

Optical Navigation System Simulation Guide V1.0

Princeton Satellite Systems 6 Market Street, Suite 926 Plainsboro, NJ 08536 (609) 275-9606

Contents

1	Sum	mary	4
2	Sim	ulation Components	4
3	Buil	ding from an Installer	5
4	Buil	ding from SVN	5
5	Arch	nitecture of the Simulation	7
6	API	S	8
7	Dem	onstration	8
	7.1	Introduction	8
	7.2	Simulation Setup File	14
	7.3	Control Deck	14
	7.4	VCI File	16
	7.5	User Operations	16
	7.6	Editing	16
8	DSir	nManager	19
9	Sim	ulation Models	19
	9.1	Spacecraft Dynamics	19
	9.2	Disturbance Model	22
	9.3	Gravity Model	24
	9.4	Camera Model	25
	9.5	IMU Model	26
	9.6	Ranging Model	26
	9.7	State Sensor Model	27
	9.8	GPS Model	27
	9.9	Intersatellite Link Model	28
	9.10	Radar Model	28
	9.11	Propulsion Model	29
	9.12	Gimbal Model	29

List of Figures

1-1	Mercury Messenger flyby 9/21/2009	4
2-1	Icons	5
3-1	Installer	5
3-2	ONS folder	6
4-1	ONS project	7
4-2	Building frameworks in Terminal	8
5-1	User interaction	9
6-1	SCControl API index page	10
7-1	VisualCommander simulation start	11
7-2	Simulation summary page	12
7-3	Orbit determination with Pluto as a target	12
7-4	Attitude control maneuver	13
7-5	Simulation setup file in DSimManager	14
7-6	Data window in VisualCommander	17
7-7	Data flow window in VisualCommander	17
7-8	Editing VisualCommander	18
8-1	DSimManager	19
8-2	DSimManager: Components window	20
8-3	DSimManager: select integrator	21
8-4	DSimManager: Drag and drop models	21
9-1	GPS model constellation	27
9-2	Radar frame of reference	28

1 Summary

This document gives a brief introduction to running the Optical Navigation System simulation in Visual-Commander. It provides two test cases that can be used as a template for developing additional simulations:

- 1. New Horizons Pluto flyby
 - (a) A spacecraft similar to NewHorizons with ONS with a thruster control system
 - (b) The spacecraft is within 1 day of Pluto encounter
 - (c) The ephemeris includes the Earth, Moon, Mars, Jupiter, Saturn, Uranus, Neptune and Pluto
- 2. Messenger Mercury flyby Figure 1-1
 - (a) A spacecraft similar to Messenger. The simulation is of the September 2009 flyby of Mercury.
 - (b) The simulation starts a before the flyby
 - (c) The ephemeris includes all of the planets out to Saturn

This document explains how to build the simulation and control software from subversion. That section can be skipped if you are building it from an installer. Also provided with this Users Guide is an HTML version of the API documentation for all of the software. VisualCommander has a built-in help system accessible from the help menu.

Figure 1-1. Mercury Messenger flyby 9/21/2009



2 Simulation Components

Table 2-1 on the next page lists the software components provided with this package. Running the complete simulation requires a computer running Mac OS 10.6+.

The icons for VisualCommander and DSimManager are shown in Figure 2-1 on the following page.

Table 2-1. Simulation components

Component	Description
VisualCommander v3.0	The client/server application for running simulations, interacting with the
	simulation and visualizing the simulation. The client portion of VC 3.0
	requires Mac OS 10.6+.
DSim	The cross-platform simulation engine
Spacecraft Control Library (SCControl)	A library of C/C++ classes and functions for many common spacecraft
	simulations
Spacecraft Package	A library of C++ classes for DSim spacecraft simulations
DSimManager v1.0	An application for building DSim simulations
ControlDeck v2.0	A C++ library for building real-time control systems
DCS	An Xcode project that contains all ControlDeck software that imple-
	ments the system
APIs	Doxygen APIs are provided for ONS, the Spacecraft package and SC-
	Control

Figure 2-1. Icons



3 Building from an Installer

The installer is shown in Figure 3-1. Double click on it and it will install VisualCommander. If you see VisualCommander in the toolbar it is preinstalled.

Figure 3-1. Installer



Some components need to be copied onto the computer. Those are shown in Figure 3-2 on the following page. These include the Xcode projects and the APIs.

4 Building from SVN

If you have access the the Princeton Satellite Systems Subversion (svn) archive you can build your simula-

Figure 3-2. ONS folder

00	ONS CONS	\bigcirc
		_
	ASCONTROL	
Name	Date Modified	
	Yesterday 1:59 PM	
CSpice	Yesterday, 2:08 PM	
► ONS	Yesterday, 1:58 PM	
ons_api	Yesterday, 2:10 PM	
sccontrol_api	Yesterday, 1:10 PM	
Spacecraft	Yesterday, 1:59 PM	
spacecraft_api	Yesterday, 1:48 PM	
1 of 7	selected, 243, 15 GB available	
1017	Sector, 215125 do avanable	

tion and control deck from svn. You need to build the following:

- 1. Frameworks frameworks consists of many different Xcode projects. You will not normally need to look at any of these except for ASControl which includes many C++ functions related to dynamics and control
- 2. Spacecraft this contains all the DSim dynamical models
- 3. ONS Optical Navigation System ControlDeck. These are all the control and navigation functions

All of the packages are found on the PSS svn server. The following commands will download the required software

```
svn co https://svn.psatellite.com/Packages/ONS
svn co https://svn.psatellite.com/Packages/Spacecraft
svn co https://svn.psatellite.com/Packages/CSpice
svn co https://svn.psatellite.com/VisualCommander
svn co https://svn.psatellite.com/Frameworks
```

The ONS Xcode project is shown in Figure 4-1 on the next page. The window shows the major folders in the project. It also shows all the libraries needed to build the ONS software.

You need to first build Frameworks. Building frameworks is done using Terminal. An example session is shown in Figure 4-2 on the following page. The sessions show all of the projects that are built. The first

Figure 4-1. ONS project



command, "make macclean" cleans out all the compiled code. "make macrelease" creates a release version of each package.

To build an Xcode project double click on the project file, xxx.xcodeproj. Under Build select "Clean all Targets". Once this is done select "Build"

5 Architecture of the Simulation

Figure 5-1 on page 9 shows how the user interacts with VisualCommander. The user interacts with the simulation via DSimManager, a self-contained application, and with the ControlDeck, which contains all of the control and navigation software and its interface with the simulation. The sequence of steps is to

- 1. Build a simulation using DSimManager
- 2. Write a ControlDeck. The ControlDeck has a link to the simulation
- 3. Start VC and connect the ControlDeck to a session
- 4. Build your interface interactively in VisualCommander

The file types are

- 1. .vci file contains the user interface
- 2. .txt file contains the control deck

Figure 4-2. Building frameworks in Terminal

\varTheta 🔿 🔿 Terminal — bash — 57×51
Last login: Fri Aug 13 22:29:36 on ttys000
Iapetus:~ mike\$ cd svn/Frameworks/
Iapetus:Frameworks mike\$ svn update
At revision 21544.
Iapetus:Frameworks mike\$ make macclean
Making Clean in AbstractNet
Making Llean in Doutlis
Making Llean in InterLomm
Making Clean in Netotils
Making Clean in MatrixLip
Making Clean in Secondarilbile
Making Clean in GeometryUtils
Making Clean in AScontrol
nuking clean in SimBuilder
Making Clean in DSimEnging?
nuktny clean in Dolmenginez
nuking Clean in ControlDeck
naking ilean in ControlDeck2
Removing Geometry Data
Clean complete, check build_results.txt for build logs.
Ignatus:Ergmaworks miket make macralegee
Delegee
Making Delegge in AbstractNet
Making Release in DSUtils
Making Release in InterComm
Making Release in NetHtils
Making Release in Matrixlib
Making Release in SDManin
Making Release in ASControl
Making Release in GeometryUtils
Making Release in DSimEngine
Making Release in SimBuilder
Making Release in DSimEngine2
Making Release in ControlDeck
Making Release in ControlDeck2
Installing Geometry Data
Make Release complete, check build_results.txt for build
logs.
Iapetus:Frameworks mike\$

3. .ds2 file - an xml file that contains the simulation definition

6 APIs

Figure 6-1 on page 10 shows the index page for the SCControl API. All three APIs (Spacecraft, ONS and SCControl) are in doxygen format. They are provided in 3 folders.

7 Demonstration

7.1 Introduction

This section takes you through a complete simulation of the Optical Navigation System as the New Horizons spacecraft approaches Pluto. The model has the same thruster layout as the actual New Horizons and its

Figure 5-1. User interaction





Figure 6-1. SCControl API index page

A O O	CCControl ADI
the file: ///lisers /mike /nm /5m	sccontrol secontrol API
Apple Amazon Bank Calenda	erwiskyrszennovyszennovyszennovymine and a start w sta
SCControl: SCControl API	
Main Page Classes Files	Q* Search
SCControl API	
	1.0
	Spacecraft Control Library
This application programming interface (API) ephemeris, and performing orbit mechanics i	document contains documentation for functions in SCControl framework. There are modules for coordinate transformation, modeling the space environment, computing scluding planetary perturbations. This library utilizes the broad range of matrix manipulation functions provided in MatrixLib.
The formation flying module, if included, is a	comprehensive set of functions for the guidance and control of formation flying spacecraft.
Using this API	
File List	
All the functions are grouped here.	
Class List	
The major classes and structures are describe	id here.
Support	
If you find a bug, please send a full descripti software engineers will respond to confirm re	on to support@psatellite.com, along with the source code you were using when you found the problem, and a summary of what the source code is supposed to do. One of ceipt of your email, and then investigate the problem.
	No part of this document may be reproduced without permission in writing from Princeton Satellite Systems.
	Generated on Thu Sep 23 2010 15:10:24 for SCControl by

mass properties are similar but there is no attempt to exactly replicate the spacecraft. The situation for the Messenger simulation is similar, differing only in the details of the orbit and objects tracked.

The spacecraft does the following:

- 1. Acquires Pluto and Polaris using the telescopes
- 2. Computes the orbit
- 3. Computes the attitude
- 4. Flies past Pluto

To start the simulation find your VisualCommander application and double click on the application icon. You will need to pull down the file menu and find the file NewHorizons_pluto.ds2. The simulation will begin to run.

The following figures show a typical simulation. The menu in the upper left hand corner allows you to change pages so that you can view other aspects of the simulation. You can also use the arrow keys for this purpose.

Figure 7-1. VisualCommander simulation sta	rt
--	----



After the program starts, ONS automatically begins to align one sensor with Pluto and the other with a polar star. The attitude estimate just uses the single frame solution and is allowed to converge. The progress of the alignment can be viewed on the Summary page in the upper right frame, or on the Tracking page where the attitudes of the cameras can be followed quantitatively.

The program is initialized with the optical sensor input to the orbit determination routine turned off. The estimator runs, but has no optical input. Once the program is running, the optical sensor input can be enabled. On the Orbit Determination page enter a 1 in the od_use_camera_command entry field, and click <Send> to execute the command. With the new input from the camera, the covariance begins to decrease eventually reaching down to about 25 km². The orbit determination is accomplished using only the radius of Pluto and Pluto centroid/star centroid measurements. Figure 7-2 on the next page shows the state vector for the spacecraft (distance vector, velocity vector) on the left hand side. The large window is a 3D view of the spacecraft, planets and stars. The display in the upper right and corner shows the sensor cones. The display below that shows the solar system and position of the planets and spacecraft. "DSimlsimulation:scale" is the ratio of sim speed to real-time. Anywhere from 1 to 10 should work on most computers.

The progress of New Horizons can be monitored from this page as the spacecraft passes Pluto. In the large frame one can observe the spacecraft and Pluto in the same view, and watch as Pluto recedes after the spacecraft passes. In the upper right frame, one can observe the attitude of the sensors as they slew to keep Pluto in the field of view. It will likely be necessary to adjust the view in the frames by dragging with the mouse on the image. There are also controls below the frame to change the position and field of view of the observer.

Figure 7-3 on the following page shows the orbit determination page. When simulation starts, the position and velocity of the spacecraft are initialized, and the state of the spacecraft thereafter is determined by the physical model. The orbit determination estimator is running with access to the model, but input from the camera is turned off. On the orbit determination page one can see the results of the orbit determination routine in the variables labeled "od_state", while in the upper right hand corner the state of the craft as determined by the simulation is shown for comparison. Also shown is the estimator's covariance matrix.

Along the bottom of the frame are displayed several commands that are used to control the orbit determination process. In particular, as mentioned above, the camera input can be turned on to add that data to

Figure 7-2. Simulation summary page

0 0	New Horizons (NewHorizons_Pluto.vci)	
✓ ► Summary S	Run	• N T
ONS New H	izons	
119720614 position: -444391111 -174766498 5. velocity: -12. -4.		
1. quaternion: 0. 0. 0.		
0.0 bodyRate: 0.0 0.0		
	Datance form: Octament Spectrum 0x068 C Visu algit: wide take O Camera target: Nesting-radio Camera cool frame EO Camera pacton Camera pacton Ump tb V	
Princeton	DSim(simulation:scale 2015-07-14 11:15:08:975 Immediate 9/13/2010 11:06:16 PM Data: 5 Value: 5	

the estimator. With the camera enabled, one can observe the covariance diagonal decrease as the estimation improves with the new data. Similarly, a simulated range message from a ground station can be added by enabling "od_usd_range_command". message sent from the simulated ground station.





You can do an attitude maneuver using the Attitude Control page. Figure 7-4 on the next page shows a roll maneuver. We waited until the targets were tracked and attitude control converged. To execute an attitude maneuver, enter the desired final attitude quaternion in the four input fields for

acs_q_eci_to_body_command, and click <Send> to load the command, then enter 1 in acs_control_mode_command followed by <Send> to execute the command. (This last step may not be necessary.) Tracking maintains alignment on its targets even during the maneuver. Graphical display of thruster parameters can be viewed on the Attitude Control page, and the changing attitude of the spacecraft can be followed on the Summary page. Also on the Summary page you can watch the camera sensors realign to keep their targets in view. One advantage of ONS is that if you start with the cameras on a star field, the sensors will stay on the star field during the maneuver. Of course if the maneuver is large enough you must make sure that the tracking targets switch to new targets that can be seen by the sensor.

Figure 7-4. Attitude control maneuver

Attitude Control \$	New Horizons (NewHorizons, Pluto.vcl)	• 1
NS New Horizons		
NewHorizons.acs_control_mode_comm Immediate 9/17/2010 6:03:29 PM Data: Send Send Send Value: 1 NewHorizons.acs_q_eci_to_body_comm Minediate 9/17/2010 6:03:29 PM Data: 0.90 0.10 6:03:29 PM Data: Send 0.00 0.00 Send 0.00 Value: 0.90 0.00 Send 0.00 Value: 0.00 Value: 0.00 Value: 0.90,01:00 Value: 0.90,01:00 Send 0.00 Send 0.00 Send 0.00 Send 0.00 Send 0.00 Send Send 0.00 Send Send	nd ind ind ind ind ind ind ind i	
ster_pulsewidth_demand: 0.00000		
uaternion: 0.99401 0.0025 0.00000 -0.00000 aca_brque_demand: -0.00000 -0.00000 aca_brque_demand: -0.00000 -0.00000 PrincetonSATELLITE	0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1	
Summary ÷	New Horizons (NewHorizons_Pluto.vcl)	• 🔊
1197185112.733 ostion: 444386535.837 -1747845528.183 5.33672 velocity: -12.28672 -4.85380 0.39400 0.0398 0.00000 -0.00000		
-0.00001 0.00000 D.00000 Dataret (Vez agis	nt Carera Sourceth Dupler add Carera Control for the Control	
PrincetonSATELLITE	DSimjsimulation:scale 2015-07-14 10:11:50.996 Immediate 9/17/2010 6:03:29 PM Data: 5 Value: 5	

7.2 Simulation Setup File

DSim is the software component that actually executes the simulation. A simulation is constructed and initialized by means of a XML file, which in the case of the New Horizons simulation is in a file NewHorizons_pluto.ds2. There is a GUI application, DSimManager, that allows the setup file to be constructed, viewed, and modified. Modules that model simulation components can be added to the simulation by drag-and-drop, variables initialized, connected, and tagged for logging, etc. For example, a spacecraft can be added to the simulation simply by dragging it's entry in the Components window into the Object Hierarchy. The file as exposed in DSimManager is shown in Figure 7-5. The Object Hierarchy tab is open showing the spacecraft modeled in the simulation. The main spacecraft hierarchy is expanded showing all of the variables associated with the spacecraft model.

Figure 7-5. Simulation setup file in DSimManager

/			Simulation	Integrat	ors C	Object	Hierarchy	Test		
bject 🔺	Туре	Library -		Object	Variable	s C	outlets	Targets	Setup Networks	Dependencies
NewHorizons	RigidBody	Spacecra	Name	Usees	1.00	тс	Tune	Links	Page Value	Quarrida
Gravity	SpiceGravity	Spacecra	name	Output	LOG		Matrix	Units m/cA2	Base value	Overnde
Ristform	Groundstatic	Spacecra	acceleration	Output			Maurix	111/5/12	[0,0,0]	
* Flatform	Beyel	Spacecra	angular_accel	e Output			Matrix	rad/s^2	[0;0;0]	
▼ OuterGimbal2	Gimbal	Spacecra	angular_mom	e Output] Matrix	kg-m^	[0;0;0]	
▼ InnerGimbal2	Gimbal	Spacecra	bodyRate	Integrated			Matrix	rad/s	[0;0;0]	
IMU	IMUMems	Spacecra	center_mass	Parameter			Matrix	m	[0;0;0]	[0.168259259259259; 0
Imager2	Imager	Spacecra	frame	Parameter			 Integer		0	
▼ OuterGimbal1	Gimbal	Spacecra	inantia	Deserveter			Materia	lin mit 7	1 0 0 0 1 0	1228 58 0 0:0 455 72 0:0
▼InnerGimbal1	Gimbal	Spacecra	mertia	Parameter			Matrix	Kg-mh2	[1, 0, 0,0, 1, 0,	[328.38 0 0;0 433.73 0;0
IMU	IMUMems	Spacecra	mass	Output		□ ₹	Double	kg	1	U
Imager1	Imager	Spacecra	mass_base	Parameter			Double	kg	1	393
Thermal	Thermal	Spacecra	position	Integrated		E E	Matrix	km	[0;0;0]	[1.197181176117546E+0
Propulsion	Propulsion	Spacecra 🏾 î	quaternion	Integrated			Matrix	n/a	[1:0:0:0]	-
Time	TimeReferen	Spacecra			0				(•
			Variable: b	odyRate					Attribute	A Malua
			Lisage: Ir	tegrated					Attribute	Value
			osuge. n	regrated						
			Units: ra	ad/s						
			Type: N	latrix						
			Rase Value: [[0.0.01						
			buse value. [e	,,0,0]						
			Set Value:							
			Description: B	ody rotation	al rates					
									+ -	

7.3 Control Deck

DSim has no control capability. It simply runs the simulation given the data and parameters it has received. DSim gets its data from some external source. One source is the Control Deck. The Control Deck can read data from the simulation, process data, and send commands and data to the simulation. That is, it performs analysis and control functions. The Control Deck contains a list of all of the modules (instantiated C++ classes) in the control system and creates mappings between DSim variables and variables it uses itself. The Control Deck for this demonstration is found in the file ONSControl_pluto.txt. The file is shown below.

Listing 1. Control deck

ONSControl_pluto.txt

1	Simulation /Library/Application	Support/VisualCommander/Model	Libraries/NewHorizons_pluto.ds2
2			
3	System NewHorizons		
4			
5	TimeScale 1		
6			
7	# Time		
8	#		
9	Module NewHorizons ons_timer	Timer	ONS

10 11	ŧ Time		
12 13 14	F Module NewHorizons ons_time	Time	ONS
15	ł Set up		
10	/ Aodule NewHorizons ons_init_nh_pluto	Setup	ONS
18	Adule NewHorizons ons_unused_variables	Unused_Variables	ONS
19	Aodule NewHorizons ons_communications	Communications	ONS
20	t Command processing module. All commands come	here	
22 23	f Module NewHorizons ons_command	Command	ONS
24	F Telemetry module. All telemetry to VC will co	ome from this module	
.6 •7	f Module NewHorizons ons telemetry	Telemetry	ONS
28	loadie Newhorizond ond_coremeery	i ci cine ci y	0110
:9	f Interfaces		
30 31	for the second sec	Image Processing	ONS
32	Adule NewHorizons ons_gimbal_processing	Gimbal_Processing	ONS
3	Adule NewHorizons ons_mems_imu_processing	IMU_Processing	ONS
4	1odule NewHorizons ons_mass	Mass_Processing	ONS
6 7	f Catalogs #		
8	Aodule NewHorizons ons_star_catalog	Star_Catalog	ONS
9	+ IItilitios		
1	+		
2	Aodule NewHorizons ons_ephemeris	Ephemeris	ONS
3	Adule NewHorizons ons_thermal_control	Thermal	ONS
4 5	Module NewHorizons ons_lault_detection	Celestial Object ID	ONS
6			
7 0	f Estimation #		
.9	' Module NewHorizons ons_od	Orbit_Determination	ONS
0	Nodule NewHorizons ons_attitude_determination	Attitude_Determination	ONS
1 2	f Control and Estimation		
4	/odule NewHorizons ons_attitude_control	Attitude_Control	ONS
5	Aodule NewHorizons ons_propulsion	Propulsion	ONS
6	Module NewHorizons ons_targeting	Targeting	ONS
, , 68	Naure NewHorr2015 Ons_gimbar_control	Gimbal_conclut	0103
9	ŧ		
50	f Inputs		
52	/ariable NewHorizons Time:julianDate	NewHorizons jd_sim	
3	Variable NewHorizons:quaternion	NewHorizons quaternion_	sim
4	/ariable NewHorizons:bodyRate	NewHorizons omega_sim	
5	/ariable NewHorizons:position	NewHorizons position_si	m
7	Variable NewHorizons:mass	NewHorizons mass sim	111
8	/ariable NewHorizons:center_mass	NewHorizons center_of_m	ass_s
9	/ariable NewHorizons:inertia	NewHorizons inertia_sim	
/0 71	# Ephemeris		
72			
73	Variable NewHorizons Gravity:observer	NewHorizons observer_si	m
74	Variable NewHorizons Gravity:rotationMatrixBody	NewHorizons rotation_ma	trix_
76 / S	Variable NewHorizons Gravity:planet_index	NewHorizons planet_inde	⊼_S⊥N anet
77	/ariable NewHorizons Gravity:muBody	NewHorizons planet_mu_s	im

78	Variable	NewHorizons Gravity:radiusBody	NewHorizons planet_radius_	_sim	
79					
80	# Ground	Station			
81	#				
				ONSControl	nluto trt

The first line gives the path to the simulation file which has the suffix .ds2. The line with "System" defines the name of a control system, NewHorizons. A simulation may contain a number of control systems. TimeScale gives the ratio of the control period to real-time.

The lines starting with Module attach processing modules to ControlDeck. Each module is an instantiation of a C++ class which has the base class cd_control_module. For example

Module NewHorizons ons_ephemeris Ephemeris ONS

Defines a Module for the control system NewHorizons using the ons_ephemeris class. It is given the name Ephemeris and is from the ONS bundle which is built by ONS.xcodeproj.

The Variable line connects the ControlDeck to the DSim model. For example

Variable NewHorizons|Time:julianDate NewHorizons jd_sim

maps the variable julianDate from the Time module (an instance of the class ons_timer) which is a child of the NewHorizons model to the ControlDeck variable jd_sim in the System NewHorizons. Now any module within the ControlDeck can access julianDate via the Control Deck variable jd_sim with a call to the request_data function.

jd_sim_ref = request_data(NULL, "jd_sim", sd_type_double);

7.4 VCI File

You run the simulation by selecting NewHorizons_Pluto.vci either from VisualCommander or double clicking on the file. The window has a number of pages that can be used to send commands to the simulation and to observe results. You can customize the pages, or add more pages, while the simulation is running by dragging tools and/or data points onto the window. See the VisualCommander online help for more information.

7.5 User Operations

Commands that the user can use during the demonstration are shown in Table 7-1 on the next page. The table shows the pages and the commands available on each page. You can add commands or telemetry to any page. You can see some of the commands and telemetry in Figure 7-6 on the following page.

7.6 Editing

Figure 7-7 on the next page shows the data flow window which can be used to connect simulation data to VisualCommander displays. You drag and drop from the menus on the left to the displays on the right.

You can customize any window even while the simulation is running. Figure 7-8 on page 18 shows an editing sequence. In this case the user is adding the raw data point "jd" to the window. You use the menu on the right to switch from run to edit mode. The simulation does not stop in edit mode but you can only send commands in run mode.

Table 7-1. Simulation commands

Page	Commands					
Summary	simulation scale - adjust the ratio of sim speed to real time.					
Attitude Determination	Single Frame Command - select the single frame mode for attitude de-					
	termination. This uses a pseudo-inverse.					
Orbit Determination	Reset command, Use range measurement command, use optical mea-					
	surement command and change orbit center command					
Attitude Control	Send a quaternion to reorient the spacecraft. You can also change the					
	attitude maneuver model					

Figure 7-6. Data window in VisualCommander

	Data		 	Data	
	Tree Sessions Network			ree Sessions Network	
NewHorizons Pluto		Metadata	spacecraft id camera 2		Metadata
▶ ControlDeck			spacecraft point 1		
▶ DSim		Type: <none></none>	spacecraft point 2		Type: Integer
		Units: <none></none>	star_data_1		Units:
acs_attitude_gain_a_command			star_data_2		
acs_attitude_gain_b_command	U		star_id_camera_1		Select attitude control mode
acs_attitude_gain_c_command			star_id_camera_2		
acs_attitude_gain_d_command			target_locked		
acs_body_align_axis_command			trk_camera_1_step		
acs_body_rate_gain_command			trk_camera_2_step		
acs_control_mode_command			trk_jd_start_command		
acs_control_period_command			trk_target_camera_1_command		6
acs_eci_align_axis_command		Attribute Value	trk_target_camera_2_command		Attribute Value
acs_inertia_command			acs_attitude_gain_a_command		
acs_omega_command			acs_attitude_gain_b_command		
acs_q_eci_to_body_command			acs_attitude_gain_c_command		
acs_rotate_angle_command			acs_attitude_gain_d_command		
acs_rotate_axis_command			acs_body_align_axis_command		
acs_run_mode_command			acs_body_rate_gain_command		
acs_thruster_matrix_command			acs_control_mode_command		
acs_torque_demand			acs_control_period_command	U.	
ad_covariance_camera_1			acs_eci_align_axis_command		
ad_covariance_camera_2			acs_inertia_command		
ad_gyro_bias_camera_1			acs_omega_command		
ad_gyro_bias_camera_2			acs_q_eci_to_body_command		
ad_initial_covariance_command			acs_rotate_angle_command		
ad_initial_state_command			acs_rotate_axis_command		
ad_meas_noise_covariance_comm	nd		acs_run_mode_command		
ad_plant_noise_covariance_comma	nd		acs_thruster_matrix_command		
ad_q_eci_to_body	*		ad_initial_covariance_command	4	
	Ŧ	+ -	ad_initial_state_command	*	+ -

Figure 7-7. Data flow window in VisualCommander



Figure 7-8. Editing VisualCommander

0 00	New Horizons 0	NewHorizons_Pluto.vcl		New Horizons (NewHorizons_Plut	a.vol
ONC New Herizone					Lot • P
UNS New Honzons			UNS New Honzons		
platform gimbal angle: 1.34483 outer gimbal angle 1: 0.67064 inner gimbal angle 1: 1.81281 outer gimbal angle 2: -1.71138 inner gimbal angle 2: 1.36033	1.5 [3] 6 10[32] 10[32] 10[34] 10[32] 10[34] 10[32] 10[34] 10[35] 10[34] 10[35] 1		jakituri girbal ingle. Ukur girbal ingle : Provi girbal ingle 1: Ukur girbal ingle 2: 201 erwi girbal ingle 2:		
<pre>9k_terpet_camora_2_command: 0 9k_terpet_camora_1_command: 0 9k_terpet_camora_1_stop: 1 9k_camora_2_stop: 1</pre>	.00300 8.00000 0.00000 .00300 125.00000 0.00000 .00300 424.00000 0.00100	B C C C C C C C C C C C C C C C C C C C	Н. Зари, сален, 1, сален И. Зари, сален, 1, сален В Н. сален, 1, яр; В	nd 2,00000 9,00000 0,00000 0,00000 0,00000 0,00000 0,00000 0,00000 0,00000 0,00000 0,00000 0,00000 0,00000 0,00000 0,00000 0,00000 0,000000	1020 1020 <td< td=""></td<>
-0.05322 gimbal, 1, target: 0.00274 -0.78643	-0.08325 sensorBoresight: 0.83558 -0.79494		ginbal_1,suget	-0.08322 -0.08325 0.05274 sensorBoresight: 0.05338 -0.78643 -0.78644	
-0.14505 gimbal,2_target: -0.97450 -0.17380	-0.14417 sensoriBoresight: -0.97404 -0.17454		grimbal, 2, target	-0.14549 0.14428 -0.07359 sensorifismsight: -0.97460 -0.17378 -0.17451	
PrincetonSATELLITE			PrincetonSATEL	LITE	



S New Horizons					
platform gimbal angle: 1.34495	1.5 -+			g 2-+	
outer gimbal angle 1: 0.67064	<u>j</u> 1		5 0.6	1.5	
inner gimbal angle 1: 1.81281	E 0.5				
outer gimbal angle 2: -1.71049	£ 0-		0 gt	0 19	
	10	02 10:04 10:06	10.02 10.04 10.06	10.02 10.04 10.06	
inner gimbal angle 2: 1.38921		time (nn.mm)	Time (nn.mm)	Time (m.mm)	
trk_target_camera_2_command:	2.00000 9.00000	0.00000	g <u>+</u>	g 1.5-+-	
	0.00000 125.00000	0.00000	≥ _0.5		
trk_target_camera_1_command:	0.00000 424.00000	0.00100	-1	0.5	
trk_camera_1_step: 1			-1.S	0	
			10:02 10:04 10:06	10:02 10:04 10:06	
trk_camera_z_step.			(internition)	Time (nn.mm)	
-0.063	22	-0.06325			
gimbal_1_target: 0.602	74 sensorBoresight:	0.60338			
-0.795	43	-0.79494			
-0.146	54	-0.14534			
gimbal_2_target: -0.973	93 sensorBoresight:	-0.97398			
-0.173	17	-0.17392			
			jd: 2457217.92140		
PrincetonSATELLITE					
SYSTEMS					

8 DSimManager

DSimManager is a an application that allows you to visually assemble dynamical models in the simulation. The models reside in the Spacecraft Package or any DSim class library that you may have.

When you open DSimManager two windows appear. One says "Untitled" and is the window for building simulations Figure 8-1. The second is the "Components" window which has your DSim Libraries shown in Figure 8-2 on the following page. The first step is to click on the Integrators tab and add an integrator. An integrator integrates the equations of motion, and effects the time evolution of system quantities. Hit the "+" button and give it a name. There are two built in integrators but you can add your own.

Once you've done that you can drag and drop models into the object hierarchy window shown in Figure 8-2 on the next page. Models are in a hierarchy. The gravity model is a child of the spacecraft model. This facilitates transferring information from one model to the other. Other mechanisms for transferring data such as targets, networks and outlets also are available, and are exposed in tabbed pages in DSimManager.

Figure 8-1. DSimManager

0	Untitled	
	Simulation Integrators Object Hierarchy Test	_
Overall Simulation S	Setup:	
Simulation Name:		
Timestep:	1 seconds	
Initial Offset:	0 seconds	
Duration:	10 Seconds 📑 🖸 Stop after duration	
Start Date:		
Calendar:	8/13/2010 9:29:28 AM 🗘 Milliseconds: 57.197	
Julian:	2455421.89546362	
Time Scale:	1 🗹 Run as fast as possible	
Message Log File:	Browse	
Data Log File:	Browse	
Simulation Commen	nts	

9 Simulation Models

9.1 Spacecraft Dynamics

The translational dynamics are

$$\dot{r} = v \tag{9-1}$$

Figure 8-2. DSimManager: Components window

\varTheta 🔿 Components	
Objects Templates	
🔻 Builtin	
Base Model	
Ideal Gravity	
Ideal Spring	
Rigid Body	
▼ Spacecraft	
Aerodynamics Model	
Air_Data	
Blow Down Propulsion Model	
Camera	
Collision model	
Constraint force due to the ground	
Continuum aerodynamics model	
GPS receiver model	
General disturbance model	
Generic Thermal Model	
Geometery file reader	
Gravity Model With SPICE	
Ground station model	
Horizon Sensor Assembly	
IMU	
ISL model	
Ideal Actuator	
Ideal Earth Gravity Model	
Imager	
Infrared_Imager	
Laser Illumination Model	
Laser Model	
Magnetic Field	
Magnetometer	
Navigation sensor model	
Newtonian aerodynamics model	
RV To Kepler	
Radar Model	
Rigid Body	
Rigid Body RWA	
Rigid Body Stage	
Rocket engine model	
Sensor class	
Single Axis Sun Sensor	
Single axis gimbal model	
Solar pressure disturbance model	
Solid rocket model	
Spherical Harmonic Earth Gravity Model	
Time model	
Tracking station model	
Transform from planet fixed to ECI	
Turbofan ramjet model	
	1

Figure 8-3. DSimManager: select integrator

• •	Untitled	
	Simulation Integrators Object Hierarchy Test]
Name 🔺 Type	Library	Integrator Description
Untitled Runge-Kutta 4th Order	\$ Builtin	A fourth-order fixed- timestep Runge-Kutta integrator implementation, supporting matrix and double integrated value
+ -		
Command Value		Setup Description
		No Selection

Figure 8-4. DSimManager: Drag and drop models

				Simulatio	n Integra	tors Obje	ct Hierarch	y Test	}			
Object Comsat	Type RigidBody	Library	=		Object	Variables	Outlets	Targets	Setup	Networks	Dependencies	
Gravity	IdealGravity	E Spacecraft		Name:	No Selectio	m						
				Type:								* *
					Model Des	cription						
					No Selec	tion						
				Integrator:								\$
			~		Integrator	Description						
					No Selec	tion						
				Comments				Attributes				
								Attribute		▲ Valu	ue	
								+ -				

$$\dot{v} = a \tag{9-2}$$

$$\dot{m}_f = -\frac{F_t}{u_e} \tag{9-3}$$

The acceleration is

$$a = \frac{F_s + F_d + F_t}{m_d + m_f} + a_c + a_p$$
(9-4)

where F_a is the force due to solar pressure, F_d is the force due to atmospheric drag and F_t is the force due to thrusters. a_c is the gravitational acceleration due to the central body and a_p is the acceleration due to the sum of accelerations from all other bodies. u_e is the thruster exhaust velocity. m_f is the fuel mass and m_d is the dry mass.

The rotational dynamics and kinematics are

$$I\dot{\omega} + \omega \times I\omega = T \tag{9-5}$$

$$\dot{q} = f(q,\omega) \tag{9-6}$$

The rotational kinematics are represented by quaternions.

9.2 Disturbance Model

The disturbances of interest are solar pressure and atmospheric drag. The latter is only important near planets with atmospheres. The spacecraft surface is represented by a triangle mesh. Each mesh element is represented by 3 vertices a normal and the following properties

- 1. drag coefficient, C_D
- 2. specular reflection coefficient, ρ_s
- 3. diffuse reflection coefficient, ρ_d
- 4. absorption coefficient, ρ_a
- 5. transmissivity ρ_t

 C_D is between 0 for no drag and 4 for when the surface perfectly reflects incoming particles.

The drag is

$$F_d = -\frac{1}{2}\rho C_d A n^T v v \tag{9-7}$$

where n is the outward unit normal, v is the velocity in the body frame and A is the area of the plate. The outward normal is

$$n = \frac{(v_1 - v_3) \times (v_2 - v_3)}{|(v_1 - v_3) \times (v_2 - v_3)|}$$
(9-8)

Solar pressure is the dominant disturbance on a spacecraft in geosynchronous orbit. Solar pressure is due to the force of photons on the surfaces of the spacecraft. A photon striking a surface can do one of four things: it can be absorbed, it can reflect specularly (meaning at the same angle with respect to the surface normal that it hit), it can reflect diffusely (meaning at any angle), or it can pass through the surface. Photons that are absorbed must either be transferred somewhere else (through heat conduction or as electricity in the case of a solar array) or be remitted locally. If the latter happens, the photon must be lumped in with the diffusely re-emitted photons. In terms of fractions of the incoming photons, the following is true for a surface

$$1 = \rho_a + \rho_s + \rho_d + \rho_t \tag{9-9}$$

where ρ stands for the fraction of photons that are absorbed, specularly reflected, diffusely reflected or transmitted.

The solar pressure force for a surface with only one side seeing free space is

$$F = -SAs^{T}n(2(\rho_{s}s^{T}n + \rho_{d}/3)n + (\rho_{a} + \rho_{d})s)$$
(9-10)

where s is the Sun vector, n is the unit normal to the surface, A is the area of the surface and S is the solar flux in N/m^2 .

For thin membranes (such as solar sails or solar arrays) we can account for front and back simultaneously. Each side of the membrane, front and back, has its own emissivity (ϵ).

In steady state the incoming absorbed solar flux must equal the outgoing re-emitted flux, which follows Boltzmann's law for thermal radiation. Assuming that the front and back temperatures of the object are the same (T_s) then we can write

$$\rho_a P = \sigma \left(\epsilon_f T_s^4 + \epsilon_b T_s^4 \right) \tag{9-11}$$

where P is the incoming solar flux, $P = \Phi \cos(\theta)$, and σ is Boltzmann's constant. We can solve for the equilibrium temperature as

$$T_s = \left(\frac{\rho_a P}{\sigma(\epsilon_f + \epsilon_b)}\right)^{1/4} \tag{9-12}$$

The re-emitted fluxes per unit area for the front and back of the membrane are

$$\Phi_f = \frac{\epsilon_f \rho_a P}{\epsilon_f + \epsilon_b} \tag{9-13}$$

and

$$\Phi_b = \frac{\epsilon_b \rho_a P}{\epsilon_f + \epsilon_b} \tag{9-14}$$

which are independent of temperature. The remaining outgoing fluxes accounting for the specular and diffuse reflected portions of the solar flux are

$$\Phi_s = \rho_s P \tag{9-15}$$

$$\Phi_d = \rho_d P \tag{9-16}$$

From the conservation of energy we know that these four outgoing fluxes must equal the incoming flux. The total force exerted is

$$F = \cos\theta \left(f_s + f_d + f_f + f_b \right) \tag{9-17}$$

and the normalized force contributions due to each flux are

$$f_s = -2\cos\theta\Phi_s\hat{n} \tag{9-18}$$

$$f_d = -\Phi_d \left(\frac{2}{3}\hat{n} + \hat{s}\right)$$

$$f_f = -\frac{2}{3}\Phi_f \hat{n} + \Phi_f \hat{s}$$

$$f_b = \frac{2}{3}\Phi_b \hat{n} + \Phi_b \hat{s}$$
(9-19)

Combining terms we get the final form of the thermal/optical force model.

$$F = -pA\cos\theta \left(2\left[\rho_s\cos(\theta) + \frac{1}{3}\left(\rho_d + \rho_a\frac{\epsilon_f - \epsilon_b}{\epsilon_f + \epsilon_b}\right)\right]\hat{n} + \left[\rho_d + \rho_a\right]\hat{s}\right)$$
(9-20)

If the surface does not have a back set ϵ_f equal to ϵ_b . The resulting equation is the conventional single sided surface model. The spacecraft will be subdivided into triangles. Each triangle will have different surface properties which will allow the disturbance model to handle a wide variety of spacecraft.

The pulse widths are decremented by the simulation time step after the force and torque have been applied to the spacecraft.

9.3 Gravity Model

9.3.1 Point Mass Model

The gravity model assumes a point mass model of the center and a perturbation model for all additional bodies. Any body available in the ephemeris model can be added.

The point mass model is

$$a = -\mu \frac{r}{|r|^3} \tag{9-21}$$

where r is the vector from the central body to the spacecraft.

The perturbation model is

$$a = -G \sum m_j \left(\frac{d_j}{|d_j|^3} + \frac{\rho_j}{|\rho_j|^3} \right)$$
(9-22)

where d is the vector from the spacecraft to the secondary body and ρ is the vector from the central body to the secondary body, i.e. $\rho = r + d$.

The model uses the SPICE library for the planetary ephemeris. The model can be used in the ECI or ecliptic frames.

9.3.2 Spherical Harmonic Model

The point mass gravity field for the central body can be replaced by a spherical harmonic gravity model. There are many ways to deal with asymmetries in planets. One would be to model the planet as a set of point masses contained within the surface of the planet. The second is to expand the gravitational potential in a spherical harmonic expansion and compute the gravity forces from the gradient of the potential. Other expansions could also be used.

The spherical harmonic expansion method works as follows. Define the perturbing gravitational potential of a planet as

$$V = -\frac{\mu}{r} \sum_{n=2}^{\infty} \left[\left(\frac{a}{r}\right)^n \sum_{m=0}^{\infty} \left(S_{n,m} \sin m\lambda + C_{n,m} \cos m\lambda\right) P_{n,m}(\sin \phi) \right]$$
(9-23)

defined in the planet fixed frame. a is the radius of the planet.

If we define i, j, k as unit vectors along the planetary x, y and z axes, the gravitational acceleration can be found as follows

$$\frac{\partial V}{\partial r} = -\sum_{n=2}^{\infty} \sum_{m=0}^{n} \frac{\partial V_{n,m}}{\partial r}$$
(9-24)

and

$$\frac{\partial V_{n,m}}{\partial r} = \frac{\mu}{r^2} \left(\frac{a}{r}\right)^n \left[-r(vH_{n,m} + B_{n,m}) + iD_{n,m} - jE_{n,m} + kH_{n,m}\right]$$
(9-25)

$$B_{n,m} = (C_{n,m}C_m + S_{n,m}S_m)(n + m + 1)P_n^{m}$$

$$E_{n,m} = -m(C_{n,m}\hat{S}_{m-1} - S_{n,m}\hat{C}_{m-1})$$

$$D_{n,m} = m(C_{n,m}\hat{C}_{m-1} + S_{n,m}\hat{S}_{m-1})$$

$$H_{n,m} = (C_{n,m}\hat{C}_m + S_{n,m}\hat{S}_m)P_n^{m+1}$$

$$\hat{C}_m = \hat{C}_1\hat{C}_{m-1} - \hat{S}_1\hat{S}_{m-1}$$

$$\hat{S}_m = \hat{S}_1\hat{C}_{m-1} + \hat{C}_1\hat{S}_{m-1}$$
(9-26)

The starting conditions are

$$\hat{C}_{0} = 1 \quad \hat{S}_{0} = 0
\hat{C}_{1} = \frac{x}{r} \quad \hat{S}_{1} = \frac{y}{r}$$
(9-27)

The derivatives of the Legendre Polynomials are

$$P_n^0 = \frac{1}{n} \left[(2n-1)v P_{n-1}^0 - (n-1)P_{n-2}^0 \right]$$
(9-28)

$$P_n^m = P_{n-2}^m + (2n-1)P_{n-1}^{m-1}$$

$$P_0^0 = 1$$
(9-29)

$$P_{1}^{0} = \frac{z}{r} \equiv v$$

$$P_{0}^{1} = 0$$

$$P_{1}^{1} = 1$$

This kind of harmonic expansion is the standard method for modeling a planets gravitational field. Harmonic models exist for the Earth, Moon, Mars and other planets. A harmonic model is convenient because it gives an idea of how much influence higher order terms have on the overall force.

9.4 Camera Model

The camera model is a pinhole camera has a single ray per point on the target and that ray maps to a point on the focal plane. A point P(X, Y, Z) is mapped to the imaging plane by the relationships

$$u = \frac{fX}{Z} \tag{9-30}$$

$$v = \frac{\bar{f}Y}{Z} \tag{9-31}$$

where u and v are coordinates in the focal plane, f is the focal length and Z is the distance from the pinhole to the point along the axis normal to the focal plane. This assumes that the Z-axis of the coordinate frame X, Y, Z is aligned with the boresight of the camera.

Two models are provided. The first is the imager.cc model which computes the centroids of

- 1. Stars
- 2. Planets
- 3. Spacecraft

In addition it computes the location of landmarks on planets and points on spacecraft. These are output as a matrix for processing by the estimation software. This bypasses processing of the images to facilitate testing of the estimators. The centroids of the planets are their centers as used by the orbit dynamics model.

The second model is the camera.cc. This creates a synthetic scene using the OpenGL pinhole camera model. The frame can be processed by image processing to compute the pixel plane locations of the centroids.

9.5 IMU Model

The gyro model is

$$\omega_m = \omega + c + b + \nu_\omega \tag{9-32}$$

$$\dot{b} = -\frac{1}{\tau}b + \nu_b \tag{9-33}$$

where c is a constant error, b is a bias and ν_{ω} is output noise. ν_b is the random walk noise. τ is the time constant for the bias. This would cause the bias to decay with time if not driven by white noise. The accelerometer model is

$$\ddot{x}_m = \ddot{x} + c + b + \nu_{\ddot{x}} \tag{9-34}$$

$$\dot{b} = -\frac{1}{\tau}b + \nu_b \tag{9-35}$$

where c is a constant error, b is a bias and ν_{ω} is output noise. ν_b is the random walk noise.

9.6 Ranging Model

The ranging model returns range and range-rate between a planet fixed point and the spacecraft.

$$\rho = \sqrt{r_x^2 + r_y^2 + r_z^2}$$
(9-36)

$$\dot{\rho} = \frac{v^T r}{\rho} \tag{9-37}$$

9.7 State Sensor Model

The state sensor state_sensor.cc is useful for testing purposes. It returns the vehicle rigid-body state in a single vector.

$$x = \begin{bmatrix} r_x \\ r_y \\ r_z \\ v_x \\ v_y \\ v_z \\ q_s \\ q_s \\ q_x \\ q_y \\ q_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$
(9-38)

The zero quaternion has $q_s = 1$.

9.8 GPS Model

The GPS model returns range and range rate between all visible GPS satellites. It also includes their Cartesian state vector. The GPS satellite positions propagated assuming that their orbits are circular. Line of sight is compute for each satellite with a selectable angle for blocking the signal. This model is not a highly accurate model of the GPS constellation but is sufficient for software testing. The GPS constellation in the model is shown in Figure 9-1. The model also accounts for the direction of the GPS constellation. Thus when a satellite is above the constellation altitude it will only see satellites that are on the other side of the earth but not blocked by the earth.





9.9 Intersatellite Link Model

The Intersatellite Link (ISL) model returns range and range rate between the target satellite and the core satellite. It also includes the target satellite Cartesian state vector.

9.10 Radar Model

Figure 9-2. Radar frame of reference



If the transformation matrix from ECI to the escort frame is m_{eb} and the transformation matrix from the body to the radar frame is m_{eb} then the relative position vector is

$$r = m_{br} \left(m_{eb} (r_t - r_e) - c \right)$$
(9-39)

where c is the vector from the spacecraft center-of-mass to the origin of the radar frame. The gravitational equations are referenced to the center-of-mass of each spacecraft. The relative rate vector is

$$\dot{r} = m_{br} \left(\dot{m}_{eb} (r_t - r_e) + m_{eb} (\dot{r}_t - \dot{r}_e) \right)$$
(9-40)

if we assume that c and m_{br} are fixed.

Range rate is determined from the Doppler shift of the signal so it is a separate measurement from range. Azimuth and elevation are from the measured angles of the radar beam. Their rates are not directly measured. If α is azimuth and β is elevation then

$$\alpha = \tan^{-1} \frac{r_y}{r_x} \tag{9-41}$$

$$\beta = \sin^{-1} \frac{r_z}{\rho} \tag{9-42}$$

where $\rho = |r|$.

9.11 Propulsion Model

The propulsion model allows an array of thrusters to be modeled with independent thrust parameters. A single tank blowdown system is assumed. The fuel pressure for the unregulated blowdown system is

$$P = \frac{m_{\rm He}R_{\rm He}T}{V - m_f/\rho_f} \tag{9-43}$$

where m_{He} and R_{He} are the mass and gas constant of the Helium pressurant, T is the fuel temperature, V is the tank volume, and m_f and ρ_f are the mass and density of the hydrazine fuel.

The thrust commands are pulse widths to each thruster. Any pulse width commands less than the thruster minimum impulse bits are ignored. The remaining commands are scaled if the pulse width is less than the simulation time step, so that the correct total force is applied to the spacecraft.

$$k = \begin{bmatrix} \Delta t_c / \Delta t_{sim}, & \Delta t_c < \Delta t_{sim} \\ 1.0, & \text{otherwise} \end{bmatrix}$$
(9-44)

The thrust of each thruster is computed from a coefficient a_0 and the ratio of the pressure to the thruster nominal pressure.

$$\mathcal{T} = a_0 k \frac{P}{P_0} \tag{9-45}$$

The mass flow is computed from the thrust and exhaust velocity of each thruster.

$$\dot{m} = -\sum \frac{\mathcal{T}_i}{u_{e_i}} \tag{9-46}$$

The total force is a sum of the thrust times each thruster's unit vector. The torque is the cross product of the force with the vector to the spacecraft center of mass.

$$F = \sum \mathcal{T}_i \hat{u}_i \tag{9-47}$$

$$T = \sum (\vec{r} - r_{CM}) \times F \tag{9-48}$$

9.12 Gimbal Model

The gimbal model is a first order model in which the input command is commanded rate

$$\dot{\theta} = \omega_c \tag{9-49}$$

The gimbals may be stacked on top of other gimbals. The model does not include dynamical effects of gimbal motion. This is a reasonable simplification because the camera assembly has much lower inertia than the core spacecraft.