles and are defined based on the mathematical principals below:

- The position of a rigid object in space can be defined by 6 parameters: 3 transformations plus 3 rotations.
- A “rotation” matrix rotates a vector in a given reference frame.
- A “transformation” matrix transforms a vector from one reference frame to another reference frame.
- Rotation and Transformation are related by transposition.
- The inverses of transformation and rotation matrices are their transposes.



$$\overrightarrow{V'} = R\overrightarrow{V} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \overrightarrow{V}$$

Figure 1. Rotation.



$$\vec{V}' = T\vec{V} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \vec{V}$$

Figure 2. Transformation.

The following are true for rotation and transformation.

$$[R] = [T]^t = [T]^{-1}$$

$$[T] = [R]^t = [R]^{-1}$$

The transformation matrices of an angle θ around the positive X, Y and Z axes are given by:

$$T(X,\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$T(Y,\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$T(Z,\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.2 Right-handed vs. Left-handed

A rotation about a directed axis is called positive, or right-handed if it causes a right-handed screw to advance in the positive direction of the axis.

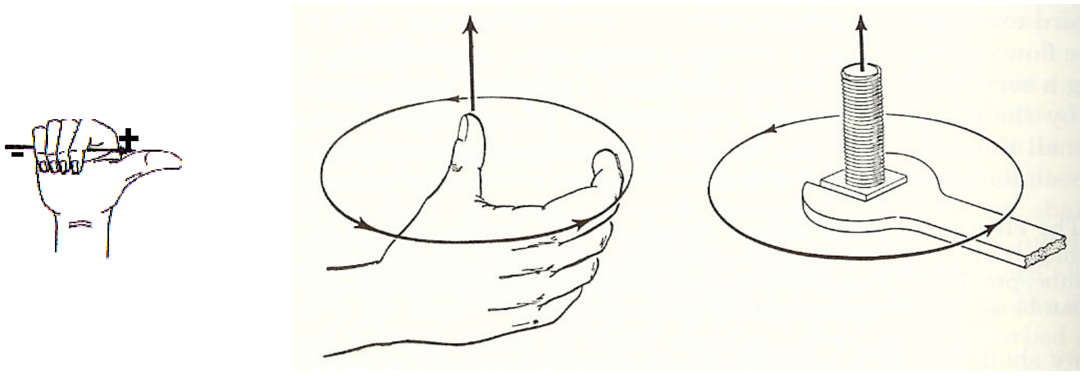


Figure 3. Right-handed Screw.

The default coordinate system is the right-handed coordinate system. In this case, the positive X and Y axes point out of the paper and right and up respectively and the positive Z axis points up. The positive rotation is counterclockwise about the axis of rotation illustrated in the following below:

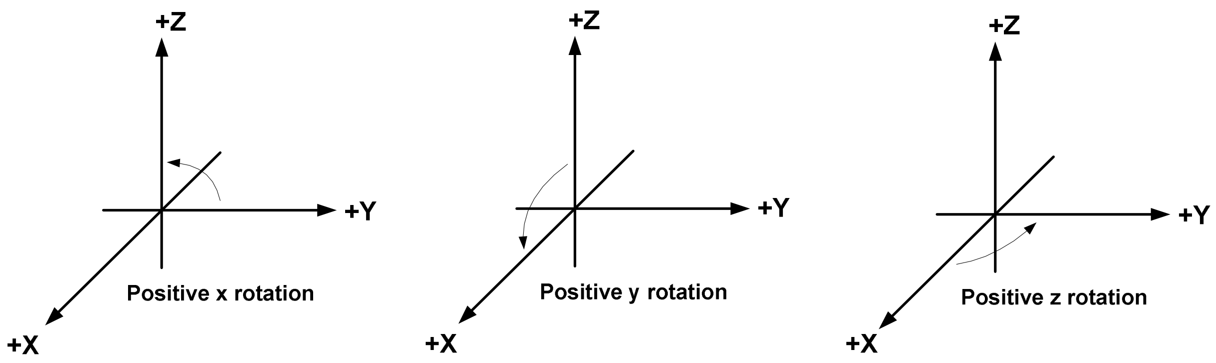


Figure 4. Right-handed Coordinate Frames.

In the left-handed coordinate system, the positive X, Y and Z axes point forward, right and down respectively. The positive rotation is clockwise about the axis of rotation as illustrated in the figures below:

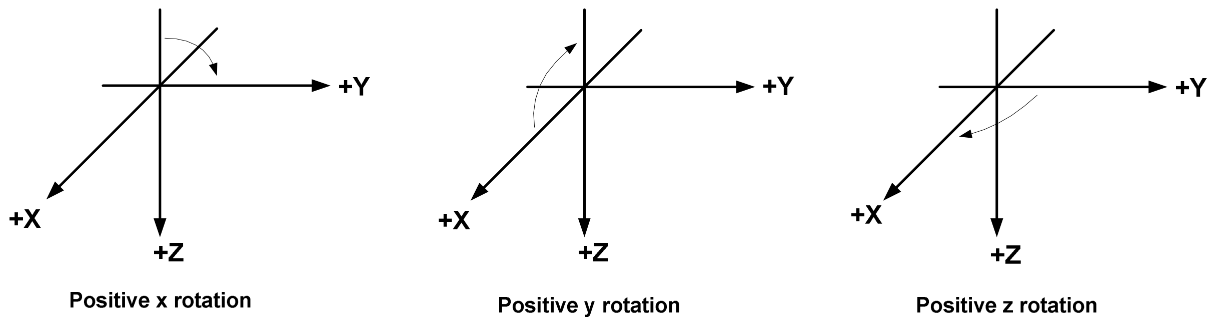


Figure 5. Left-handed Coordinate Frames.

A standard right-handed coordinate system with right-handed rotation about the positive X, Y, Z axes are shown below:

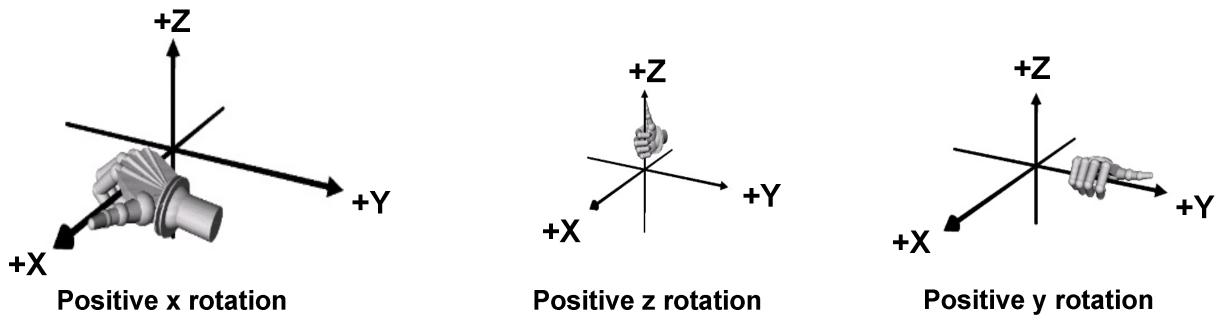


Figure 6. Right-handed Rotation.

- The right handed motion along +X would cause +Y to move toward +Z
- The right handed motion along +Y would cause +Z to move toward +X
- The right handed motion along +Z would cause +X to move toward +Y

2.3 Standard Cartesian Coordinate Systems

2.3.1 Two Dimensional Coordinates

The two dimensional Cartesian coordinate system is used to represent a point uniquely in the plane. It is usually defined by two orthogonal axes, the x-axis and the y-axis defining a plane that is referred to as the xy-plane. Given each axis, choose a unit length, and mark off each unit along the axis, forming a grid. To specify a particular point on a two dimensional coordinate system, indicate the *x* unit first, followed by the *y* unit in the form (x,y), called an ordered pair.

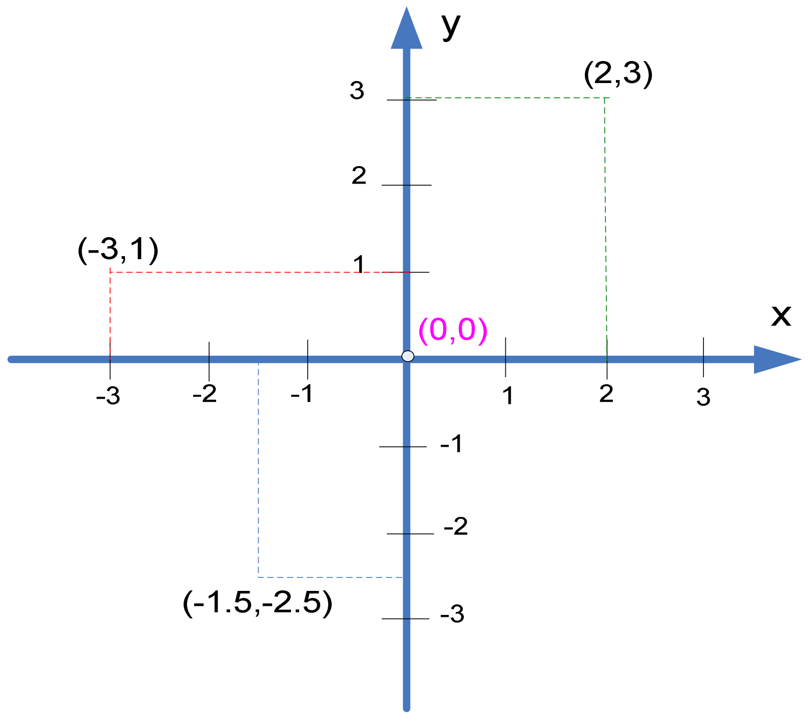


Figure 7. 2-D Cartesian Coordinate Frame.

2.3.2 Three Dimensional Coordinates

The three dimensional Cartesian coordinate system provides the three physical dimensional of the space which includes length, width and height, usually represented by three orthogonal axes at the intersection point, the x-axis, the y-axis and the z-axis.

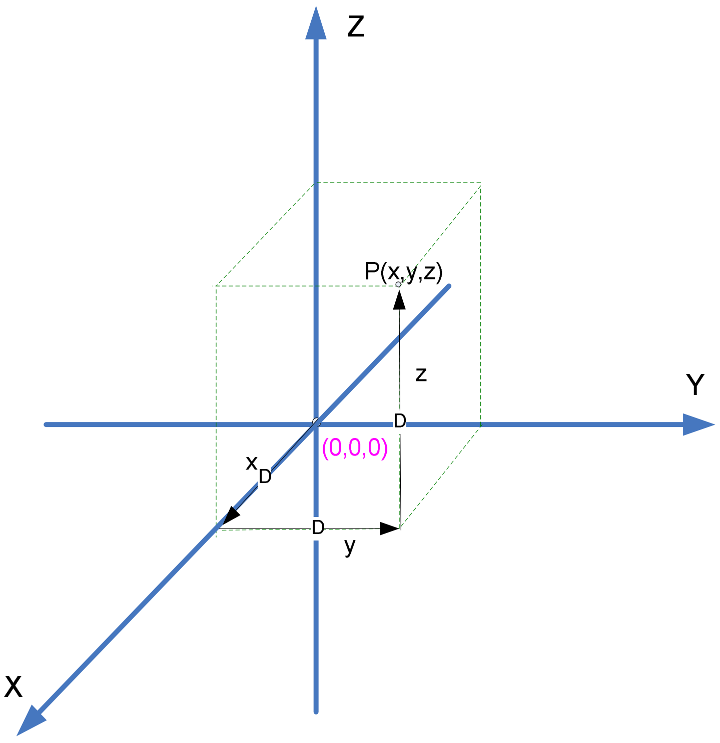


Figure 8. 3-D Cartesian Coordinate Frame.

2.4 Polar Coordinates

The polar coordinate system is a two-dimensional coordinate system in which each point on a plane is determined by a distance and an angle.

2.4.1 Circular Coordinates

The circular coordinate system is a two-dimensional polar coordinate system, defined by an origin and a polar axis. A point is represented by a pair (r,θ) where the radius r is the distance from the origin to the point and the azimuth θ ($0 \leq \theta \leq 360$℃) is the angle between the positive *x*-axis and the line from the origin to the point.

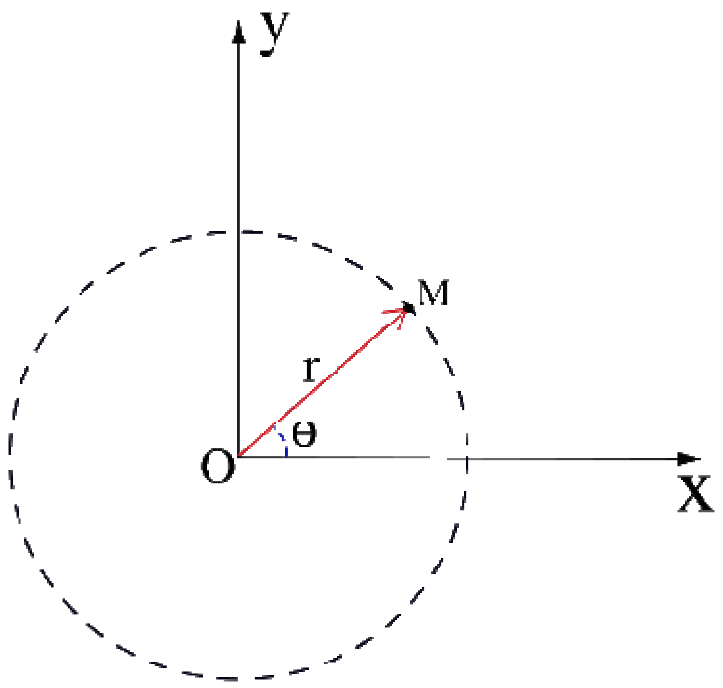


Figure 9. Circular Coordinate Frame.

2.4.2 Cylindrical Coordinates

The cylindrical coordinate system is a three-dimensional coordinate system which extends the circular polar coordinate by adding a third coordinate to measure the height of a point above the plane. A point in a cylindrical coordinate is represented by (r,θ,h).

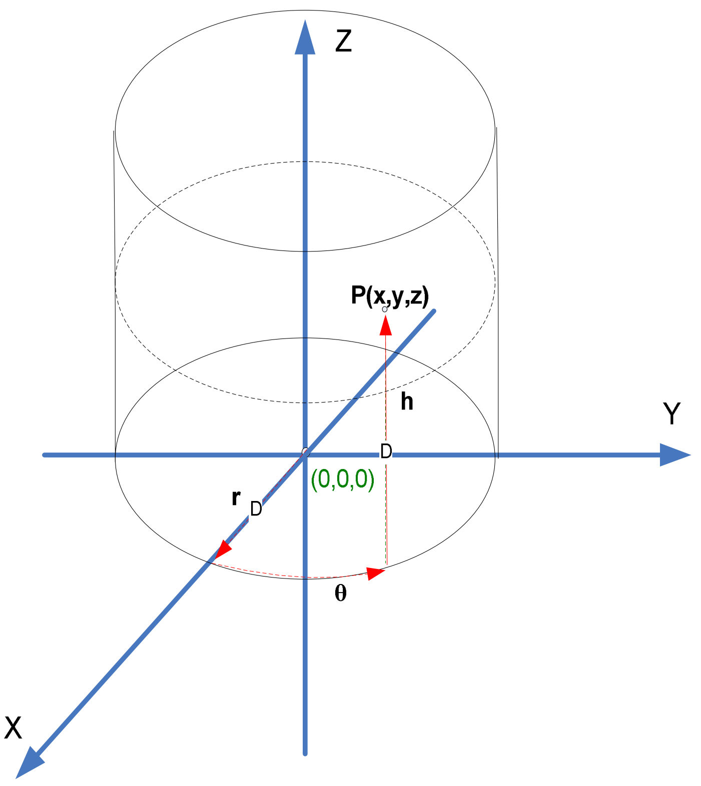


Figure 10. Cylindrical Coordinate Frame.

2.4.3 Spherical Coordinates

The spherical coordinate system is a coordinate system for representing geometric figures in three dimensions. A point is represented by (ρ,φ,θ), where $\rho$is the radial distance of a point from the origin, φ is the zenith angle from the positive z-axis and θ is the azimuth angle from the positive x-axis.

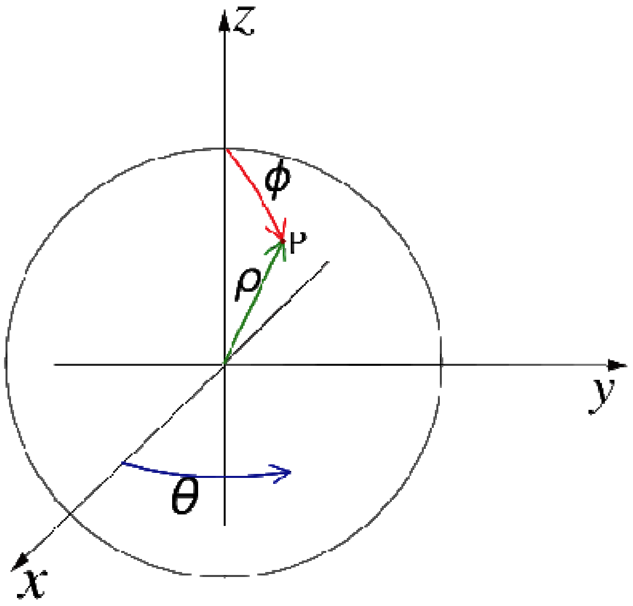


Figure 11. Spherical Coordinate Frame.

2.5 Transformations between some basic coordinate systems

2.5.1 Cylindrical vs. Spherical Coordinates

Given a point (r, θ, h) in the cylindrical coordinate system can be converted in the spherical coordinate system by the following formulas:

$$\begin{cases} \rho = \sqrt{r^2 + h^2} \\ \varphi = \mathrm{atan2}(r, h) \\ \theta = \theta \end{cases}$$

And vice versa, a point (ρ, φ, θ) in the spherical coordinate is represented in the cylindrical coordinate with the following formulas:

$$\begin{cases} r = \rho sin\varphi \\ \theta = \theta \\ h = \rho cos\varphi \end{cases}$$

2.5.2 Cartesian vs. Spherical Coordinates

The figures below show the conversions from the 3-D Cartesian coordinate to the Spherical coordinate:

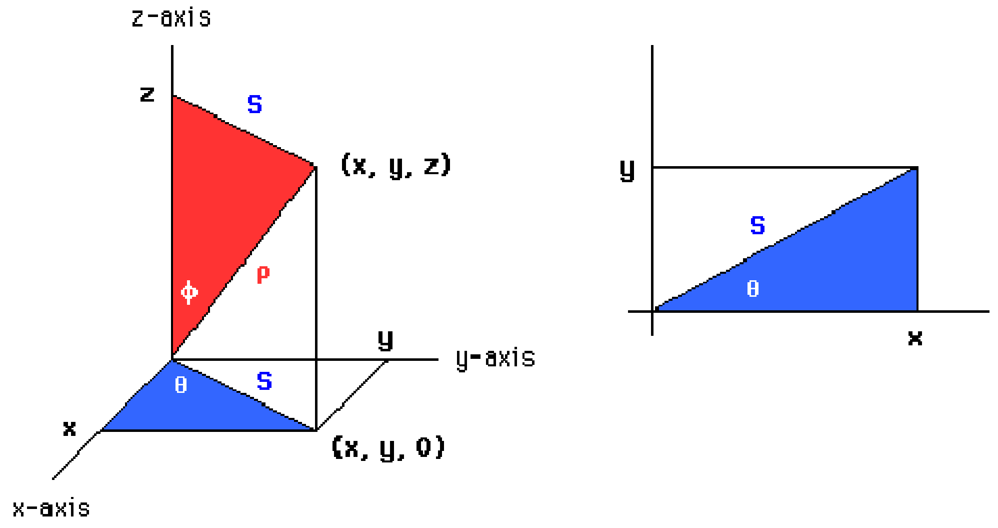


Figure 12. Transformations Processes.

The blue color illustrates the xy-plane from both figures. Using the Pythagorean Theorem we obtain the following formulas:

$$\begin{aligned} S &= \sqrt{x^2 + y^2} = \rho \sin \phi \\ \rho &= \sqrt{S^2 + z^2} = \sqrt{x^2 + y^2 + z^2} \\ x &= S \cos \theta \\ &= \rho \sin \phi \cos \theta \\ y &= S \sin \theta \\ &= \rho \sin \phi \sin \theta \\ z &= \rho \cos \phi \end{aligned}$$

Therefore the conversion from the Spherical coordinate to the 3-D Cartesian coordinate is represented by the given formulas:

$$\begin{cases} x = \rho sin\varphi cos\theta \\ y = \rho sin\varphi sin\theta \\ z = \rho cos\varphi \end{cases}$$

Conversely we have the following:

$$\begin{aligned} \rho &= \sqrt{x^2 + y^2 + z^2} \\ S &= \sqrt{x^2 + y^2} \\ \phi &= \arccos\left(\frac{z}{\rho}\right) \\ \theta &= \begin{cases} \arcsin(y/S), & \text{if } 0 \le x; \\ \pi - \arcsin(y/S), & \text{if } x < 0. \end{cases} \end{aligned}$$

This gives the conversion formulas from Cartesian to Spherical coordinates given by:

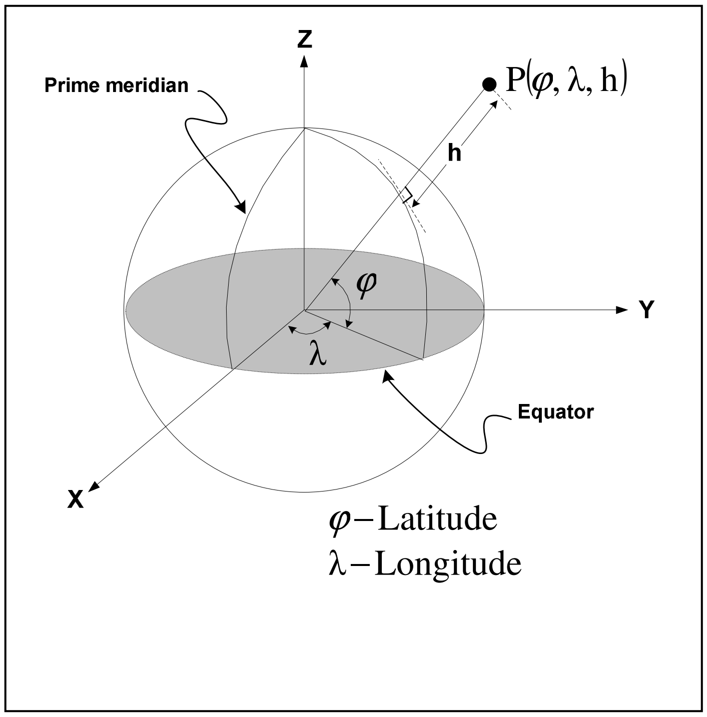
$$\begin{cases} \rho = \sqrt{x^2 + y^2 + z^2} \\ \varphi = atan2\left(\sqrt{x^2 + y^2}, z\right) = arcos\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right) \\ \theta = atan2(y, x) \end{cases}$$

With the following constraints:

$$\begin{cases} \rho \ge 0 \\ -\frac{\pi}{2} \le \varphi \le +\frac{\pi}{2} \\ -\pi \le \theta \le +\pi \end{cases}$$

2.5.3 Cartesian vs. Spherical Coordinates

Depending on whether one considers the earth as a Sphere or an Ellipse, a point P on earth is represented in the figures below:



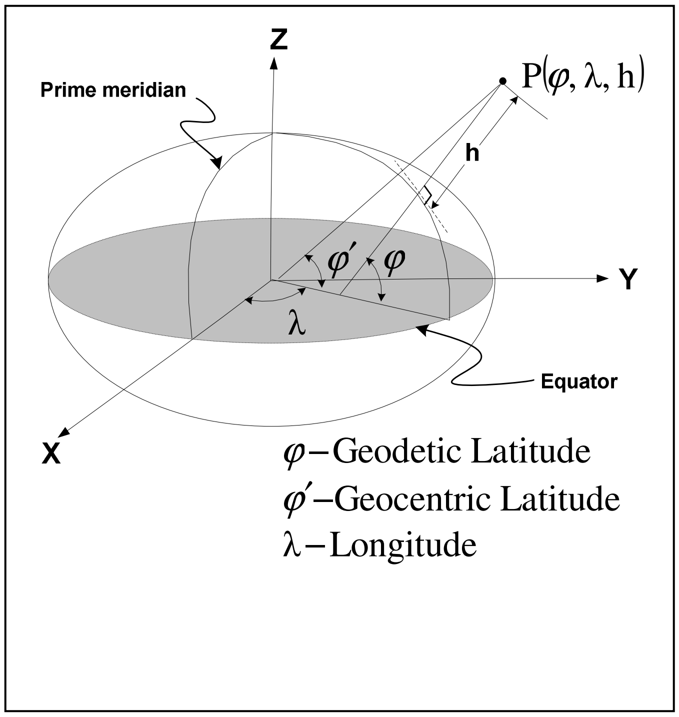


Figure 13. Spherical vs. Ellipsoidal.

CHAPTER III
REQUIREMENTS AND COORDINATE SYSTEMS APPLICATIONS

3.1 Project Description

Besides finding the location of a target and computing the distance between targets, Coordinate Systems have several applications with the EOIR Sensor, the GPS and the ANS. These sensors combined with several platforms help in Target Tracking, Gap and Distance, Sensitivity Map, Blind Arc, Sight to Crest, Follow Up Surveillance, etc. Only platform independent coordinate systems will be discussed in this document. System engineering principles will be used to derive and implement some algorithms between these platform independent coordinate systems.

Build a System of Coordinate Frames that allows one to convert the coordinates of any given point in a given coordinate system to a different coordinate frame using the Systems Engineering and Architecture Principles [3, 4]. Some applications will include the GPS, EOIR and the ANS sensor coordinate systems.

The project will consider the different transformations and derive some algorithms that will help in computing Target location, Gap and Distance, the Sight to Crest, Follow up Surveillance or Blind Arc. Because this document will be available to the general public, only general information will be provided concerning the above applications. However different applications will be considered to illustrate how System Engineering Process helps in problem solving. The information in this document will be solely for educational purpose. The figure below gives an overview.

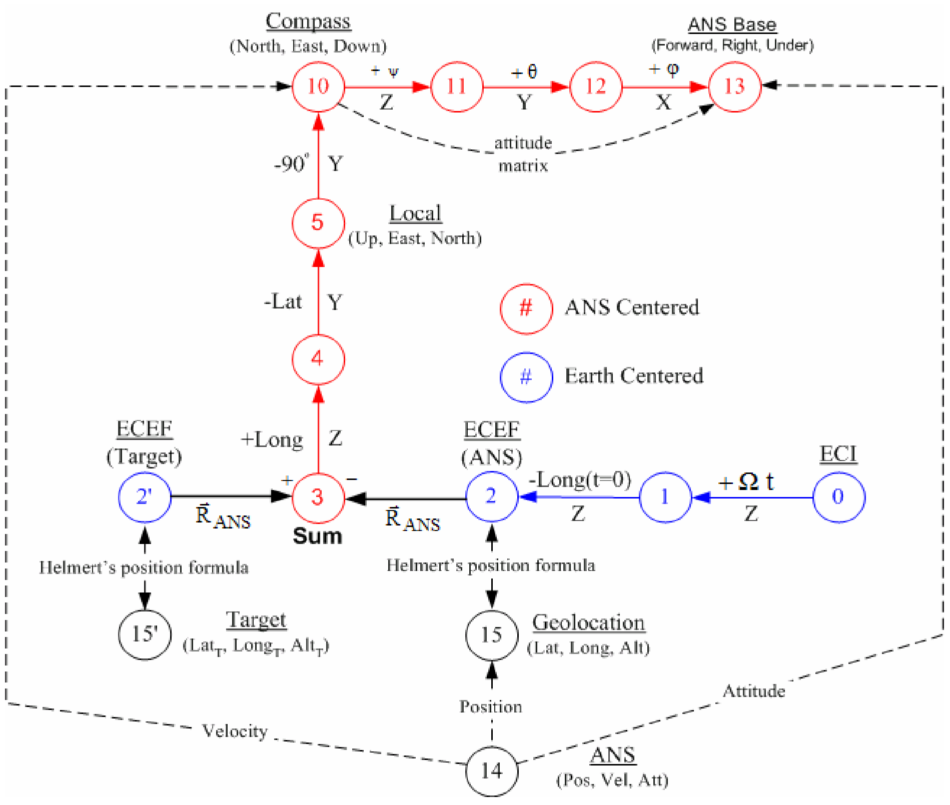


Figure 14. Platform Independent Coordinate Systems Overview.

Normal arrows between nodes represent direction of rotation angles about an axis.

Node 0 $\rightarrow$ node 1: Ωt is rotational angel about Z axis

Origin of axes is unchanged

Bold arrows between nodes represent vector shifts from origin.

Node 2 $\rightarrow$ node 3: R_{ANS} shifts from ECEF to ANS centered;

V3 = V2' - V2

V3 (vector from ANS to target)

V2' (ECEF vector of target)

V2 (ECEF vector of ANS)

Direction of axes are unchanged.

- Transformation from node 10 to node 11 is obtained by a z-axis rotation about $+\psi$ radians (nose right of north)
- Transformation from node 11 to node 12 is obtained by y-rotation about $+\theta$ radians (nose about horizon)
- Transformation from node 12 to node 13 is obtained by a x-axis rotation about $+\varphi$ radians (right side down)

3.2 Contextual Architecting

3.2.1 Objectives

Develop an algorithm document of library functions involving the EOIR and ANS sensors and associated coordinate systems. The high level requirements are listed below and any possible derived requirement will follow during the development of this project.

- The algorithm document shall define some basic coordinate systems and the mathematical processes used to show transformations between them.
- The algorithm document shall show data flows among the system functions.
- The algorithm document shall focus on how a vector in one coordinate system is represented in another coordinate system in 3-D.
- The algorithm document shall show some real life applications involving the systems of coordinates used.
- The algorithm document shall illustrate each algorithm showing some test cases.
- The computations in these algorithms shall be more accurate than the sensor measurements.
- The library functions derive from the algorithms shall execute fast.

3.2.2 Tasks

The tasks for the Contextual Architecture consist of gathering some interrogative data in order to build the system of coordinate. By answering the Who, What, Where, Why, How and When questions one tries to identify the following:

- Organize interrogative data
- Perform review of interrogative data
- Define scope and boundaries
- Perform review of scope and boundaries
- Identify and prioritize quality attributes
- Perform review of quality attributes

3.2.2.1 Why

The goals for the system of coordinate are listed below:

- Describe the relative position and orientation of objects.
- Define mathematical transformations between coordinate systems.
- Represent a view of the scene in multiple coordinate frames.
- Represent the positions in a user-defined "common" coordinate frame.
- Define coordinate frames so that vectors in different coordinate systems can be easily transformed and combined (this process normally occurs in ECEF coordinate frame instead of NED coordinate frame).
- Compute the Gap And Distance between objects using the Laser Range Finder. Will use only this example for this project.

3.2.2.2 Who

Stakeholders of the System of Coordinate frames are:

- Raytheon

- Texas Tech University
- General Public
- FCS Vehicles
- EOIR Sensor
- ANS
- Fire Control

3.2.2.3 What

The entities used are:

- Sensor coordinate frame
- EOIR coordinate frame
- Turret coordinate frame
- Barrel coordinate frame
- Vehicle coordinate frame
- ANS coordinate frame
- Compass coordinate frame
- ANS Base coordinate frame
- ECEF coordinate frame
- Geolocation coordinate frame

3.2.2.4 How

The System of Coordinate Frames will show the transformations between different coordinate frames.

3.2.2.5 Where

The System of Coordinate Frame will be used anywhere on Earth and on any platform (military vehicles or airplanes).

3.2.2.6 When

The System can be used at anytime, on a stationary or moving platform.

3.3 Operational Architecting

This part documents key operational information for the coordinate systems.

3.3.1 Objectives

Define the different coordinate frames involved in the system.

- ECI – Earth Centered Inertial. An inertial frame is a reference frame in which Newton's laws of motion apply, and so this coordinate frame does not rotate and also does not accelerate, but it translates with the earth. The origin of ECI is arbitrary and the axes may point in any three mutually perpendicular directions. The axes obey the right-handed coordinate systems. For practical reasons ECI may be defined to be coincident with the earth's center of mass at any given time. Classical mechanics assume the equivalence of all reference frames as time flows at the same rate in all reference frames.
- ECEF – Earth Centered Earth Fixed rectangular coordinate frame (a.k.a. geocentric) of the target (ECEFT) and of the ANS (ECEFA). The origin is fixed to the center of the earth and the axes rotate relative to the inertial frame at 7.3$\times 10^{-5}$ rad/s. The Z axis through the North Pole, the X axis through the intersection of the equator and the Greenwich Meridian (longitude and latitude are equal to zero) and the y axis is defined to complete the right-handed coordinate system. ECEF coordinate frame rotates and translates with the earth.

- Sensor – A sensor frame is defined for every direction sensitive axis. It includes each accelerometer, gyro and magnetometer axis. The sensor coordinate frame allows for formal accounting of sensor misalignment and the axes will be Forward, Up and Transverse.
- Body or Vehicle – The body or vehicle coordinate frame is the natural frame for control surface and vehicle attitude control. It determines the position of a vehicle based on measurements from various sensors attached to a sensor platform on the vehicle. It is attached to the vehicle of interest, usually at a fixed point such as the center of gravity. The orientation of the axis is not unique, but for aircraft and underwater vehicle the axes are defined as follows: the x axis is defined in the forward direction, the z axis is defined pointing to the bottom of the vehicle and the y axis completes the right-handed orthogonal coordinate system. This coordinate frame is frequently used in navigation and the body to platform transformation is important for calculating control outputs during active guidance.
- Platform – The platform coordinate frame defines three orthogonal axes with a fixed orientation relative to the body frame. The platform coordinate frame is transformed to the navigation frame when aggregating navigation information for vehicle control. Its coordinate axes are usually normally defined to have the same directions as the vehicle axes. The origin of the platform coordinate system is an arbitrary point on the platform, usually the location of the accelerometers. The origins of the body and the platform coordinate systems may be offset by a constant vector.
- Tangent Plane – The tangent plane coordinate frame is the locally assigned Cartesian coordinate frame. The ground level as at zero altitude and the axes are usually East, North and Up.
- Compass – North, East, Down reference frame (Geographic). The X axis through the local longitude line toward the North Pole and the Z axis down. The y axis points east to complete the orthogonal, right-handed rectangular coordinate system.
- Geolocation – ANS/GPS reference frame: Latitude, Longitude, Altitude. This coordinate frame is based on the WGS84 ellipsoid earth model.

- Navigation – Autonomous Navigation System (ANS). It is the coordinate frame used for path planning and other tracking and control purpose using the position, the velocity and the attitude.
- ANS Base – Forward, Right, Under reference frame.
- Target – LatitudeT, LongitudeT, AltitudeT of the target.
- Local – Up, East, North reference frame.
- Long(t=0) is the vehicle longitude at time = 0.
- Ω is the constant Earth rotational rate (7.3×10^{-5} rad/s)
- Long is ANS longitude.
- Lat is ANS geolocation latitude.
- ψ, θ, φ are ANS heading, pitch, roll about the Z, Y, X axis respectively.

3.3.2 Tasks

We define an operational concept for a solution by describing the different parameters used in the coordinate systems definition. The rotations and transformations used refer to the definitions in the background chapter.

3.4 Logical Architecting

We define a logical organization for the system of coordinates.

3.4.1 Objectives

- Describe the relative position and orientation of objects.
- Define transformations between coordinate systems.
- Represent a view of the scene in multiple coordinate frames.
- Represent the positions in a user-defined “common” coordinate frame.

- Define coordinate frames so that vectors in different coordinate systems can be easily transformed and combined (this process normally occurs in ECEF coordinate frame instead of NED coordinate frame).

3.4.2 Tasks

The diagram below defines a concept for a solution of the systems of coordinate frames. The works here are broken down from high level to low level coordinate systems using the principle of divide and conquer.

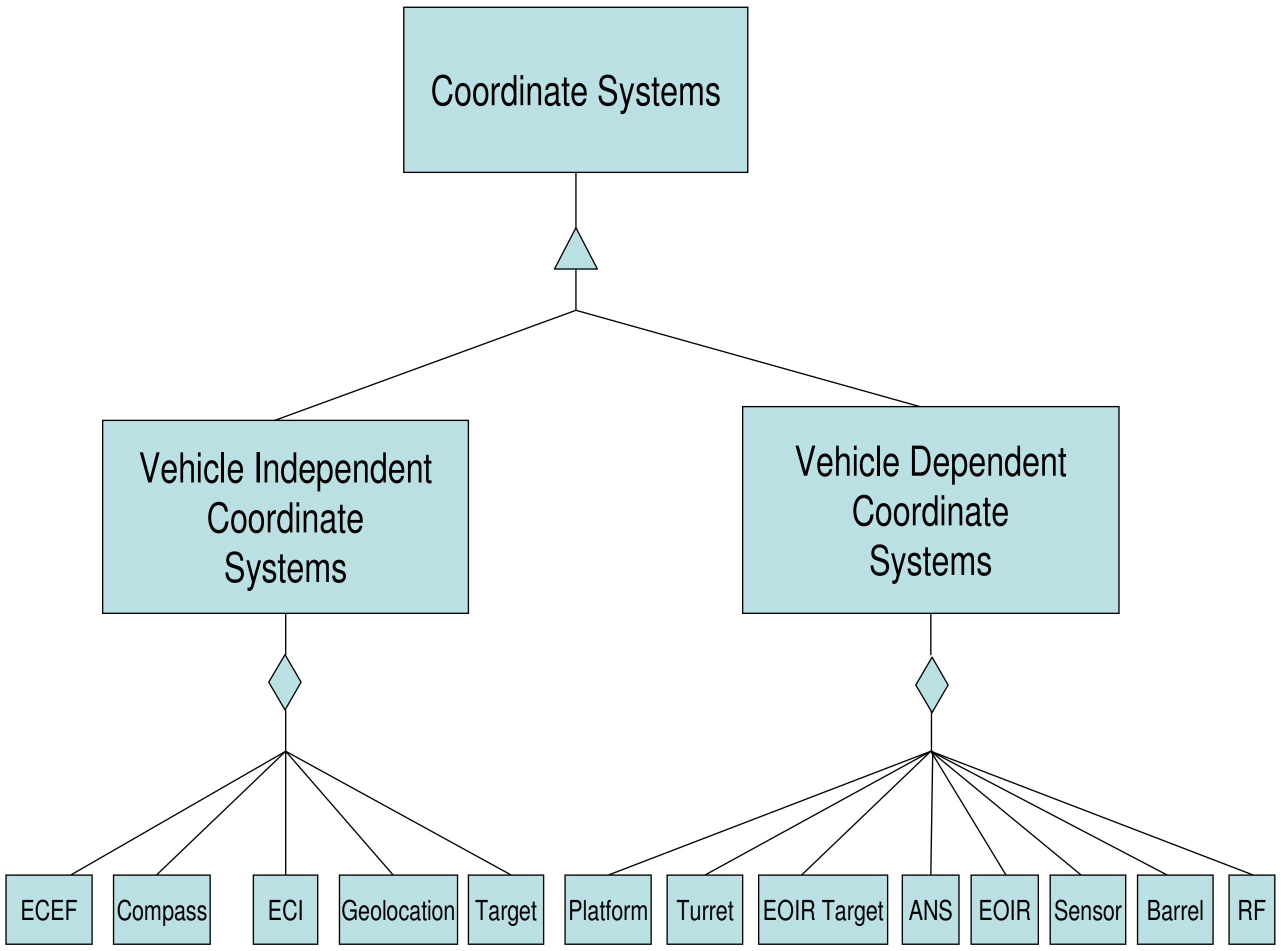


Figure 15. Coordinate Systems Flowchart

3.5 Physical Architecting

A physical organization, which includes the stakeholders, is defined as follow:

3.5.1 Objectives

Show some examples from the coordinate frames below:

- Compass (NED) to ANS Base.
- Compass to Geocentric (ECEF).
- Geolocation to Geocentric (ECEF).
- Geocentric to Geolocation.

3.5.2 Tasks

Show the process of representing a vector in the above coordinate frames.

3.5.2.1 Compass to ANS Base

A vector in the ANS Base coordinate frame (Forward, Right, Under) is given by:

$$\overrightarrow{V_{13}} = T_{12}^{13} \times T_{11}^{12} \times T_{10}^{11} \times \overrightarrow{V_{10}} = T_{10}^{13} \times \overrightarrow{V_{10}}$$

Notice that a vector in the Compass coordinate frame (North, East, Down) is given by:

$$\overrightarrow{V_{10}} = \begin{bmatrix} x_{10} \\ y_{10} \\ z_{10} \end{bmatrix}$$

The transformation matrix from the Compass coordinate frame to the ANS Base coordinate frame is given by:

$$T_{10}^{13} = T_{12}^{13} \times T_{11}^{12} \times T_{10}^{11} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & \sin(\varphi) \\ 0 & -\sin(\varphi) & \cos(\varphi) \end{bmatrix} \times \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \times \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ -\sin(\psi)\cos(\varphi)+\sin(\varphi)\sin(\theta)\cos(\psi) & \sin(\psi)\sin(\theta)\sin(\varphi)+\cos(\varphi)\cos(\psi) & \sin(\varphi)\cos(\theta) \\ \sin(\varphi)\sin(\psi)+\cos(\varphi)\sin(\theta)\cos(\psi) & -\sin(\varphi)\cos(\psi)+\cos(\varphi)\sin(\theta)\sin(\psi) & \cos(\varphi)\cos(\theta) \end{bmatrix}$$

3.5.2.2 Compass to Geocentric (ECEF)

$$T_{10}^{3} = T_{4}^{3} \times T_{5}^{4} \times T_{10}^{5} = \begin{bmatrix} \cos(\text{long}) & -\sin(\text{long}) & 0 \\ \sin(\text{long}) & \cos(\text{long}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos(\text{lat}) & 0 & -\sin(\text{lat}) \\ 0 & 1 & 0 \\ \sin(\text{lat}) & 0 & \cos(\text{lat}) \end{bmatrix} \times \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -\sin(\text{lat})\cos(\text{long}) & -\sin(\text{long}) & -\cos(\text{long})\cos(\text{lat}) \\ -\sin(\text{long})\sin(\text{lat}) & \cos(\text{long}) & -\sin(\text{long})\cos(\text{lat}) \\ \cos(\text{lat}) & 0 & -\sin(\text{lat}) \end{bmatrix}$$

Where the parameters lat and long represent respectively the geolocation latitude and the longitude of the ANS. Therefore a vector in the ECEF coordinate frame is:

$$\overrightarrow{V_{2'}} = T_{10}^{3} \times \overrightarrow{V_{10}} + \vec{R}_{\text{ANS}} \qquad \text{where } \vec{R}_{\text{ANS}} \text{ is ECEF coordinates of ANS}$$

Note that a vector in ECEF coordinate frame can also be written as:

$$\overrightarrow{V_{2'}} = T_{10}^{3} \times T_{13}^{10} \times \overrightarrow{V_{13}} + \vec{R}_{\text{ANS}}$$

3.5.2.3 Geolocation to Geocentric (ECEF)

Given the spatial ellipsoidal coordinates parameters latitude, longitude and altitude (φ, λ, h) in the Geolocation (Geodeic) coordinate frame, use Helmert's WGS-84 formula to transform these coordinate parameters into rectangular coordinate parameters x, y, z in the Geocentric coordinate frame as follows:

- Inputs Parameters in Geolocation position will be:
    - ♦ latitude in radians (similar to ANS)
    - ♦ longitude in radians (similar to ANS)
    - ♦ altitude in meters (similar to ANS)
    - ♦ Constants:
        - R_E = 6378137.0 meters (equatorial radius), also called semimajor axis.
        - R_P = 6356752.314 meters (polar radius), also called semiminor axis.
    - ♦ Computed constants:
        - $f = (R_E - R_P)/R_E = 0.003352811$ (Earth's flattening coefficient)
        - $e^2 = f(2 - f) = 0.006694380$ (eccentricity of the ellipsoidal)
        - $v = \dfrac{R_E}{\sqrt{1 - e^2 \sin^2(\varphi)}} = \dfrac{6378137}{\sqrt{1 - 0.006694380 \times \sin^2(\varphi)}}$

          v is the length of the normal to the ellipsoid, from the surface of the ellipsoid to its intersection with the ECEF z axis.
- Output parameters in Geocentric coordinate system are in meters and are given by the formulas below:

$$\begin{cases} x = [v + h]\cos(\varphi)\cos(\lambda) \\ y = [v + h]\cos(\varphi)\sin(\lambda) \\ z = [v(1 - e^2) + h]\sin(\varphi) \end{cases}$$

Example: A geodetic target position defined by: $(\lambda, \varphi, h) = (34.0^o, 117^o 20mn, 251.702m)$, has a computed position in the ECEF coordinate frame given by:

$$(x, y, z) = (-2430601.829, -4702442.706, 3546587.345) \text{ meters.}$$

3.5.2.4 Geocentric To Geolocation

We convert the WGS-84 Cartesian parameters x, y, z in the Geocentric coordinate frame into spatial ellipsoidal WGS-84 parameters in the Geolocation reference frame using the inverse of Helmerts's formulas.

- Inputs parameters (3×1 ECEF Geocentric vector)
  - ♦ x in meters (intersects 0° longitude & equatorial plane)
  - ♦ y in meters (intersects 90° longitude & equatorial plane)
  - ♦ z in meters (parallel to Earth Polar Axis)
  - ♦ Constants:
    - R_E = 6378137.0 meters (equatorial radius)
    - f = 0.003352811 = 1 / 298.257223563 (Earth's flattening coefficient)
    - Computed constants:
      - $R_P = R_E (1 – f)$ = 6356752.314 meters (polar radius)
      - $e^2 = f (2 – f)$ = 0.006694380

$$\varphi = \arctan2\left[z, \left(1 - e^2 \frac{v}{v + h}\right)\sqrt{x^2 + y^2} \right]$$
$$\lambda = \arctan2[y, x]$$
$$h = \frac{\sqrt{x^2 + y^2}}{\cos(\varphi)} - v$$

The longitude λ is straight forward computed given x and y.

If $x^2+y^2 < 10000$, position is < 16 microradians from either North or South Pole and so the longitude and the altitude are given by:

$$\varphi = \mathrm{sign}(z)\times\left[\frac{\pi}{2}-\frac{(1-e^2)\sqrt{x^2+y^2}}{|z|}\right]$$

$$h = |z| - R_p$$

Otherwise the longitude and the altitude are computed by iterative formulas given by:

$$\varphi_0 = \arctan2\left[z,\left(1-e^2\right)\sqrt{x^2+y^2}\right]$$

$$v_{i+1} = \frac{R_E}{\sqrt{1-e^2\times\sin^2(\varphi_i)}}$$

$$h_{i+1} = \frac{\sqrt{x^2+y^2}}{\cos(\varphi_i)} - v_{i+1}$$

$$\varphi_{i+1} = \arctan2\left[z,\left(1-e^2\times\frac{v_{i+1}}{v_{i+1}+h_{i+1}}\right)\sqrt{x^2+y^2}\right]$$

Matlab Implementation:

```
%**********************************************************
% Given the parameters x, y and z in the Geocentric coordinate system,        *
% this subroutine computes the equivalent position Latutude, Longitue and     *
% Altitude in the Geodetic coordinate system using the WGS84 parameters      *
%**********************************************************

function geocentric_to_geodetic(tol);
% Inputs parameters in meters
x=1171068.71;
y=4370487.94;
z=4494419.14;

%Given constant numbers in WGS84
```

```
a=6378137.0;   % RE major axis----equatorial radius
b=6356752.314;   % RP minor axis----polar radius
% WGS84 ellipsoid constants:
a = 6378137;
e = 8.1819190842622e-2;
%Computed variables
f=(a-b)/a;    %flattening
e=sqrt(f*(2.0-f));   % eccentricity

%Initial guess
lat0=atan((z/sqrt(x^2+y^2))*1/(1.0-e^2));
long=atan(y/x);   % This value is exact
v0=a/sqrt(1.0-e^2*sin(lat0)*sin(lat0));
h0=(sqrt(x^2+y^2))/cos(lat0)-v0;

% First iteration
v=a/sqrt(1.0-e^2*sin(lat0)*sin(lat0));
h=(sqrt(x^2+y^2))/cos(lat0)-v;
lat=atan((z/sqrt(x^2+y^2))*1/(1.0-v*e^2/(v+h)));

%First iteration errors
error1=abs(lat-lat0);
error2=abs(h-h0);
error=max(error1,error2);

iter = 1;

while ( error > tol )
  lat1=lat;
  h1=h;
  v=a/sqrt(1.0-e^2*sin(lat)*sin(lat));
  h=(sqrt(x^2+y^2))/cos(lat)-v;
  lat=atan((z/sqrt(x^2+y^2))*1/(1.0-v*e^2/(v+h)));
  error1=abs(lat-lat1);
  error2=abs(h-h1);
  error=max(error1,error2);
  iter = iter + 1;
  fprintf(1,'%d   %5.10f   %5.10f   %5.10f\n',iter,error1,error2,error);
end
fprintf(1,'%d    %5.10f    %5.10f     %5.10f\n',iter,lat,long,h);
```

Euler angles can also be calculated by a close form of the iterative formulas above, derived by Paul [5].

$$p = \sqrt{x^2 + y^2}$$

$$E^2 = a^2 - b^2$$

$$F = 54b^2z^2$$

$$G = p^2 + (1-e^2)z^2 - e^2E^2$$

$$c = \frac{e^4Fp^2}{G^3}$$

$$s = \left(1 + c + \sqrt{c^2 + 2c}\right)^{\frac{1}{3}}$$

$$p = \frac{F}{3\left([s + \frac{1}{s} + 1)\right]^2 G^2}$$

$$Q = \sqrt{1 + 2e^4P}$$

$$r_0 = -\frac{Pe^2p}{1+Q} + \sqrt{0.5a^2\left(1 + \frac{1}{Q}\right) - \frac{P(1-e^2)z^2}{Q(1+Q)} - 0.5Pp^2}$$

$$U = \sqrt{(p - e^2r_0)^2 + z^2}$$

$$V = \sqrt{(p - e^2r_0)^2 + z^2 + (1-e^2)z^2}$$

$$z_0 = \frac{b^2z}{aV}$$

$$e' = \frac{a}{b}e = 0.00820944\ WGS - 84$$

$$h = U\left(1 - \frac{b^2}{aV}\right)$$

$$\lambda = \arctan\left(\frac{z + (e')^2z_0}{p}\right)$$

$$\varphi = atan2(y, x)$$

The matlab codes of this close form is given below:

```
% ECEF2LLA - convert earth-centered earth-fixed (ECEF)
%           cartesian coordinates to latitude, longitude,
%           and altitude
% USAGE:
% [lat,lon,alt] = ecef2lla(x,y,z)
% lat = geodetic latitude (radians)
% lon = longitude (radians)
% alt = height above WGS84 ellipsoid (m)
% x = ECEF X-coordinate (m)
% y = ECEF Y-coordinate (m)
% z = ECEF Z-coordinate (m)
% Notes: This function assumes the WGS84 model.
%        Latitude is customary geodetic (not geocentric).
%

        function [lat,lon,alt] = ecef2lla(x,y,z);

        % WGS84 ellipsoid constants:
        a = 6378137;
        e = 8.1819190842622e-2;

        % calculations:
        b   = sqrt(a^2*(1-e^2));
        ep  = sqrt((a^2-b^2)/b^2);
        p   = sqrt(x.^2+y.^2);
        th  = atan2(a*z,b*p);
        lon = atan2(y,x);
        lat = atan2((z+ep^2.*b.*sin(th).^3),(p-e^2.*a.*cos(th).^3));
        N   = a./sqrt(1-e^2.*sin(lat).^2);
        alt = p./cos(lat)-N;

        % return lon in range [0,2*pi)
        %lon = mod(lon,2*pi);
        fprintf(1,'%5.10f    %5.10f    %5.10f\n',lat,lon,alt);
        return
```

Example: The computed ECEF target position can be converted in the geodetic coordinate frame. The iterative method gives: $(\lambda, \varphi, h) = (0.5934119521 rd, 2.0478571082 rd, 251.6973350393 m)$ with a tolerance of 10^{-7} and after five iterations.

The close form gives: $(\lambda, \varphi, h) = (0.5934119521 rd, 2.0478571082 rd, 251.6972585507 m)$, which shows that both results are very close. Converting the angle back to degrees will prove the reverse process, meaning converting from ECEF to Geodetic.

3.5.2.5 NED Position Vector

We compute the NED (North, East, Down) vector from the ANS location to the target location.

- Inputs parameters
  - ♦ Target position
    - lat_t = latitude of target in radians
    - $long_t$ = longitude of target in radians
    - alt_t = altitude of target in meters
  - ♦ ANS position
    - lat_a = latitude of ANS in radians
    - $long_a$ = longitude of ANS in radians
    - alt_a = altitude of ANS in meters
- Output parameters:

$$rc_t = \frac{R_E}{\sqrt{1 - e^2 \times \sin^2(lat_t)}}$$

$$rc_a = \frac{R_E}{\sqrt{1 - e^2 \times \sin^2(lat_a)}}$$

$$\vec{v}_5 = \begin{bmatrix} x_5 \\ y_5 \\ z_5 \end{bmatrix} = \begin{bmatrix} (rc_t + alt_t) \times \cos(lat_t) \times \cos(long_t - long_a) - (rc_a + alt_a) \times \cos(lat_a) \\ (rc_t + alt_t) \times \cos(lat_t) \times \sin(long_t - long_a) \\ \{(1 - e^2) \times rc_t + alt_t\} \times \sin(lat_t) - \{(1 - e^2) \times rc_a + alt_a\} \times \sin(lat_a) \end{bmatrix}$$

$$\vec{v}_{10} = \begin{bmatrix} x_{10} \\ y_{10} \\ z_{10} \end{bmatrix} = \begin{bmatrix} -x_5 \times \sin(lat_a) + z_5 \times \cos(lat_a) \\ y_5 \\ -x_5 \times \cos(lat_a) - z_5 \times \sin(lat_a) \end{bmatrix}$$

The output vector V_{10} in NED has its components x_{10}, y_{10}, z_{10} in meters.

3.5.2.6 Slant Range To Target

Compute the distance between the ANS and target location.

- Inputs parameters:

- ♦ Target position:
    - lat_t = latitude of target in radians.
    - $long_t$ = longitude of target in radians.
    - alt_t = altitude of target in meters.
- ♦ ANS position (or 2nd target position):
    - lat_a = latitude of ANS in radians.
    - $long_a$ = longitude of ANS in radians.
    - alt_a = altitude of ANS in meters.

- Output parameters:

Use ‘Geolocation position to Geocentric position’ function to convert the ANS and the target positions to get the following:

$V_a = (x_a, y_a, z_a)$ for ANS and $V_t = (x_t, y_t, z_t)$ for the target. The slant range is then the distance between the two points given by:

$$\text{Slant range} = \text{sqrt}[(x_a-x_t)^2 + (y_a-y_t)^2 + (z_a-z_t)^2]$$

3.5.2.7 Compute Euler Angles From The DCM

Any rotation matrix can be constructed as the product of at most three simple rotation matrices. The three angles in these rotations are referred to as Euler angles given as:

Ψ is called the azimuth, heading or yaw angle. It is the rotation about the z axis.

3.5.2.6 NED Position Vector

Compute the North East Down vector from the ANS position to the target position.

- Inputs parameters:
    - ♦ Target position:

- lat_t = latitude of target in radians.
- $long_t$ = longitude of target in radians.
- alt_t = altitude of target in meters.

♦ ANS position:

- lat_a = latitude of ANS in radians.
- $long_a$ = longitude of ANS in radians.
- alt_a = altitude of ANS in meters.

- Output parameters:

A vector V_{10} in node 10 given by:

$$rc_t = \frac{R_E}{\sqrt{1-e^2\times\sin^2(lat_t)}}$$

$$rc_a = \frac{R_E}{\sqrt{1-e^2\times\sin^2(lat_a)}}$$

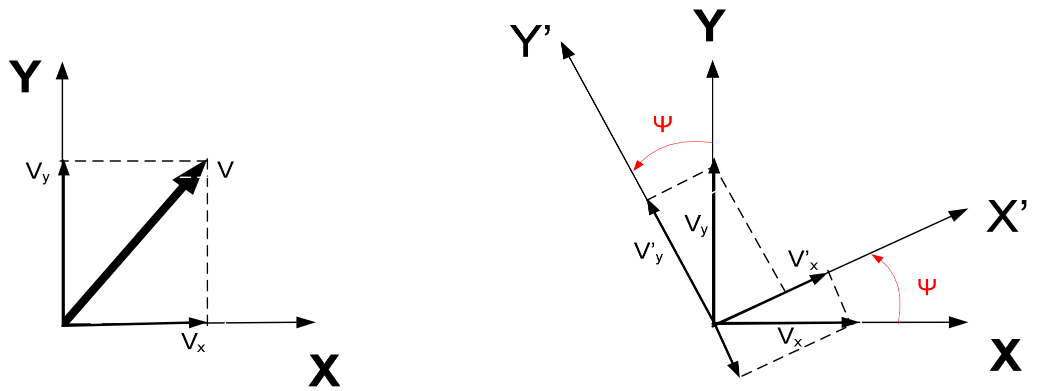
$$\vec{v}_5 = \begin{bmatrix} x_5 \\ y_5 \\ z_5 \end{bmatrix} = \begin{bmatrix} (rc_t + alt_t)\times\cos(lat_t)\times\cos(long_t - long_a) - (rc_a + alt_a)\times\cos(lat_a) \\ (rc_t + alt_t)\times\cos(lat_t)\times\sin(long_t - long_a) \\ \{(1-e^2)\times rc_t + alt_t\}\times\sin(lat_t) - \{(1-e^2)\times rc_a + alt_a\}\times\sin(lat_a) \end{bmatrix}$$

$$\vec{v}_{10} = \begin{bmatrix} x_{10} \\ y_{10} \\ z_{10} \end{bmatrix} = \begin{bmatrix} -x_5\times\sin(lat_a) + z_5\times\cos(lat_a) \\ y_5 \\ -x_5\times\cos(lat_a) - z_5\times\sin(lat_a) \end{bmatrix}$$

3.5.2.7 Compute Euler Angles From The DCM

Any rotation matrix can be constructed as the product of at most three simple rotation matrices. The three angles in these rotations are referred to as Euler angles given as:

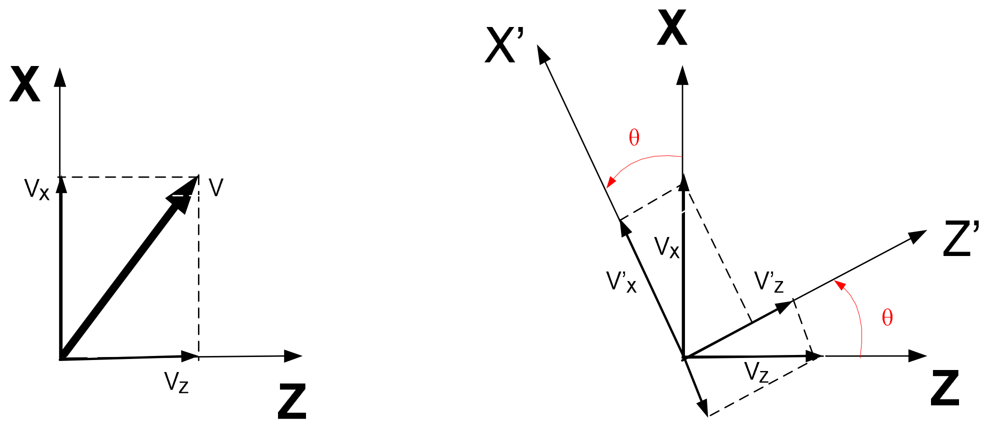
- is called the azimuth, heading or yaw angle. It is the rotation about the z axis.



$$\begin{pmatrix} V'_x \\ V'_y \\ V'_z \end{pmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} = [\mathrm{Yaw}(\psi)] \times \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix}$$

Figure 16. Yaw Euler Angle Rotation.

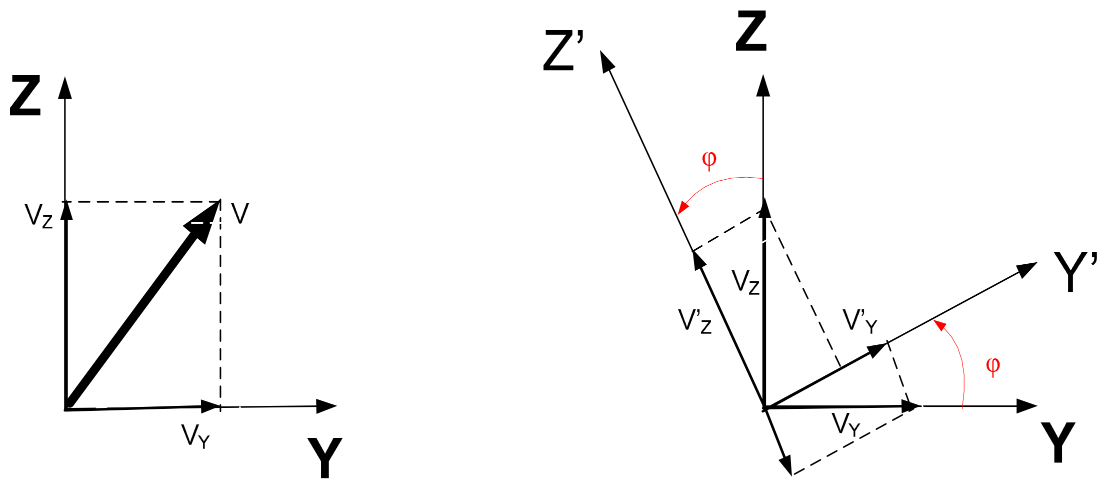
- θ is called elevation or pitch angle. It is the rotation about the y axis.



$$\begin{pmatrix} V'_x \\ V'_y \\ V'_z \end{pmatrix} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \times \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} = [\mathrm{Pitch}(\theta)] \times \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix}$$

Figure 17. Pitch Euler Angle Rotation.

- φ is the roll angle. It is the rotation about the x axis.



$$\begin{pmatrix} V_x' \\ V_y' \\ V_z' \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & \sin(\varphi) \\ 0 & -\sin(\varphi) & \cos(\varphi) \end{bmatrix} \times \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} = [\mathrm{Roll}(\varphi)] \times \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix}$$

Figure 18. Roll Euler Angle Rotation.

The Direction Cosine Matrix (DCM) is the transformation matrix from node 10 to node 13 (see figure 14). This matrix is obtained by a series of rotations first about +Z, second about +Y, third about +X and is given by:

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \cos(\theta)\cos(\psi) & \cos(\theta)\sin(\psi) & -\sin(\theta) \\ -\sin(\psi)\cos(\varphi)+\sin(\varphi)\sin(\theta)\cos(\psi) & \sin(\psi)\sin(\theta)\sin(\varphi)+\cos(\varphi)\cos(\psi) & \sin(\varphi)\cos(\theta) \\ \sin(\varphi)\sin(\psi)+\cos(\varphi)\sin(\theta)\cos(\psi) & -\sin(\varphi)\cos(\psi)+\cos(\varphi)\sin(\theta)\sin(\psi) & \cos(\varphi)\cos(\theta) \end{bmatrix}$$

The rotation angles ψ, θ, φ are heading, pitch and roll.

The corresponding Euler angles are given by:

$$\psi = \text{atan2}(a_{01}, a_{00}) \; ; \qquad -\pi \le \psi \prec \pi$$
$$\theta = \text{atan2}(-a_{02}, \sqrt{a_{00}^2 + a_{01}^2} \; ; \qquad -\frac{\pi}{2} \le \theta \le \frac{\pi}{2}$$
$$\varphi = \text{atan2}(a_{12}, a_{22}) \; ; \qquad -\pi \le \varphi \prec \pi$$

The Euler angles are in radians. One useful application of the Euler angles is that instead of specifying the attitude of one coordinate system relative to another by using the 9 direction cosine angles, a transformation can be represented by the product of three successive transformations about the coordinate's axis. For example, in artillery nomenclature, the barrel of the gun can be oriented to point anywhere in the sky by specifying azimuth (yaw) and elevation (pitch). The third transformation (rolling about the barrel axis) is not required.

3.5.2.8 Converting Pixel Coordinates To Sensor Coordinates

- N_x = number of horizontal x-pixels
- N_y = number of vertical y-pixels
- x_p = x-pixel position for WMI cursor
- y_p = y-pixel position for WMI cursor
- x_c = (Nx-1)/2 = center x-pixel
- y_c = (Ny-1)/2 = center y-pixel
- FOVy = the Field Of View in the vertical direction
    - ♦ Wide FOV
    - ♦ Narrow FOV

- ♦ Ultra-Narrow FOV

- FOVx = the Field Of View in the horizontal direction. It is usually a function of FOVy.
- The direction of the position vector pointing from the center of the Sensor to the position of the WMI. cursor in the Sensor coordinate frame is given by:

$$\vec{v}_{LOS} = \begin{bmatrix} V_{LOS}(x) \\ V_{LOS}(y) \\ V_{LOS}(z) \end{bmatrix} = \begin{bmatrix} 1.0 \\ \dfrac{(x_p - x_c)}{(N_x/2)} \times \tan(FOV_x/2) \\ \dfrac{(y_p - y_c)}{(N_y/2)} \times \tan(FOV_y/2) \end{bmatrix}$$

3.5.2.9 Converting From Azimuth, Elevation, Range To NED

Given the azimuth, elevation and range of a point (az, el, r) from the EOIR coordinate system, one can find the corresponding point (x, y, z) in the NED coordinate using the following formulas:

$$\begin{cases} x = r \times \cos(az) \times \cos(el) \\ y = r \times \sin(az) \times \cos(el) \\ \quad -r \times \sin(el) \end{cases}$$

A matlab codes to implement the above transformation is given below:

```
function ned = convertAzElRange2NED(az, el, r)
%   Float64 azElRange[X_Y_Z_COORDINATE],    // in
%   Float64 ned[X_Y_Z_COORDINATE])          // out
%--------------------------------------------------------------------
% This function converts from Az, El, Range coordinates to  NED
% coordinates.
%
```

```
% ned
%    North, East, Down coordinates in meters
%
% azElRange
%    Azimuth, elevation in radians and range in meters.
%--------------------------------------------------------------------
  % az = azimuth radians
  % el = elevation radians
  % r = range meters
   ned = zeros(3,1);
   ned(1) = r * cos(az) * cos(el);  % north meters
   ned(2) = r * sin(az) * cos(el);  % east meters
   ned(3) = -r * sin(el);           % down meters
```

3.5.2.10 Converting from Latitude, Longitude, Altitude To Azimuth, Elevation And Range

Converting a target position (olat, olon, oalt) to azimuth, elevation and range takes into account the position of the ownship and a matlab code to implement this conversion process is given below.

```
function AzElRange = convertLtLgAl2AzElRange(olat, olon, oalt, elat, elon, ealt)

% in oLtLgAl[X_Y_Z_COORDINATE], tLtLgAl[X_Y_Z_COORDINATE]
% out ned[X_Y_Z_COORDINATE])
%--------------------------------------------------------------------
% This function converts latitude, longitude, and altitude
% coordinates to north, east, down coordinates.
%
% ltLgAl
%    Latitude, Longitude, Altitude coordinates
%
% ned
%    North, east, down coordinates.
%
% -------------------------------------------------------------------

   % Initializing the matrices
   % oLtLgAl = zeros(3,1);
   % tLtLgAl = zeros(3,1);
   ned = zeros(3,1);
   % ellipsoidal model constants for WGS-84
   RE = 6378137.0; % equatorial radius (meters)
   F = 1.0 / 298.257223563; % flattening
   ESQUARED = (2.0 - F)*F; % eccentricity squared
   ONEMINUSESQUARED = 1.0 - ESQUARED; % 1.0 - e^2
```

```
% Float dlon; delta longitude (radians)
% cosolat, sinolat, coselat, sinelat;  cosines & sines of latitudes
% orc, erc; origin & endpoint longitudinal radius of curvature (meters)
% x component parallel to a vector from Earth center to equator at olon
% z component parallel to a vector from Earth center to equator at
% olon+pi/2
% dlon; delta longitude

% olat = oLtLgAl(0); % geodetic latitude of the ownship of the NED
% coordinate set (radians)
% olon = oLtLgAl(1); % geodetic longitude of the ownship of the NED
% coordinate set (radians)
% oalt = oLtLgAl(2); % geodetic altitude of the ownship of the NED
% coordinate set (meters)
% elat = tLtLgAl(0); % geodetic latitude of the target of the NED vector
% (radians)
% elon = tLtLgAl(1); % geodetic longitude of the target of the NED vector
% (radians)
% ealt = tLtLgAl(2); % geodetic altitude of the target of the NED vector
% (meters)

% Compute delta longitude
  dlon = elon - olon;

% Compute cosines & sines of latitudes & longitudes of origin & endpoint
  cosolat = cos(olat);
  sinolat = sin(olat);
  coselat = cos(elat);
  sinelat = sin(elat);

% Compute the latitudinal radius of curvature for the origin and endpoint
  orc = RE / sqrt(1.0 - ESQUARED * sinolat * sinolat);
  erc = RE / sqrt(1.0 - ESQUARED * sinelat * sinelat);

% Compute the component that is parallel to a line from the earth's
% origin to the equator at the longitude of the NED vector's origin
  x = (erc + ealt) * coselat * cos(dlon) - (orc + oalt) * cosolat;

% Compute the east component of the NED vector
  ned(2) = (erc + ealt) * coselat * sin(dlon);

% Compute the component that is parallel to a line from the earth's
% center to the north pole, with z=0 at the (olat, olon, oalt) location
  z = ((ONEMINUSESQUARED) * erc + ealt) * sinelat - ((ONEMINUSESQUARED) *
  orc + oalt) * sinolat;

% Transform x and z to produce the north and down components
  ned(1) = cosolat * z - sinolat * x;
  ned(3) = -sinolat * z - cosolat * x;

% Transform ned back to Az, El, Al
```

```
AzElRange = zeros(3,1);
AzElRange = convertNED2AzElRange(ned(1), ned(2), ned(3));
```

3.5.2.11 Converting From Latitude, Longitude, Altitude To NED

```
function ned = convertLtLgAl2NED(olat, olon, oalt, elat, elon, ealt)

% in oLtLgAl[X_Y_Z_COORDINATE], tLtLgAl[X_Y_Z_COORDINATE]
% out ned[X_Y_Z_COORDINATE])
%--------------------------------------------------------------------
% This function converts latitude, longitude, and altitude
% coordinates to north, east, down coordinates.
%
% ltLgAl
%    Latitude, Longitude, Altitude coordinates
%
% ned
%    North, east, down coordinates.
%
% --------------------------------------------------------------------
 % Initializing the matrices
  % oLtLgAl = zeros(3,1);
  % tLtLgAl = zeros(3,1);
   ned = zeros(3,1);
    % ellipsoidal model constants for WGS-84
   RE = 6378137.0; % equatorial radius (meters)
   F = 1.0 / 298.257223563; % flattening
   ESQUARED = (2.0 - F)*F; % eccentricity squared
   ONEMINUSESQUARED = 1.0 - ESQUARED; % 1.0 - e^2
  % Float dlon; delta longitude (radians)
  % cosolat, sinolat, coselat, sinelat;  cosines & sines of latitudes
  % orc, erc; origin & endpoint longitudinal radius of curvature (meters)
  % x component parallel to a vector from Earth center to equator at olon
  % z component parallel to a vector from Earth center to equator at
  % olon+pi/2
  % dlon; delta longitude
   % olat = oLtLgAl(0); % geodetic latitude of the ownship of the NED
  % coordinate set (radians)
  % olon = oLtLgAl(1); % geodetic longitude of the ownship of the NED
  % coordinate set (radians)
  % oalt = oLtLgAl(2); % geodetic altitude of the ownship of the NED
  % coordinate set (meters)
  % elat = tLtLgAl(0); % geodetic latitude of the target of the NED vector
  % (radians)
  % elon = tLtLgAl(1); % geodetic longitude of the target of the NED vector
  % (radians)
  % ealt = tLtLgAl(2); % geodetic altitude of the target of the NED vector
  % (meters)
   % Compute delta longitude
    dlon = elon - olon;
   % Compute cosines & sines of latitudes & longitudes of origin & endpoint
    cosolat = cos(olat);
    sinolat = sin(olat);
    coselat = cos(elat);
    sinelat = sin(elat);

  % Compute the latitudinal radius of curvature for the origin and endpoint
    orc = RE / sqrt(1.0 - ESQUARED * sinolat * sinolat);
    erc = RE / sqrt(1.0 - ESQUARED * sinelat * sinelat);
   % Compute the component that is parallel to a line from the earth's
  % origin to the equator at the longitude of the NED vector's origin
    x = (erc + ealt) * coselat * cos(dlon) - (orc + oalt) * cosolat;
   % Compute the east component of the NED vector
    ned(2) = (erc + ealt) * coselat * sin(dlon);
   % Compute the component that is parallel to a line from the earth's
  % center to the north pole, with z=0 at the (olat, olon, oalt) location
    z = ((ONEMINUSESQUARED) * erc + ealt) * sinelat - ((ONEMINUSESQUARED) *
    orc + oalt) * sinolat;
   % Transform x and z to produce the north and down components
    ned(1) = cosolat * z - sinolat * x;
    ned(3) = -sinolat * z - cosolat * x;
%************************************************************************
% Given the Latitude, Longitude and Altitude in the Geodetic coordinate   *
% system, this subroutine computes the equivalent position x, y, z        *
% in the Geocentric coordinate system using the Helmert's formula         *
%************************************************************************
function geodetic_to_geocentric;
% Prompt for the inputs
 lat0=input('Please enter the latitude in degree:');
 long0=input('Please enter the longitude in degree:');
 h=input('Please enter the altitude in meters:');
 % Conversion from degree into radians
 lat=(pi/180.0)*lat0;
 long=(pi/180.0)*long0;
 % Given constant numbers
 a=6378137.0;   % major axis----equatorial radius
 b=6356752.314;   % minor axis----polar radius
 % Computed variables
 f=(a-b)/a;    %flattening
 e=sqrt(f*(2.0-f));   % eccentricity
 v=a/sqrt(1.0-e^2*sin(lat)*sin(lat));
 x=(v+h)*cos(lat)*cos(long);
 y=(v+h)*cos(lat)*sin(long);
 z=(v*(1-e^2)+h)*sin(lat);
 fprintf(1,'%5.10f    %5.10f    %5.10f\n',x,y,z);
```

```
%*************************************************************************
% This SSC Library function computes the NED vector from the ANS location *
% to the target location.                                                  *
%*************************************************************************
  function w=NED_Position(lat_t1,long_t1,alt_t,lat_a1,long_a1,alt_a);
 % Inputs: lat, long in degree and alt in meters
% Target position(lat_t, long_t, alt_t)
% ANS position(lat_a, long_a, alt_a)
% Output
% w = 3x1 NED vector
 lat_t=(pi/180.0)*lat_t1;
 long_t=(pi/180.0)*long_t1;
 lat_a=(pi/180.0)*lat_a1;
 long_a=(pi/180.0)*long_a1;
  re=6378137.0;
 e=0.006694380;
  rc_t=re/sqrt(1.0-e*e*sin(lat_t)*sin(lat_t));
 rc_a=re/sqrt(1.0-e*e*sin(lat_a)*sin(lat_a));
 % Coordinates of the corresponding vector in node 10
 x_10=(rc_t+alt_t)*cos(lat_t)*cos(long_t-long_a)-(rc_a+alt_a)*cos(lat_a);
 y_10=(rc_t+alt_t)*cos(lat_t)*sin(long_t-long_a);
 z_10=((1.0-e*e)*rc_t+alt_t)*sin(lat_t)-((1.0-e*e)*rc_a+alt_a)*sin(lat_a);
% w is the output 3x1 vector
 w(1,1)=-x_10*sin(lat_a)+z_10*cos(lat_a);
 w(2,1)=y_10;
 w(3,1)=-x_10*cos(lat_a)-z_10*sin(lat_a);
 return
```

3.5.2.12 Computing Distance In ECEF

A matlab function to compute the distance between two points in the ECEF coordinate frame is given by:

```
  function distance = NEDBSE_Distance(neda1, neda2, neda3, nedb1, nedb2,
                      nedb3)

  distance = sqrt((nedb1-neda1)^2 + (nedb2-neda2)^2 + (nedb3-neda3)^2);
```

CHAPTER IV
SUMMARY AND CONCLUSIONS

A natural question is the need of all these coordinate systems. Why not simplifying life with only one universal coordinate system so that everyone gets used to it? The answer is that nowadays several sensors are available and each one performing different task. Therefore each sensor provides measurements with respect to a given coordinate system based on the available data so that the obtained information or measurement is meaningful to the user. Furthermore things make different amount of sense in different coordinate systems and in different coordinate spaces. Notice that a coordinate system is the frame of reference and a coordinate space is the values that are used in the coordinate system. Every object has its own coordinate space referred to as its local space and different coordinate spaces can be used to represent the same point in different ways.

Coordinate systems are extensively used in geometry and kinematics, not only to describe the position of points, but also to describe the angular position of axes, planes, and rigid bodies.

For example the ANS system provides estimates of antenna position in the ECEF coordinate system, vision and radar provide distance measures in a local vehicle-relative coordinate system, and accelerometers and gyros provide inertial measurements expressed relative to the instrument axes. Given that different sensors provide measurements relative to different frames, the measurements in different frames are comparable only if there are convenient means to transfer the measurements between the coordinate systems. For example if a target is moving north in the sensor coordinate frame, then there is no way to indicate its position in a global/world frame unless some coordinate transformations are used, because north in the sensor coordinate frame can mean south, east, west, or anything in another coordinate frame. A GPS/INS performing navigation relative to a fixed tangent-plane frame of reference will typically be:

- Transform acceleration and angular rate measurements to platform coordinates.
- Compensate the platform angular rate measurements for navigation-frame rotation.

- Integrate the compensated platform-frame angular rates to maintain an accurate vector transformation from platform to navigation coordinates.
- Transform platform-frame acceleration to tangent plane by using the angular rates.
- Integrate the compensated tangent-plane accelerations to calculate tangent-plane velocity and position.
- Make GPS measurements, transformed from ECEF to tangent-plane coordinate, to estimate and correct errors in the sensed and the calculated INS quantities.

A coordinate transformation being a set of steps (conversion) of the position of a target form one coordinate frame to another one does not modify the position of that target. Instead the transformation describes the same space such that the new coordinates of the image of the target are the same as the old coordinates of the initial or original target.

REFERENCES

[1] Peter H. Dana, Department of Geography, University of Texas at Austin, 1995.
[2] Wikipedia, the free encyclopedia.
[3] Raytheon Systems Engineering.
[4] Raytheon Systems Architecture.
[5] Ward, H.R. 1973. Calculation of the factor k_m used in angle error equations.
Memo HRW-250, Raytheon Co., Equipment Division,
Wayland, MA, May 7, 1973.
[6] Laurila, Simo H. 1976. Electronic Surveying and Navigation. New York: John Wiley & Sons.
[7] Definition and Relationships with Local Geodetic Systems, 3rd Edition. Washington, DC: National Imagery and Mapping Agency.
[8] Yeddanapudi, M., Bar-Shalom, Y., and Pattipati, K.R., 1997
IMM estimation for multitarget-multisensor air traffic surveillance.
IEEE Proceedings, 85, 1 (Jan. 1997), 80-94.
[9] Olson, D. K., 1998
Calculation of geodetic coordinates from Earth-centered, Earth-fixed coordinates.
Journal of Guidance, Control and Dynamics, 11,2 (Mar.-Apr. 1998), 188-190.
[10] Gersten, R. 11. 1961
Geodetic sub-latitude and altitude of a space vehicle.
Journal of the Astronautical Sciences, 3, 1 (Spring 1961), 28-29
[11] Zhu, J. 1994
Conversion of Earth-centered, Earth-fixed coordinates to geodetic coordinates
IEEE Transactions on Aerospace and Electronic Systems, 30, 3 (July 1994), 957-962
[12] Ewing, C. E., and Michael, M. M., 1970
Introduction to Geodesy
New York: American 1:Isevier, 1970.
[13] Karl T. Ulrich, Steven D. Eppinger, 2004
Product Design And Development, third edition.
[14] Sitharama Lyengar and Richard R. Brooks, 2005
Distributed Sensor Networks.

APPENDIX A
BASIC MATHEMATICS

Length of a vector

Given a vector V = (x, y, z) its length or magnitude ||V|| is given by the following formula:

$$\| \mathrm{V} \| = \sqrt{\mathrm{x}^2 + \mathrm{y}^2 + \mathrm{z}^2}$$

Dot product

The dot product of two vectors $V_1 = (x_1, y_1, z_1)$ and $V_2 = (x_2, y_2, z_2)$ is a scalar d given by:

$$d = V_1 \bullet V_2 = (x_1 \bullet x_2) + (y_1 \bullet y_2) + (z_1 \bullet z_2)$$

It also can be given by:

$$d = V_1 \bullet V_2 = \| V_1 \| \bullet \| V_2 \| \bullet \cos(\theta)$$

where θ is the angle between the two vectors V_1 and V_2.

Cross product

The dot product of two vectors $V_1 = (x_1, y_1, z_1)$ and $V_2 = (x_2, y_2, z_2)$ is a vector V given by:

$$V = V_1 \times V_2 = (y_1 \bullet z_2 - y_2 \bullet z_1, x_2 \bullet z_1 - x_1 \bullet z_2, x_1 \bullet y_2 - x_2 \bullet y_1)$$

It also can be given by:

$$V = V_1 \times V_2 = \| V_1 \| \bullet \| V_2 \| \bullet \sin(\theta)$$

where θ is the angle between the two vectors V_1 and V_2.

Rotation matrix about the x-axis

The rotation matrix of an angle θ about the x-axis is given by:

$$\begin{bmatrix} \mathrm{T}(x,\theta) \\ 3\times 3 \end{bmatrix} = [\mathrm{X}(\theta)] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & +\sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

Rotation matrix about the y-axis

The rotation matrix of an angle θ about the y-axis is given by:

$$\begin{bmatrix} \mathrm{T}(y,\theta) \\ 3\times 3 \end{bmatrix} = [\mathrm{Y}(\theta)] = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ +\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

Rotation matrix about the z-axis

The rotation matrix of an angle θ about the z-axis is given by:

$$\begin{bmatrix} \mathrm{T}(z,\theta) \\ 3\times 3 \end{bmatrix} = [\mathrm{Z}(\theta)] = \begin{bmatrix} \cos(\theta) & +\sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrix × Vector

The product of a 3× 3 matrix A and a vector 3× 1 Vin = (xin, yin, zin) is a 3× 1 vector Vout

given by:

$$A = \begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix}$$

V = (xout, yout, zout) where
xout = (a00 × xin) + (a01 × yin) + (a02 × zin)
yout = (a10 × xin) + (a11 × yin) + (a12 × zin)
zout = (a20 × xin) + (a21 × yin) + (a22 × zin)

Matrix × Matrix

The product of a 3× 3 matrix A by another 3× 3 matrix B is a 3× 3 matrix C given below:

$$[A] = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix}$$

$$[B] = \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix}$$

$$[C] = [A] \times [B] = \begin{bmatrix} c_{00} & c_{01} & c_{02} \\ c_{10} & c_{11} & c_{12} \\ c_{20} & c_{21} & c_{22} \end{bmatrix}$$

with

$$\begin{aligned}
c00 &= (a00 \times b00) + (a01 \times b10) + (a02 \times b20) \\
c01 &= (a00 \times b01) + (a01 \times b11) + (a02 \times b21) \\
c02 &= (a00 \times b02) + (a01 \times b12) + (a02 \times b22) \\
c10 &= (a10 \times b00) + (a11 \times b10) + (a12 \times b20) \\
c11 &= (a10 \times b01) + (a11 \times b11) + (a12 \times b21) \\
c12 &= (a10 \times b02) + (a11 \times b12) + (a12 \times b22) \\
c20 &= (a20 \times b00) + (a21 \times b10) + (a22 \times b20) \\
c21 &= (a20 \times b01) + (a21 \times b11) + (a22 \times b21) \\
c22 &= (a20 \times b02) + (a21 \times b12) + (a22 \times b22)
\end{aligned}$$

Matrix Inverse

The inverse of a 3× 3 matrix A is a 3× 3 matrix B given by the formulas below:

$$[A] = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix}$$

det = a00 (a11 a22 – a21 a12) + a01 (a12 a20 – a10 a22) + a02 (a10 a21 – a11 a20)
b00 = (a11 a22 – a21 a12) / det
b10 = (a12 a20 – a10 a22) / det
b20 = (a10 a21 – a11 a20) / det
b01 = (a02 a21 – a01 a22) / det
b11 = (a00 a22 – a02 a20) / det
b21 = (a01 a20 – a00 a21) / det
b02 = (a01 a12 – a02 a11) / det
b12 = (a02 a10 – a00 a12) / det
b22 = (a00 a11 – a01 a10) / det

With the matrix B given below:

$$[B] = \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix}$$

if the determinant is not zero. Otherwise the matrix A is called a singular matrix.