Catalysts

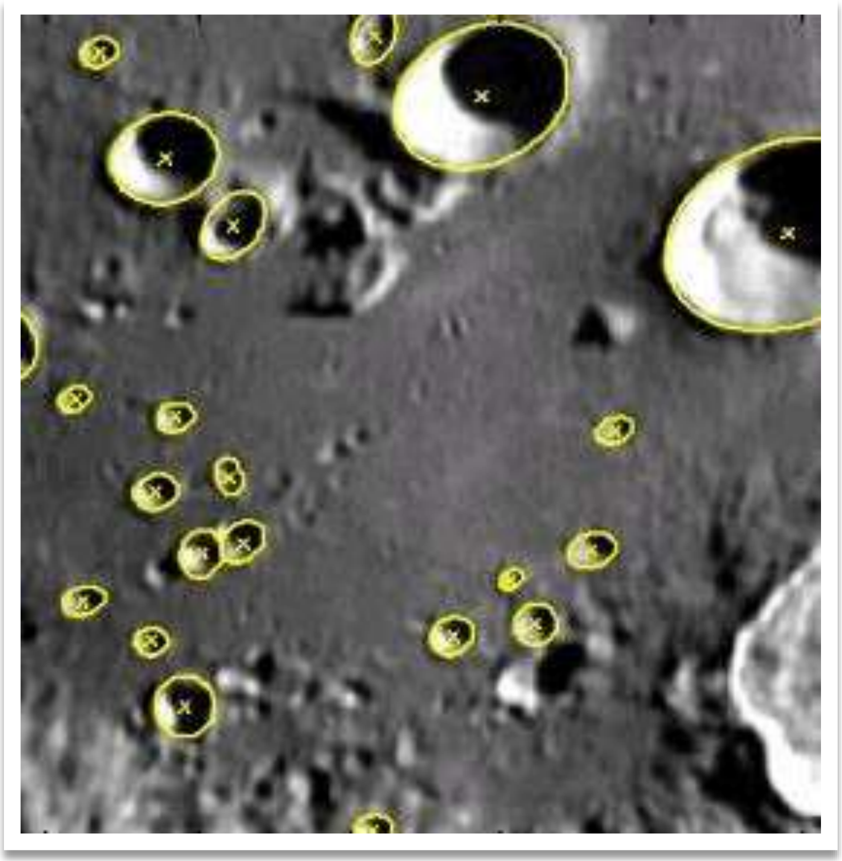
7th Catalysts' Coding Contest

Hagenberg / Austria

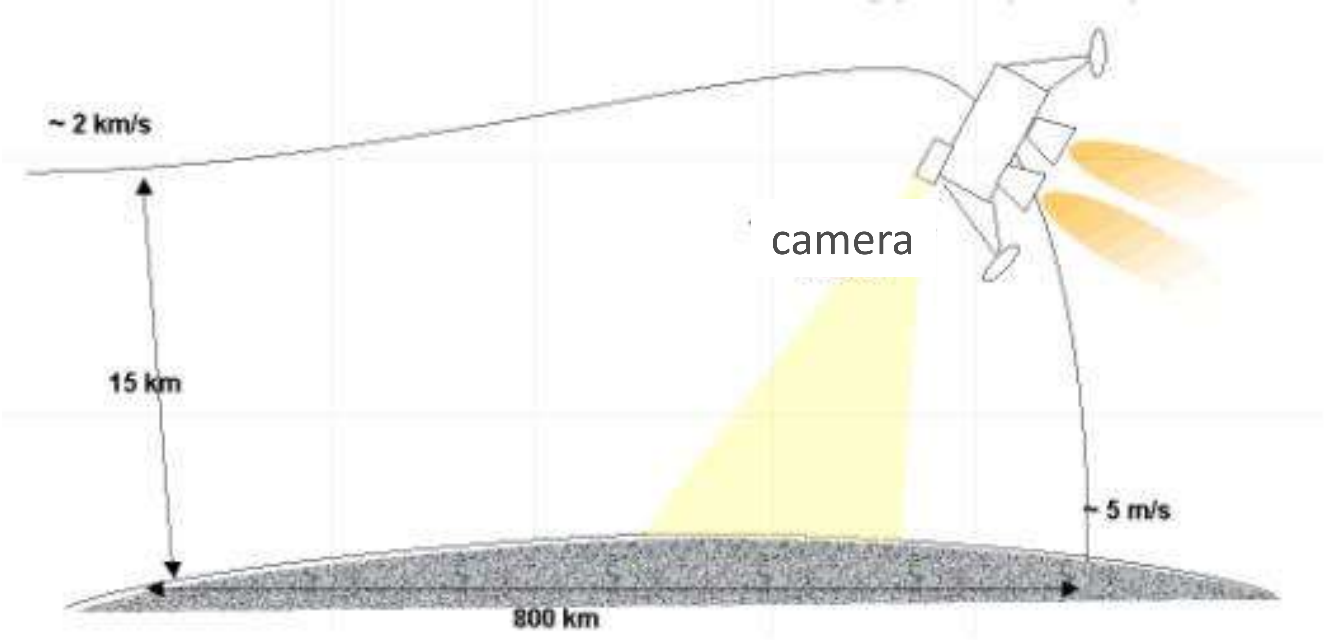
December 16th, 2011

Crater Detection

- The Goal: Automatically detect craters on the surface of planets, moons or asteroids.
- Why? We don’t want a rover to land in a narrow crater when the entire mission costs 200 Million dollars!
- Today’s space craft navigation systems operate with an error of about 600 meters.


Picture from: http://perso.telecom-paristech.fr/~tupin/JTELE/PRES10/Spigai.pdf

Goal



During the braking phase of a space probe mission (from 2 km/s to 5 m/s), detect in an image as many craters as possible.

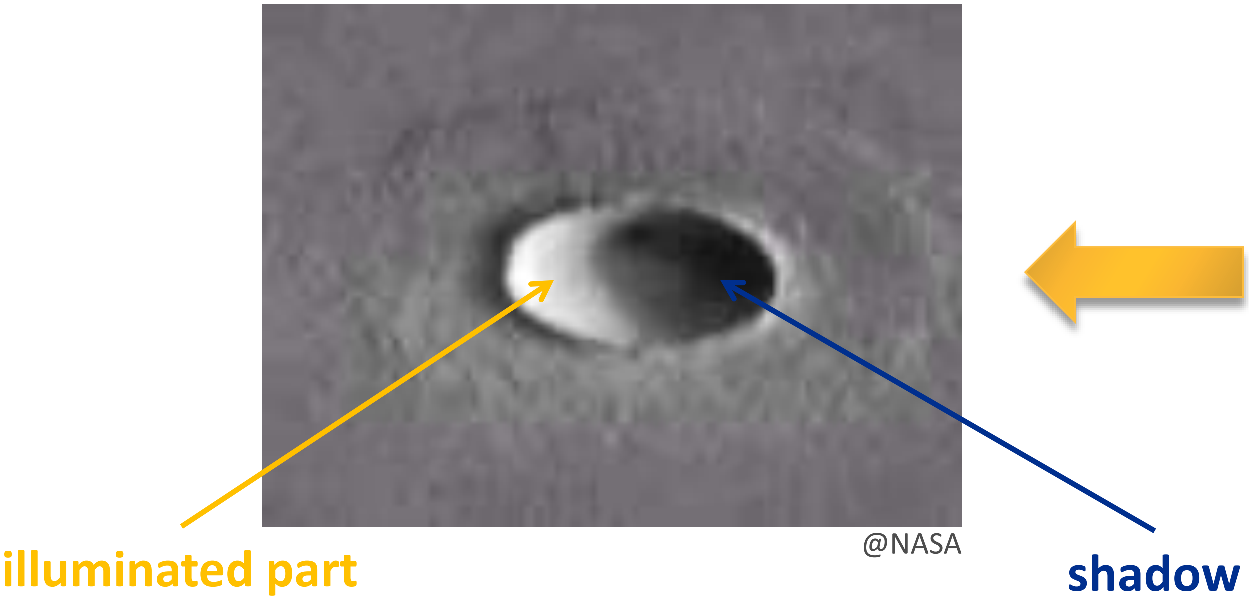
Due to the real time requirements, the algorithm should be relatively simple.

The algorithm cannot rely on any “a priori” information concerning potential crater positions.

Picture from: http://perso.telecom-paristech.fr/~tupin/JTELE/PRES10/Spigai.pdf

Main Idea

The image of a typical crater consists of a shadow and an illuminated part.



Proposed Solution

Image

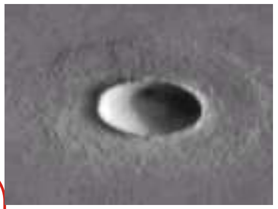
- Basic Image Processing Algorithms
- Done by hardware

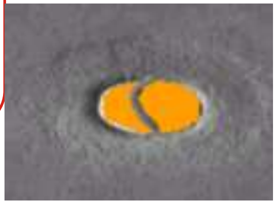
Treatment

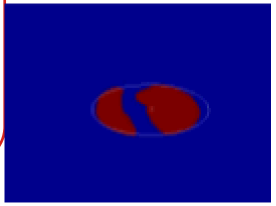
- Segmentation Algorithms: K-Means, Eigenvalue
- Done by well known companies like NASA or ESA
- Output: Matrix showing areas of different brightness

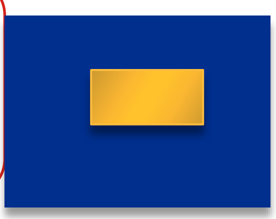
Detection

- Crater Detection Algorithm
- Done by you
- Output: rectangle enclosing the crater

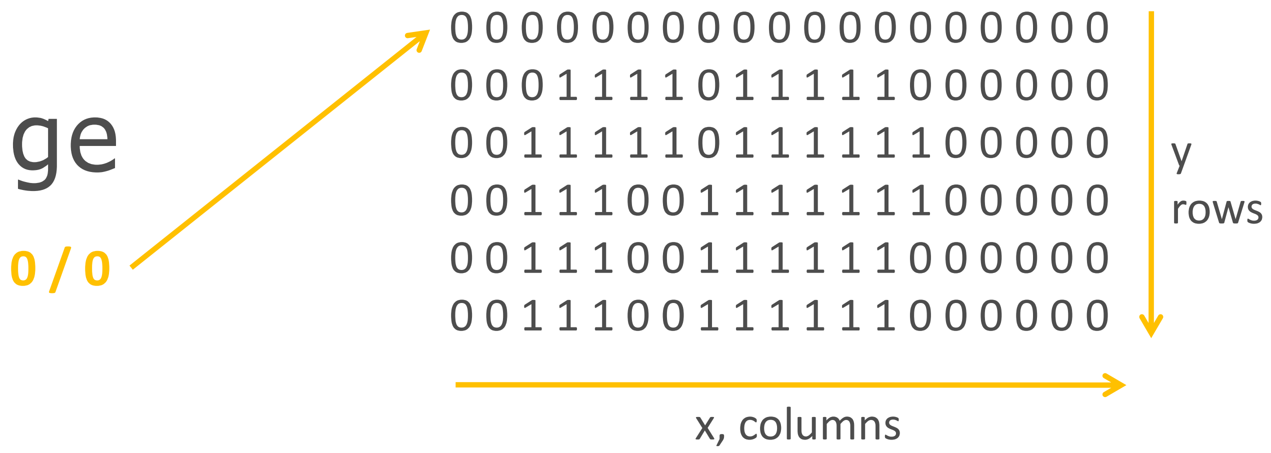








2D Image



```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0
0 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0
0 0 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 0
0 0 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0
0 0 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0
```

For better readability, we show the image data for the examples on the slides with line feeds at the end of each row (see picture above).

In the test cases, there are no line feeds (see picture below).
The first two numbers in the image input data are about the image size with nrOfRows and nrOfColumns.

```
12 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 0 0 ....
```

Level 1:
Find the enclosing rectangle

Enclose the detected crater with a rectangle.

Input: nrOfRows nrOfColumns data

nrOfRows = number of rows of the 2D image

nrOfColumns = number of columns of the 2D image

data = list of numbers (with nrOfRows x nrOfColumns elements)

Possible values for data:

- “1” = pixel belongs to crater (either to the illuminated or the shadow part)
- “0” = pixel does not belong to crater

Output: x y deltaX deltaY

Important Assumption

The illuminated part and the shadow part both extend from the bottom of the rectangle to the top of the rectangle.

They cannot differ in their height.

Example
(see also next page)

Input: 12 19 0 0 0 0

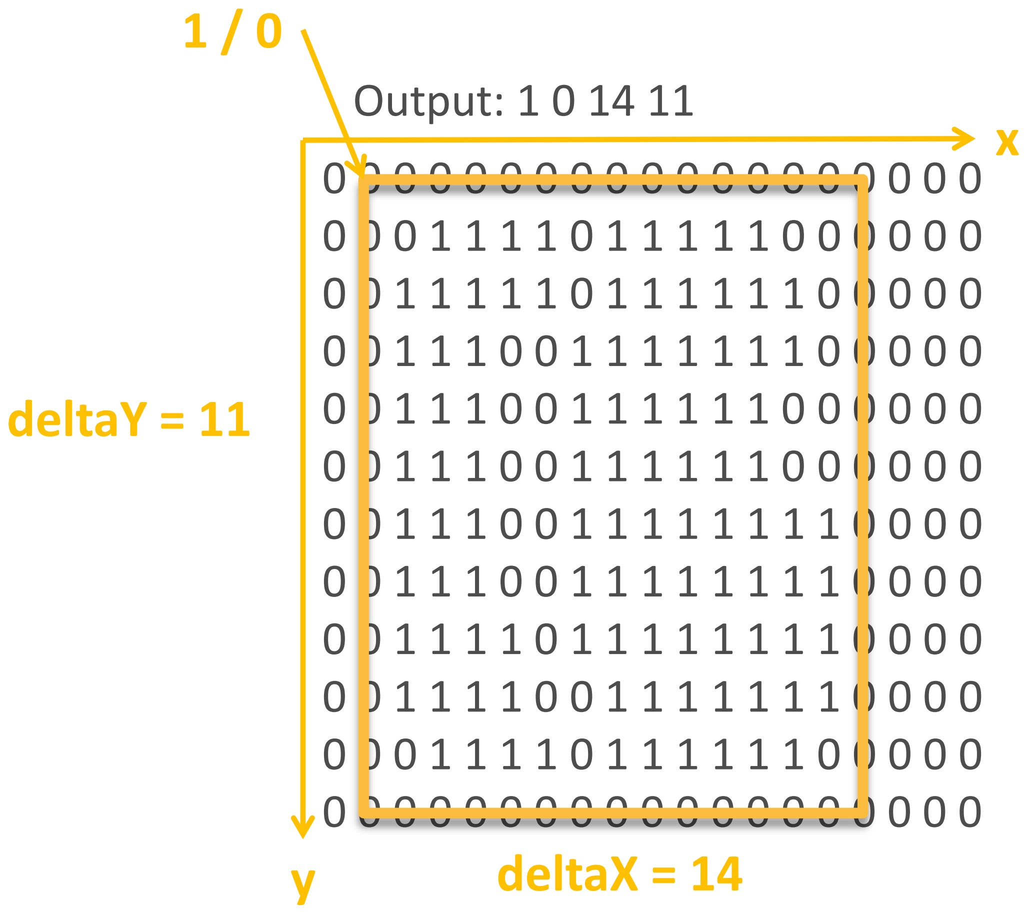
```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0
0 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0
0 0 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 0
0 0 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0
0 0 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0
0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0
0 0 0 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Output: 1 0 14 11

Example: Output explained

Input: 12 19 0 0 0 0 ….

Output: 1 0 14 11



```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0
0 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0
0 0 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 0
0 0 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0
0 0 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0
0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 0
0 0 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0
0 0 0 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```