Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

Document ID: Basilisk-horizonOpNav

LIMB-BASED OPTICAL NAVIGATION MODULE

Prepared by	T. Teil

Status: Draft
Scope/Contents
Converter that takes circle information from image processing and turns it into a inertial position. This module also maps the uncertainty from center and apparent diameter into a position uncertainty.

Rev	Change Description	By	Date
1.0	Initial Release	T. Teil	2019-05-27

Contents

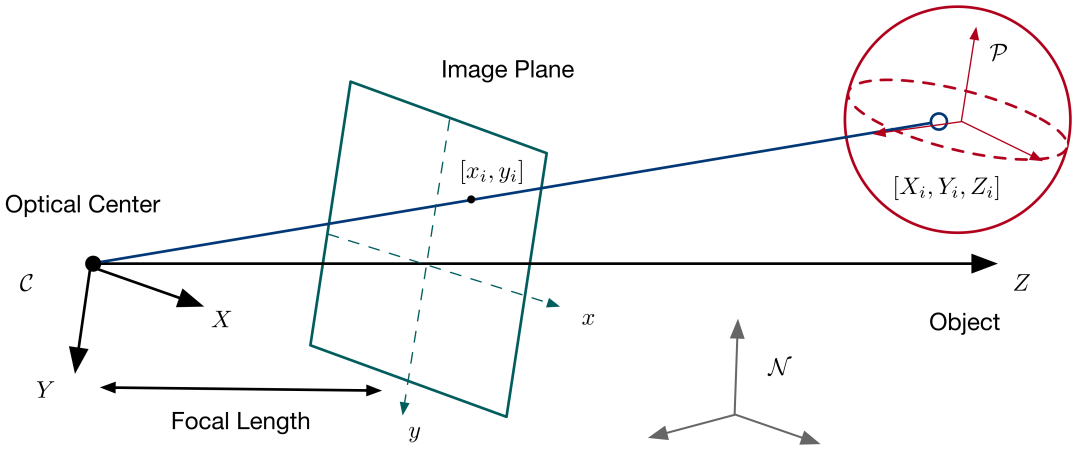

Fig. 1: Camera Model

1 Model Description

1.1 Input and Output

This converter module processes the output of a limb finding method to extract spacecraft inertial position. It does this by reading spacecraft attitude (coming from star tracker or other means), camera parameters, and the limb data.

Messages read:

- CameraConfigMsgPayload: containing focal length, resolution, and sensor size. These values are needed for the following computations. Notably the camera frame relative to the body frame is used.
- LimbInMsgPayload: Limb points, and uncertainty around these values in pixels.

- NavAttMsgPayload: Used for the spacecraft attitude. This allows to move from the body frame to the inertial frame.

Message written:

- OpNavMsgPayload: Message containing ${}^{\mathcal{N}}\boldsymbol{r}$ and it's covariance.

1.2 Position computation

The details of the algorithm are summarized in the papers attached. The engineering note contains a summary of the algorithms and covariance analysis. The journal paper[?] contains the details of the development and assumptions.

The component that is chosen in the implementation is the way to solve the least squares for equation

$$[H]\boldsymbol{x} = \mathbf{1}$$

This is done in this module by performing a QR-decomposition via a a Gram-Schmidt process. This leads to the following equation:

$$[R]\boldsymbol{x} = [Q]^T\mathbf{1}$$

Since $[R]$ is upper-triangular, this is solved with an implemented back-substitution.

2 Module Functions

- **Update**: The state computation described in the references is all done in the update method for the module.
- **QRDecomp**: A QR decomposition is developed in order
- **BackSub**: A Back-Substitution method is implemented

3 Module Assumptions and Limitations

The main assumptions used in this module are :

- Light-time is not modeled
- The uncertainty is mapped using a first variation method, higher order terms are not taken into account

4 Test Description and Success Criteria

- The first test runs the QR-decomposition and the back-substitution
- The second test for this converter modules creates all three input messages. Using the same values added in those messages, it mirrors the code to output the position and covariance of the spacecraft.

5 Test Parameters

The unit test verify that the module output message states match expected values.

Table 2: Error tolerance for first test.

Output Value Tested	Tolerated Error
QR	1.0E-10
BackSub	1.0E-10

Table 3: Error tolerance for second test.

Output Value Tested	Tolerated Error
r_N	0.001
covar_N	1e-05

6 Test Results

The unit test is expected to pass.

Table 4: Test results

Check	Pass/Fail
1	PASSED
2	PASSED

7 User Guide

If the modules outputting the messages needed are in place, the message names just need to be matched. If not, the module content looks like the following:

- Create the input messages:
```
inputCamera = messaging.CameraConfigMsgPayload()
inputLimb = messaging.LimbOpNavMsgPayload()
inputAtt = messaging2.NavAttMsgPayload()
```

- Set camera:
```
inputCamera.focalLength = 1.
inputCamera.sensorSize = [10, 10]
inputCamera.resolution = [512, 512]
inputCamera.sigma_BC = [1.,0.,0.]
```

- Set Limb:
```
inputLimb.limbPoints = [226., 113., 227., 113., 223., 114., 224., 114., 225.,
114., 219., 115.]
inputLimb.valid = 1
inputLimb.numLimbPoints=6
inputLimb.timeTag = 12345
```

- Set attitude:
```
inputAtt.sigma_BN = [0., 1., 0.]
```

- Set module for Mars:
```
pixelLine.planetTarget = 2
```