Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

Document ID: Basilisk-pixelLineConverter

PIXEL/LINE INFORMATION TO NAVIGATION MESSAGE CONVERTER

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

1 Model Description 1
1.1 Input and Output . . . 1
1.2 Position computation . . . 2
1.3 Uncertainty computation . . . 2
2 Module Functions 2
3 Module Assumptions and Limitations 3
4 Test Description and Success Criteria 3
5 Test Parameters 3
6 Test Results 3
7 User Guide 3

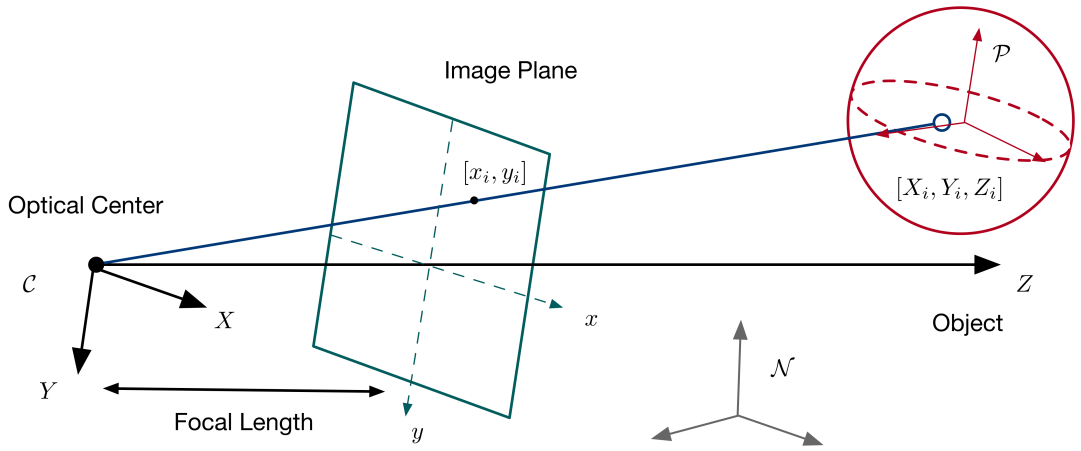

Fig. 1: Camera Model

1 Model Description

1.1 Input and Output

This converter module processes the output of a circle 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 circle properties.

Messages read:

- CameraConfigInMsg: 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.
- CirclesInMsg: Circle radius, center pixel and line, and uncertainty around these values in pixels.

- NavAttInMsg: 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

A geometrical method can be used to extract pose information from center and apparent diameter information. The norm of the position vector is given by the apparent size, it's direction is given by the pixel and line data. Using ${}^{\mathcal{C}}\boldsymbol{r}_c = {}^{\mathcal{C}}\begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}^T$ as the relative vector of the camera with respect to the celestial center, A as the apparent diameter of the celestial body, D as the actual diameter:

$$|\boldsymbol{r}_c| = \frac{1}{2}\frac{D}{\sin\left(\frac{1}{2}A\right)} \qquad \frac{1}{r_3}\begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = \frac{1}{r_3}\tilde{\boldsymbol{r}} = \frac{1}{f}\begin{bmatrix} x \\ y \end{bmatrix} \tag{1}$$

These equations have been used in multiple instances.[?,?] The third component of $\boldsymbol{r}_c$ provides the range measurement to the body which can be extracted using the apparent diameter measurements. Hence the definition of $\tilde{\boldsymbol{r}}$ which only contains the first two components of $\boldsymbol{r}_c$. The vector components of $\boldsymbol{r}_c$ can be expressed relative to the inertial frame assuming inertial attitude knowledge from other instruments. Using the position of the camera on the spacecraft this provides the measurement value for an orbit determination filter using a circle-finding algorithm.

1.3 Uncertainty computation

In the case of the geometric formula, the partials allow to quantify error due to the camera specifications. Indeed, if X, Y the pixel sizes (in their respective directions), x, y are the position on the camera, and x_i, y_i, ρ_i the pixel values for the CAD measurements:

$$\tilde{\boldsymbol{r}} = \frac{r_3}{f}\begin{bmatrix} x \\ y \end{bmatrix} = \frac{r_3}{f}\begin{bmatrix} x_i X \\ y_i Y \end{bmatrix} \tag{2}$$

$$|\boldsymbol{r}_c| = \frac{1}{2}\frac{D}{\sin\left(\frac{1}{2}A\right)} = \frac{1}{2}\frac{D}{\sin\left(\arctan\left(\frac{\rho}{f}\right)\right)} = \frac{1}{2}\frac{D}{\sin\left(\arctan\left(\frac{\rho_i X}{f}\right)\right)} \tag{3}$$

Eq. (2) provides a simple partial with respect to the measurement $\boldsymbol{c}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^T$

$$\frac{\partial\tilde{\boldsymbol{r}}}{\partial\boldsymbol{c}_i} = r_3\begin{bmatrix} \frac{X}{f} & 0 \\ 0 & \frac{Y}{f} \end{bmatrix} \quad \Rightarrow \quad \mathbb{E}\left[\delta\tilde{\boldsymbol{r}}\delta\tilde{\boldsymbol{r}}^T\right] = r_3^2\begin{bmatrix} \frac{X}{f} & 0 \\ 0 & \frac{Y}{f} \end{bmatrix}\left[\delta\boldsymbol{c}_i\delta\boldsymbol{c}_i^T\right]\begin{bmatrix} \frac{X}{f} & 0 \\ 0 & \frac{Y}{f} \end{bmatrix} \tag{4}$$

The partial for Eq. (3) is:

$$\frac{\partial|\boldsymbol{r}_c|}{\partial\rho_i} = \frac{Dd_x}{2}\sqrt{f^2+\rho^2 d_x^2}\left(\frac{1}{f^2+\rho^2 d_x^2} - \frac{1}{\rho^2 d_x^2}\right) \tag{5}$$

2 Module Functions

- **Update**: The math described previously is all done in the update method for the module.

3 Module Assumptions and Limitations

The main assumptions used in this module are :

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

4 Test Description and Success Criteria

The unit test for this converter modules creates all three input messages. Using the same values added in those messages, it computes the expected 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 each test.

Output Value Tested	Tolerated Error
r_N	1e-10
covar_N	1e-10
timeTag	1e-10

6 Test Results

The unit test is expected to pass.

Table 3: Test results

Check	Pass/Fail
1	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 = pixelLineConverter.CameraConfigMsg()
inputCircles = pixelLineConverter.CirclesOpNavMsg()
inputAtt = pixelLineConverter.NavAttIntMsg()
```

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

- Set circles:
```
inputCircles.circlesCenters = [152, 251]
inputCircles.circlesRadii = [75]
inputCircles.uncertainty = [0.5, 0., 0., 0., 0.5, 0., 0., 0., 1.]
inputCircles.timeTag = 12345
```

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

- Set module for Mars:
```
pixelLine.planetTarget = 2
pixelLine.opNavOutMsgName = "output_nav_msg"
unitTestSim.TotalSim.logThisMessage(pixelLine.opNavOutMsgName)
```