Autonomous Vehicle Simulation (AVS) Laboratory, University of Colorado

Basilisk Technical Memorandum

Document ID: Basilisk-limbFinding

MODULE TO FIND PLANET LIMB

Prepared by	T. Teil

Status: 1st Draft
Scope/Contents
This module either reads in a message of type `CameraImageMsgPayload`, or a filename for an image. It then uses the OpenCV's Canny method to extract the limb points. It then writes these messages in a message of type `LimbOpNavMsgPayload`.

Rev	Change Description	By	Date
1.0	First release	T. Teil	September 16, 2019

Contents

---

1 Model Description

This module imports OpenCV which is installed if the cmake option is triggered. Two cases are possible: a filename for an image can be given, or an image message containing a size and pointer to image data can be used. If the message is read, the size is used to decode the image into an OpenCV matrix for image processing.

The following methods are then applied to the image:

- Greyscale
- Blurr
- Canny

This allows for the limb points to be detected. The default parameters are set to be efficient for planet edge detection, as the test examples show.

2 Module Functions

- **Update State**: The image processing takes place in the Update method.

3 OpenCV Functions

More documentation is available on at https://docs.opencv.org/4.0.0/.

- void **cv::Canny**(*InputArray* image, *OutputArray* edgeImage, *double* cannyThresh1 = 100, *double cannyThresh1* = 200)
- void **cv::cvtColor**(*nputArray* src, *OutputArray* dst, *int* code)
- void **cv::blur**(*InputArray* src,*OutputArray* dst,*Size* ksize,*Point* anchor = Point(-1,-1),*int* borderType = BORDER_DEFAULT)

4 Module Assumptions and Limitations

The limitations of this module are in the image processing capabilities of the components. Current the main limitation is the lack of uncertainty measure around the limb point estimates.

5 Test Description and Success Criteria

In order to test the proper function of this module, three test images are provided. The algorithm needs to find the limbs and match the expected results.

6 Test Parameters

For each image, four tests are run:

Table 2: Error tolerance for each test.

Test	Absolute Error
Validity Flag	1E-5 [-]
Covariance Values	1E-5 [px^2]
Number of Limb Points	1E-5 [-]
First Limb Point Values	1 (Relative Error)

7 Test Results

The following table shows the results of the unit test described above.

Table 3: Test results

Check	Pass/Fail
MarsBright	PASS
MarsDark	PASS
Moon	PASS

The test does not generate the result image unless called explicitly from python in order to not add images to the repository.

The Figures draw in red the exact pixel points that are detected at the limb.

8 User Guide

This section contains information directed specifically to users. It contains clear descriptions of what inputs are needed and what effect they have. It should also help the user be able to use the model for the first time.

- Construct algorithm and associated C++ container:
```
moduleConfig = limbFinding.LimbFinding()()
moduleConfig.ModelTag = "limb"
```

- Add test module to runtime call list:
```
unitTestSim.AddModelToTask(unitTaskName, moduleConfig)
```

- Image processing parameters:
```
moduleConfig.filename = imagePath
moduleConfig.cannyThreshHigh = cannyHigh
```

Fig. 1: Mars Circles

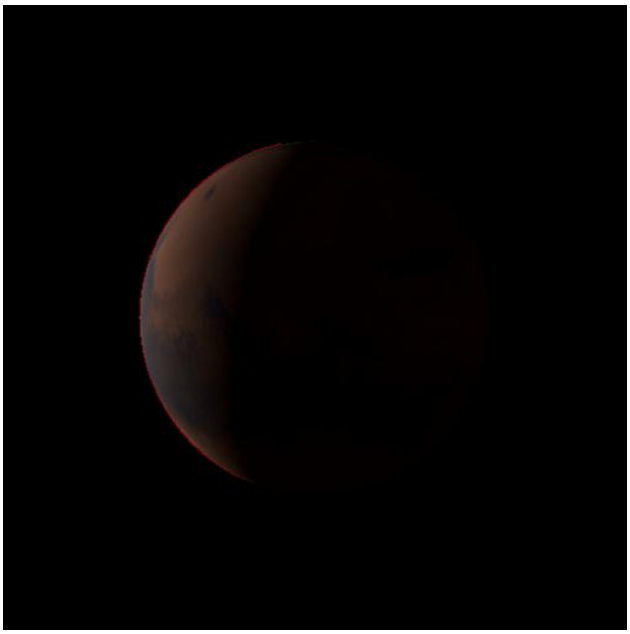

Fig. 2: Mars Circles

```
moduleConfig.cannyThreshLow = cannyLow
moduleConfig.blurrSize = blur
```

Fig. 3: Moon crescents circles