Technical Resources

1. Rover model details and controls algorithms used:
   - https://www.mathworks.com/help/sm/ug/mars_rover.html
   - https://www.mathworks.com/matlabcentral/fileexchange/105700-mars-rover-model-in-simscape/?s_tid=LandingPageTabfx
   - Pure Pursuit controller: https://www.mathworks.com/help/robotics/ug/pure-pursuit-controller.html
2. Path planner used:
   - Hybrid A star planner https://www.mathworks.com/help/nav/ref/plannerhybridastar.html
3. Deep Learning algorithm used:
   - Yolov2: https://www.mathworks.com/help/deeplearning/ug/object-detection-using-yolo-v2.html
   - https://www.mathworks.com/help/vision/ug/getting-started-with-object-detection-using-deep-learning.html
4. Stereo Vision algorithm:
   - https://www.mathworks.com/help/vision/ug/depth-estimation-from-stereo-video.html
   - https://www.mathworks.com/help/vision/ug/using-the-stereo-camera-calibrator-app.html

Details regarding how the detector network was constructed/trained:

Architecture

We retrained one of the models from the following example: https://www.mathworks.com/help/vision/ug/train-yolo-v2-network-for-vehicle-detection.html

Our model is a series network with the following details:

Idx	Layer name	Layer type	Properties
1	input	Image Input	128x128x3 images
2	conv_1	Convolution	16 3x3 convolutions with stride [1 1] and padding [1 1 1 1]
3	BN1	Batch Normalization	Batch normalization
4	relu_1	ReLU	ReLU
5	maxpool1	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
6	conv_2	Convolution	32 3x3 convolutions with stride [1 1] and padding [1 1 1 1]
7	BN2	Batch Normalization	Batch normalization
8	relu_2	ReLU	ReLU
9	maxpool2	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
10	conv_3	Convolution	64 3x3 convolutions with stride [1 1] and padding [1 1 1 1]
11	BN3	Batch Normalization	Batch normalization

12	relu_3	ReLU	ReLU
13	maxpool3	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
14	conv_4	Convolution	128 3x3 convolutions with stride [1 1] and padding [1 1 1 1]
15	BN4	Batch Normalization	Batch normalization
16	relu_4	ReLU	ReLU
17	yolov2Conv1	Convolution	128 3x3 convolutions with stride [1 1] and padding same
18	yolov2Batch1	Batch Normalization	Batch normalization
19	yolov2Relu1	ReLU	ReLU
20	yolov2Conv2	Convolution	128 3x3 convolutions with stride [1 1] and padding same
21	yolov2Batch2	Batch Normalization	Batch normalization
22	yolov2Relu2	ReLU	ReLU
23	yolov2ClassConv	Convolution	24 1x1 convolutions with stride [1 1] and padding [0 0 0 0]
24	yolov2Transform	YOLO v2 Transform Layer.	YOLO v2 Transform Layer with 4 anchors.
25	yolov2OutputLayer	YOLO v2 Output	YOLO v2 Output with 4 anchors.

For more details, load the network object and type the following command:

```
>> deepNetworkDesigner(detector.Network)
```

Dataset

Our dataset is formed by 19953 synthetic images with dimensions 420x560x3 pixels, generated using a simulator. We split them into training and validation datasets, using 2.5% of them for validation. Labels are bounding boxes represented as 1x4 vectors for each rock in a specified image, representing [*x,y,w,h*]:

- *x* and *y* represent the top corner of the bounding box
- *w* and *h* represent the width and height of the rectangle

Training

The proposed model was trained for 24 hours using a Nvidia RTX A5000, using a minibatch size of 3000 images for 500 epochs. The optimizer was SGDM, with a learn rate of 0.0001.

Find more information

https://www.mathworks.com/discovery/object-detection.html
https://www.mathworks.com/help/vision/ug/train-yolo-v2-network-for-vehicle-detection.html