Virtual Simulation for Testing a Vision-Based Autonomous Docking and Navigation Algorithm

John Karasinski, Christopher Lorenzen, and Melanie Stich
Department of Mechanical & Aerospace Engineering
University of California, Davis
Davis, CA, USA

***Abstract*—The abstract goes here.**

***Index Terms*—robots, satellite**

I. Introduction

This demo file is intended to serve as a “starter file” for IEEE journal papers produced under LaTeX using IEEEtran.cls version 1.8 and later.

I wish you the best of success.

A. Background

Subsection text here.

B. Motivation

Motivation here

C. Objective

What is our specific question?

D. Outline

Don’t know if we want this section, but just copied it over from Joshi’s example paper.

II. Literature Review

A. SPHERES Vertigo

B. Anti-Collision Algorithms

C. Visual Simulations

III. The Environment

A. The Spacecraft Environment

Space is an extremely dangerous environment with many challenges of different natures and magnitudes. Some effects on spacecraft can arise from radiation, space debris and meteoroid impact, upper atmospheric drag, spacecraft electrostatic charging, and many other factors. While the resulting dynamics of these phenomena have various degrees of effect on small spacecraft, these are outside the scope of this work. For the purposes of the study, all of these factors are ignored. It is assumed here that the spacecraft is operating under nominal conditions in an otherwise empty environment.

B. Dynamical Equations of Motion

The Clohessy-Wiltshire equations were used to model the dynamics of the spacecraft while it was in close proximity to the target vehicle. The Clohessy-Wiltshire equations describe a simplified model of orbital relative motion, in which the target is in a circular orbit, and the chaser spacecraft is in an elliptical or circular orbit. This model gives a first-order approximation of the chaser's motion in a target-centered coordinate system. It is used here in planning rendezvous of the chaser with the target [2].

These equations of motion can be expressed as

$$\begin{aligned} \ddot{x} &= 3n^2 + 2n\dot{y} \\ \ddot{y} &= -2n\dot{x} \\ \ddot{z} &= -n^2 z \end{aligned} \tag{1}$$

and have the closed form solution given by

$$\begin{aligned} x(t) &= (4 - 3\cos nt)x_0 + \frac{sinnt}{n}\dot{x}_0 + \frac{2}{n}(1 - \cos(nt))\dot{y}_0 \\ y(t) &= 6(\sin nt - nt)x_0 + y_0\frac{-2}{n}(1 - \cos nt)\dot{x}_0 \\ &\quad + \frac{4\sin nt - 3nt}{n}\dot{y}_0 \\ z(t) &= z_0 \cos nt + \frac{\dot{z}_0}{n}\sin nt \end{aligned} \tag{2}$$

where

$$n = \sqrt{\frac{\mu}{a_t^3}} \tag{3}$$

and a_t is the semi-major axis of the target vehicle's orbit and μ is the standard gravitational parameter.

IV. Simulation

The virtual simulation was driven with a variety of software packages. The dynamics and control systems were driven by several Python modules, while the visualization was rendered in EDGE. EDGE is a graphics display tool developed at NASA's Johnson Space Center that combines key elements from graphics software developed for the space shuttle and the International Space Station programs and adapts them for integration with other engineering simulations and facilities [1].

EDGE makes use of a node tree to structure data, objects, and models. Each node has many properties, the most notable of which are position, orientation, and the node's parent. Each node's position and orientation are defined relative to it's parent's position and attitude. A node's parent can be changed to a different node, at which time the node's position and attitude are automatically updated to the correct values.

Three Python Python 2.7.9 scripts were developed for use in the virtual simulation:

cv
: A computer vision image processing module which takes advantage of OpenCV-Python version 2.4.9.

dynamics
: A dynamics and controls module.

dcomm
: A wrapper library developed to interface with EDGE's C++ DCOMM library.

A. Python Modules

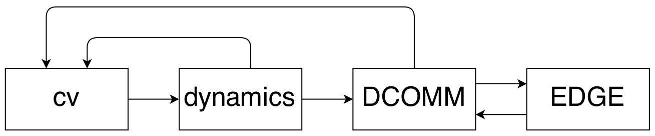

Fig. 1. Block diagram of software package interactions

1) cv: The *cv* module takes advantage of Python bindings for OpenCV. OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. This module takes the live video feed from EDGE and attempts to identify features visible on the target spacecraft. The features of the target are supplied to the module a priori, and these descriptions were used to estimate the 3D pose of the target vehicle relative to the spacecraft's camera. This pose information is passed on to the *dynamics* module

2) dynamics: The *dynamics* module drives the dynamics and control of the simulation. The movement of the spacecraft around the target craft is modeled by the Clohessy-Wiltshire equations, see section III-B. This module controls how the spacecraft operated in its different modes. Depending on pilot input, the spacecraft can navigate purely on line of sight, laser guided sensors to approach and hold a distance from a target, make use of the optical camera and machine vision to identify and dock with docking ports, or display guidance for a human pilot.

3) DCOMM: The *DCOMM* module was previously developed to network between Python scripts and EDGE. The Python DCOMM interface allows a user to call various C++ functions from EDGE's DCOMM module and communicate with an EDGE server. Users can move and rotate nodes, and can also change a node's parents, units, and principal axis definitions. Commands to set the spacecraft's attitude and position were sent from the *dynamics* module through DCOMM and passed on to EDGE. This module was also responsible for requesting and handing off the video feed from the EDGE server.

V. Architectures

Use this section to describe which Joshi-defined architecture was implemented for our simulation (ie. Reactive, Deliberative Architecture, or both) **** DOES THIS SECTION WAIT TILL THE FINAL PAPER??

VI. Techniques and Algorithms

not sure if this section should only be used for the final paper

VII. Conclusion

The conclusion goes here.

References

[1] NASA, "Enhanced graphics tools for advanced 3d simulations," 2015.
[2] Ocampo, "Clohessy-wiltshire equations," 2015.
[3] Opencv.org, "Opencv," 2015.