To Infinity And Beyond

Computer Vision for Astronomy

Ryan Fox

ryan@foxrow.com

@ryan_fox

foxrow.com

1. Image Processing
2. Computer Vision
3. To Infinity and Beyond

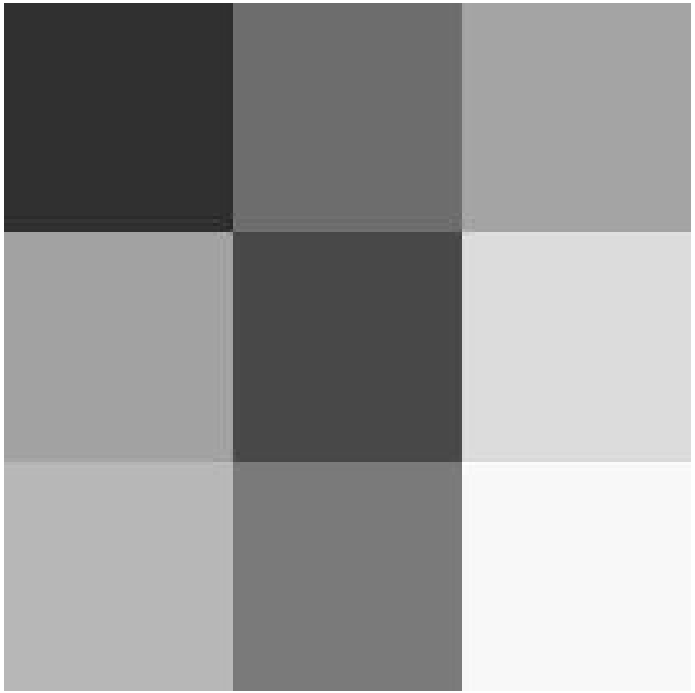
How computers see

How computers see

006	094	142
166	052	201
179	121	249

How computers see

006	094	142
166	052	201
179	121	249



How computers see

006	000	000
166	000	000
179	000	000
000	094	000
000	052	000
000	121	000
000	000	142
000	000	201
000	000	249

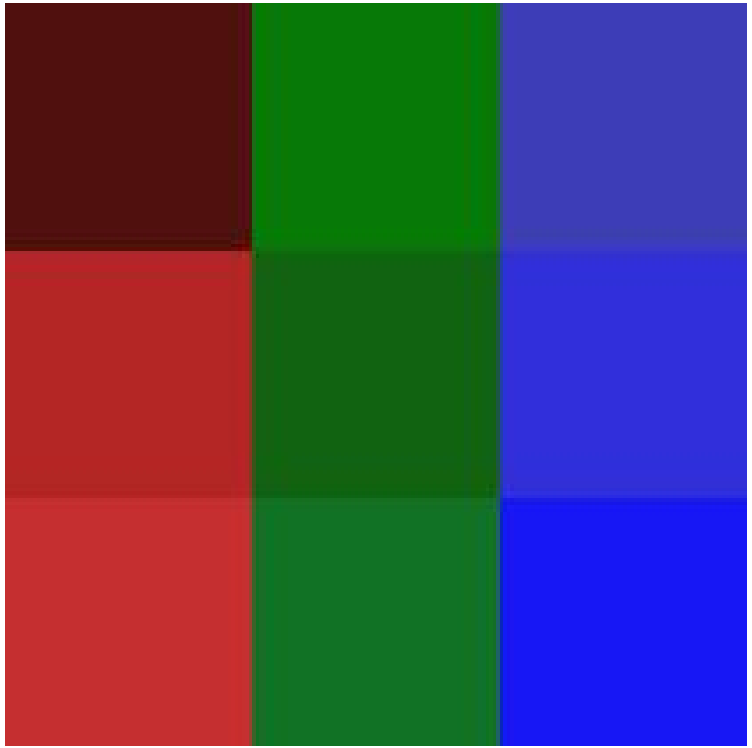


Image Processing vs Computer Vision

Image processing

Low level algorithms

Works on individual pixels

Building blocks for larger applications

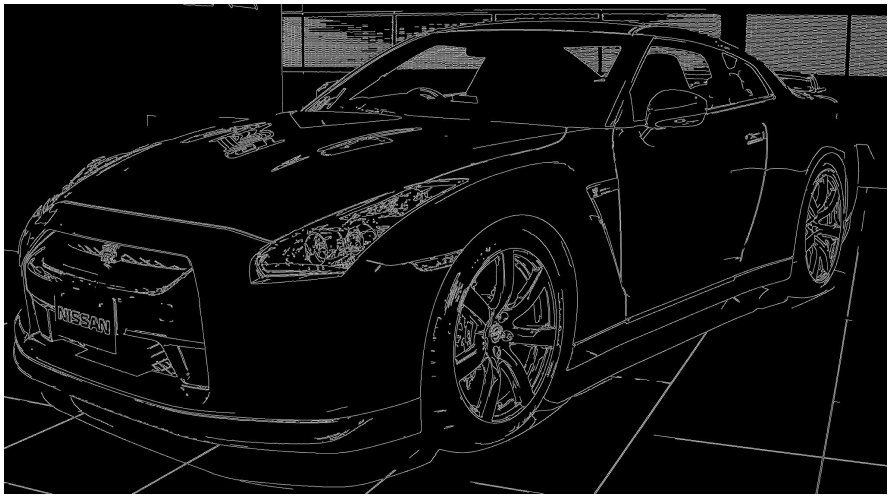
Computer Vision

Higher level of abstraction

Works on whole images or image sets

Understand image contents

Image processing



Computer Vision

Use OpenCV

http://opencv.org


```
$ pip install opencv-python
```

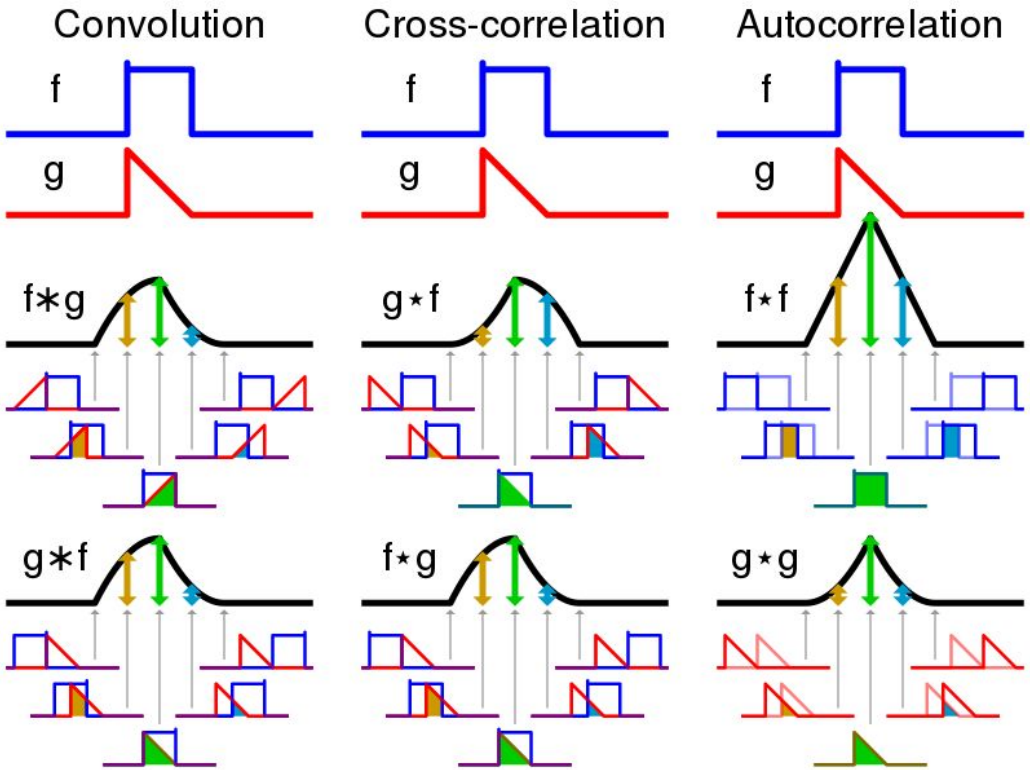
Use OpenCV

```
>>> import cv2

>>> img = cv2.imread('my_image.jpg')

...

>>> cv2.imwrite('new_image.jpg', img)
```

Image Processing

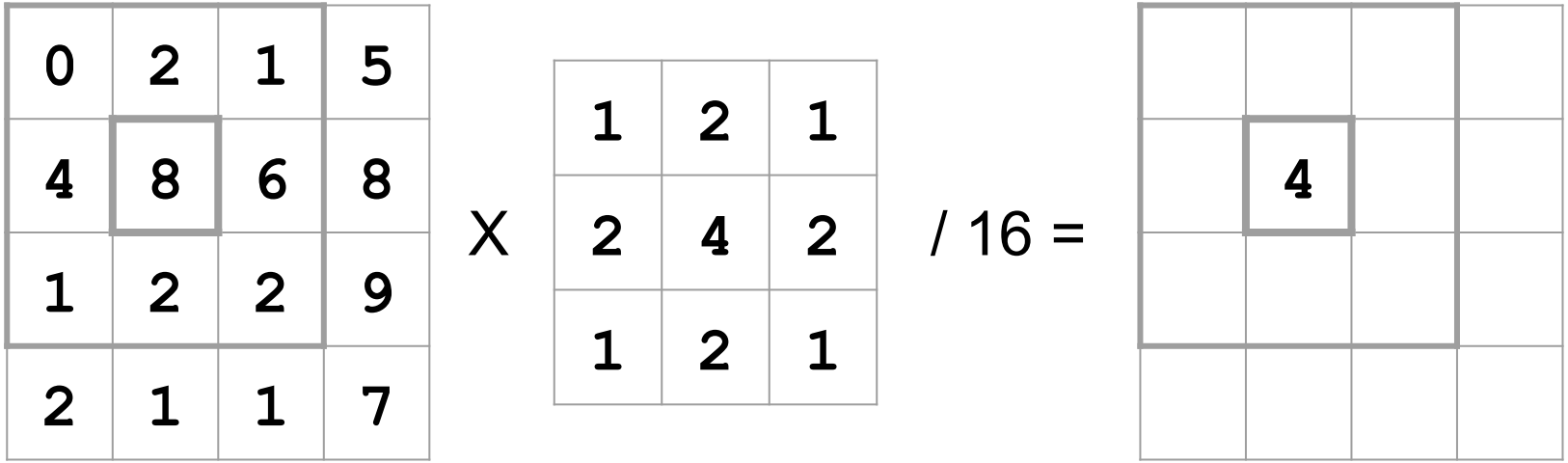
- Convolution
- Feature extraction
- Feature descriptors
- Image Segmentation
- Thresholding
- Erosion/Dilation
- Contours
- Histograms
- Image registration
- Panoramas
- Stacking

Convolution



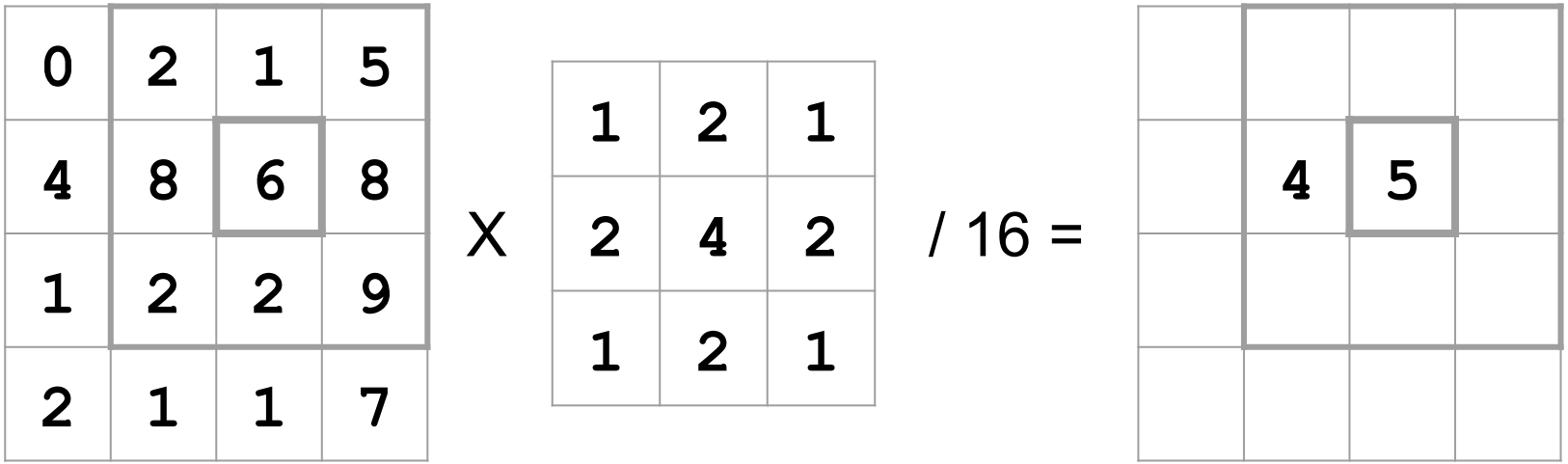
https://commons.wikimedia.org/wiki/File:Comparison_convolution_correlation.svg

Convolution



(0*1 + 2*2 + 1*1 + 4*2 + 8*4 + 6*2 + 1*1 + 2*2 + 2*1)/16 = 4

Convolution



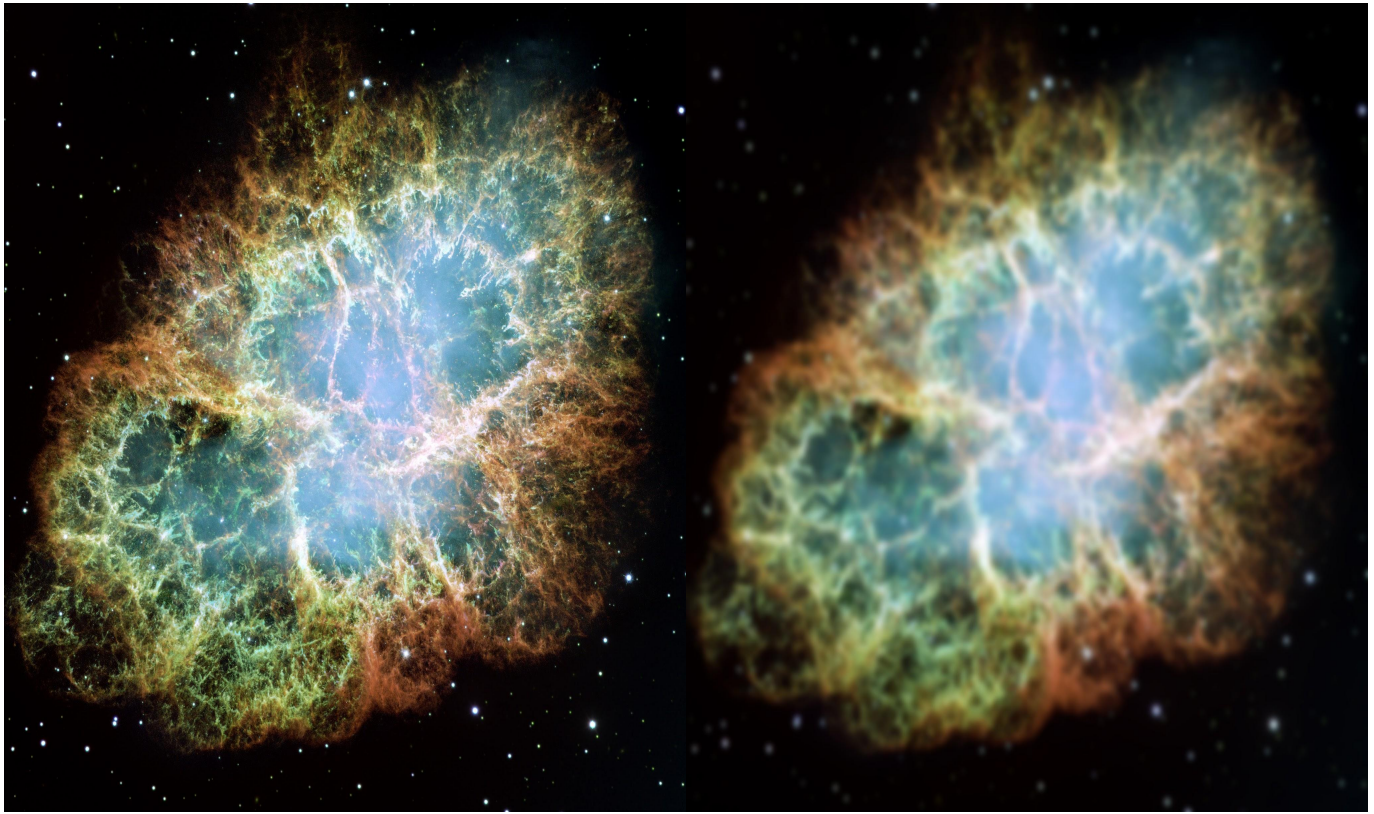
(2*1 + 1*2 + 5*1 + 8*2 + 6*4 + 8*2 + 2*1 + 2*2 + 9*1)/16 = 5

Convolution

Blur

Kernel:

1	2	1
2	4	2
1	2	1



Feature Extraction

Edge Detection

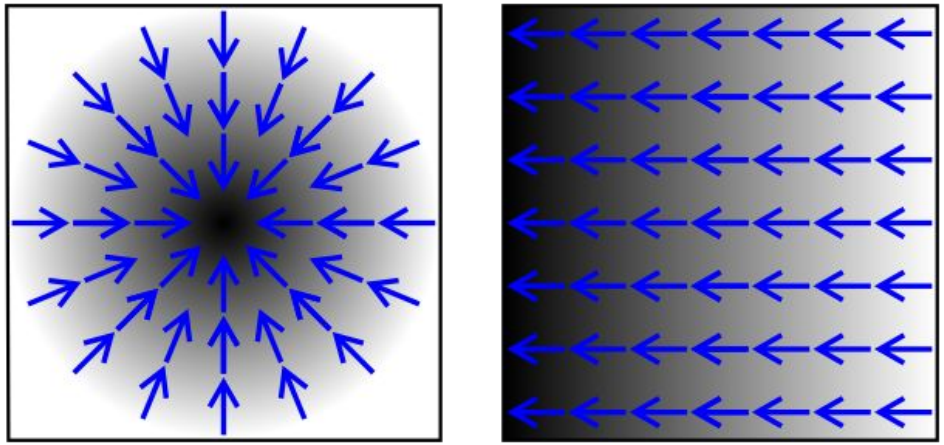
Corner Detection

Hough Transform

Feature Descriptors

Canny Edge Detector

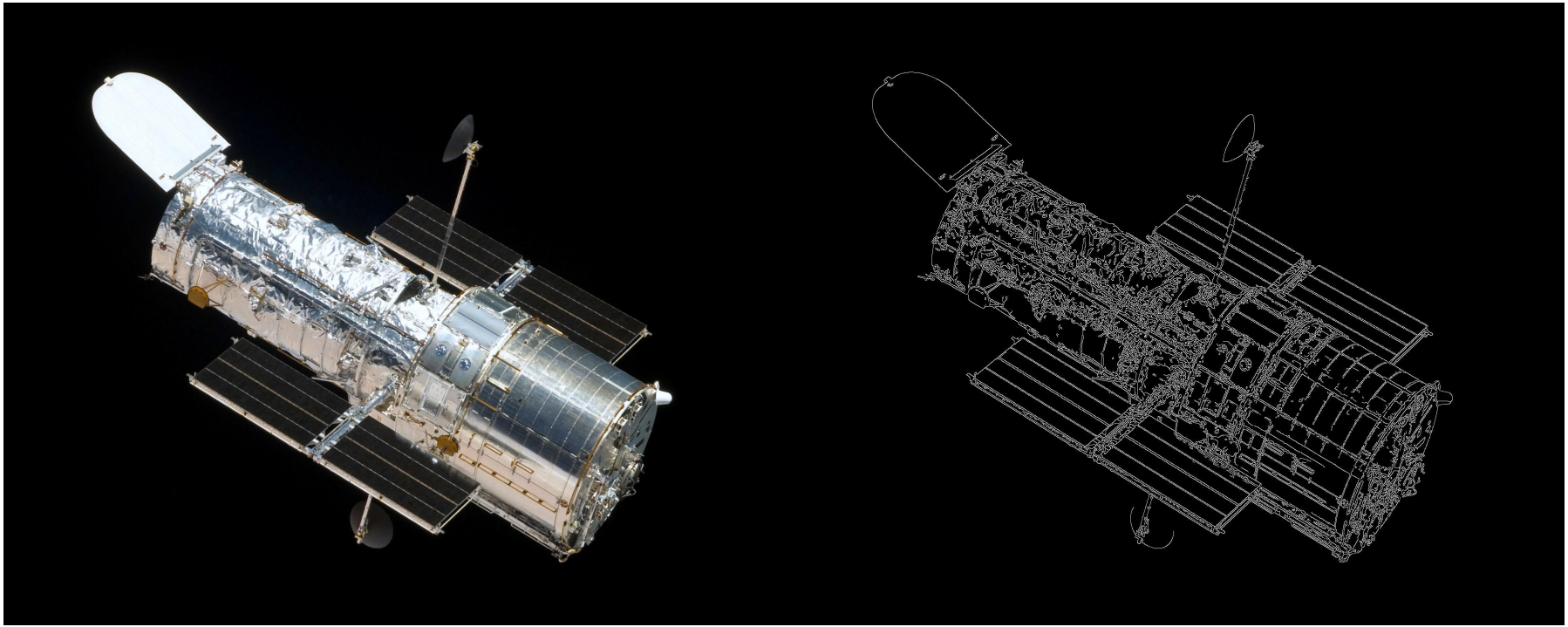
1. Blur the image
2. Calculate gradient
3. Places with strong gradients are likely to be edges



https://commons.wikimedia.org/wiki/File:Gradient2.svg

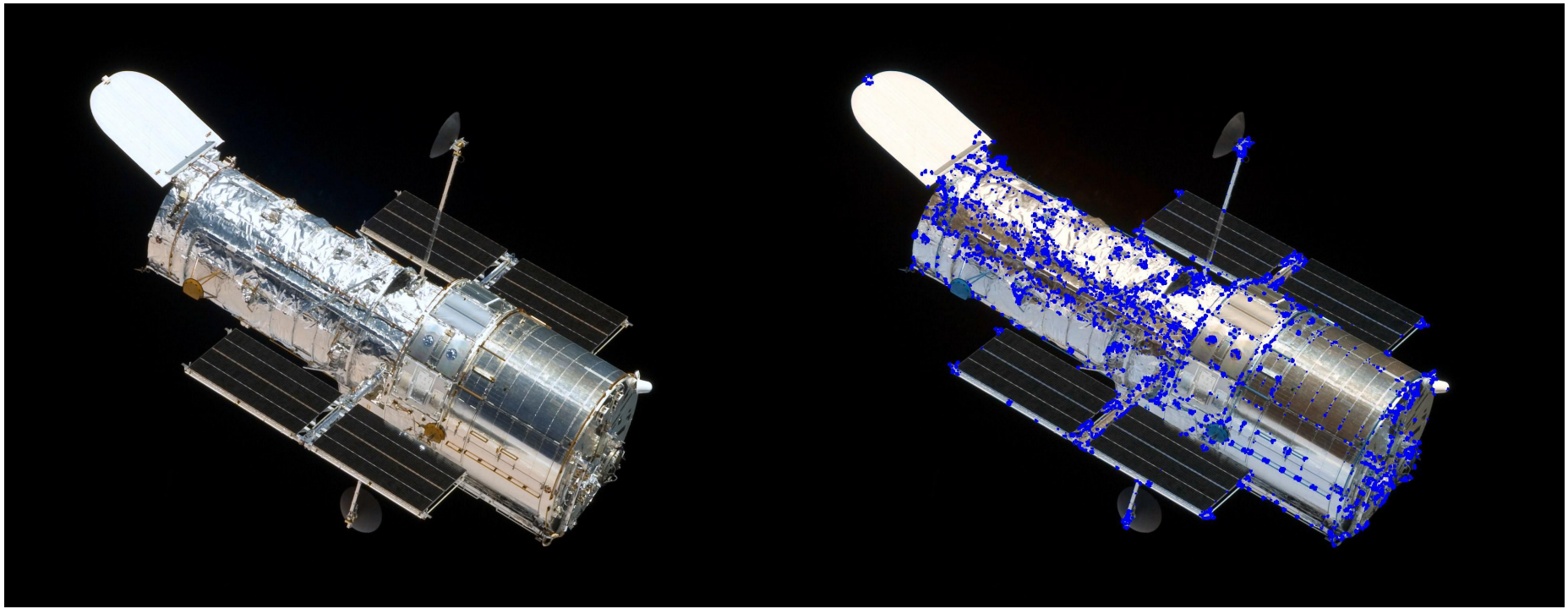
Canny Edge Detector

`cv2.Canny()`



Harris Corner Detector

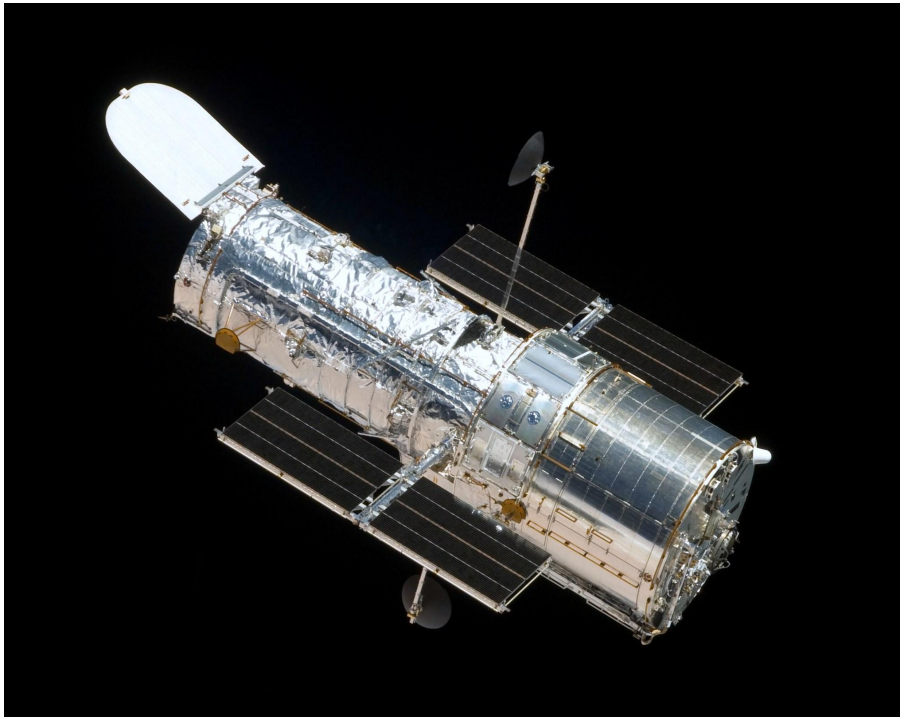
`cv2.cornerHarris()`

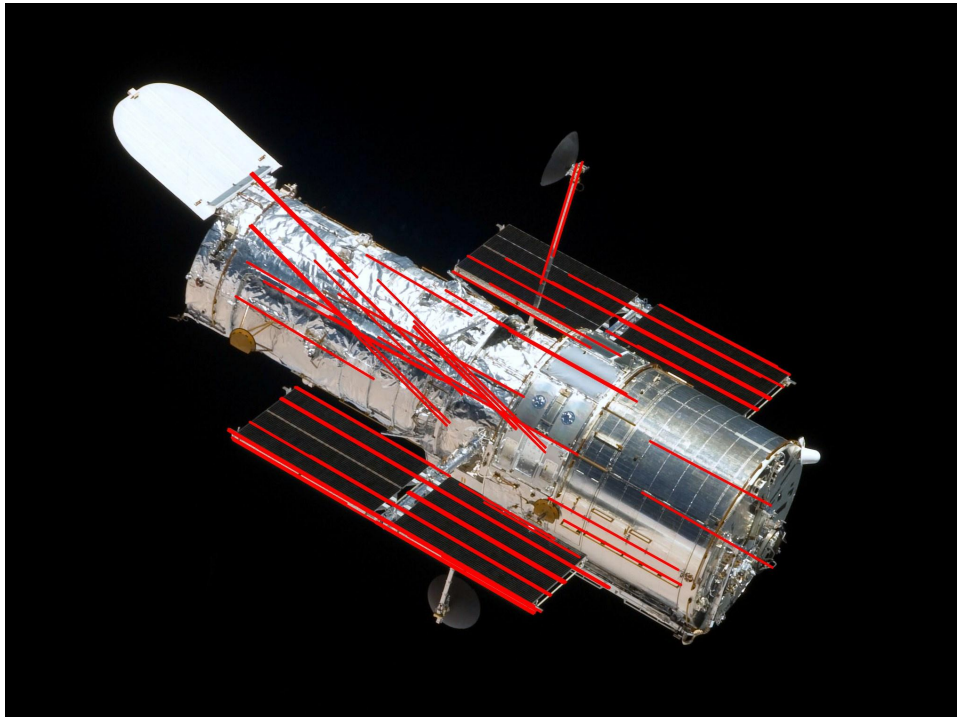


Hough Transform

```
cv2.HoughLines()
cv2.HoughCircles()
```

Hough Line Transform





Feature Descriptors

SIFT and SURF

Scale Invariant Feature Transform

Speeded Up Robust Features

Feature Descriptors

ORB

BRISK

BRIEF

KAZE

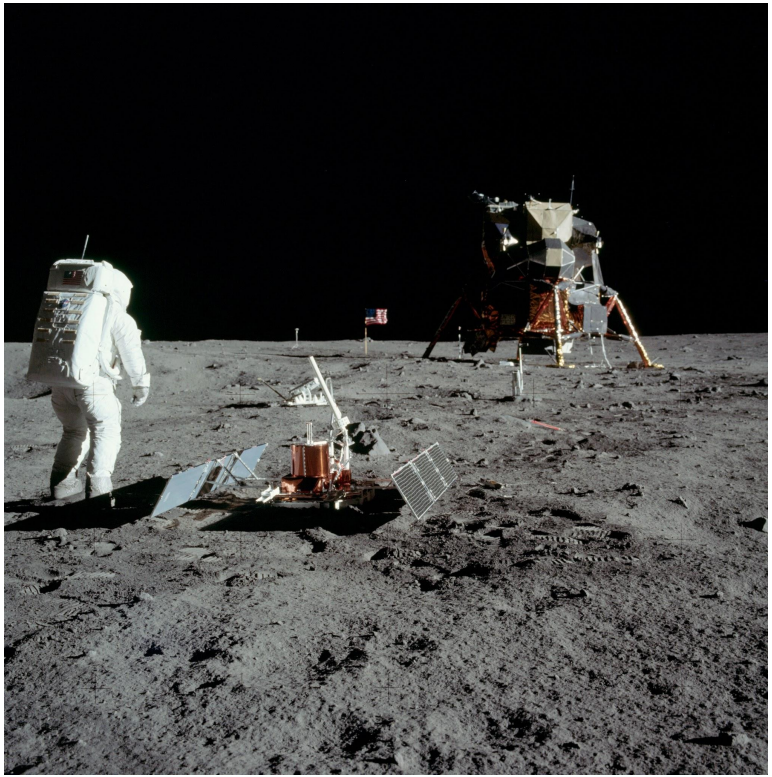
AKAZE

ORB

Oriented FAST and Rotated BRIEF

```
cv2.ORB_create()
```

ORB



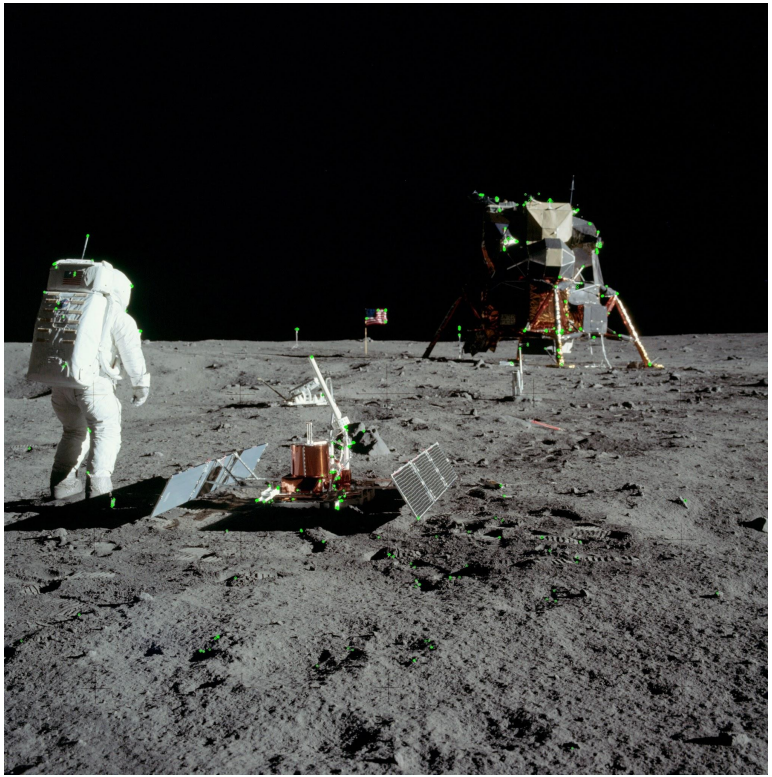
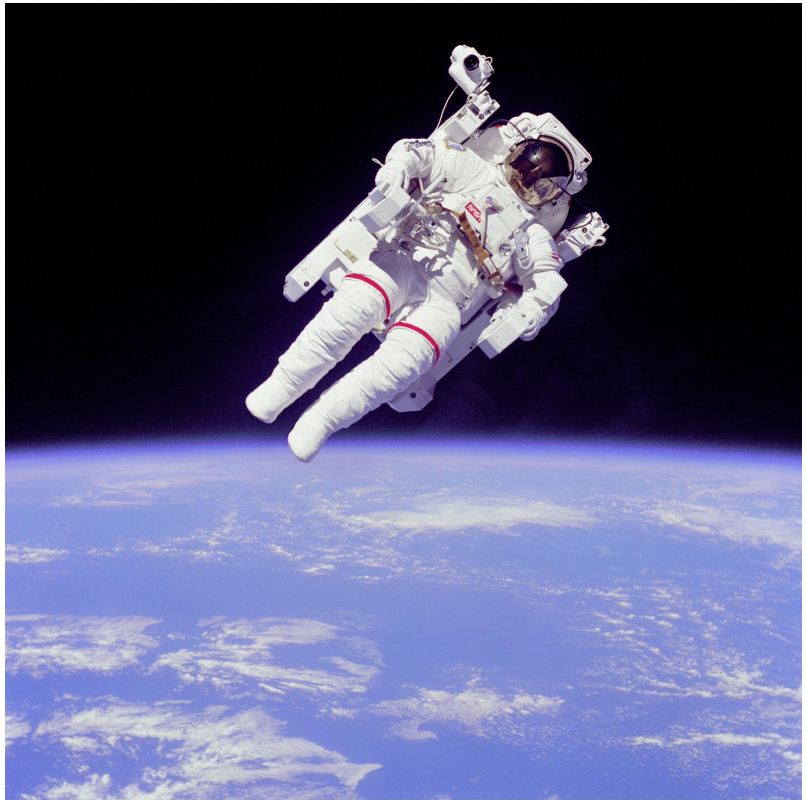


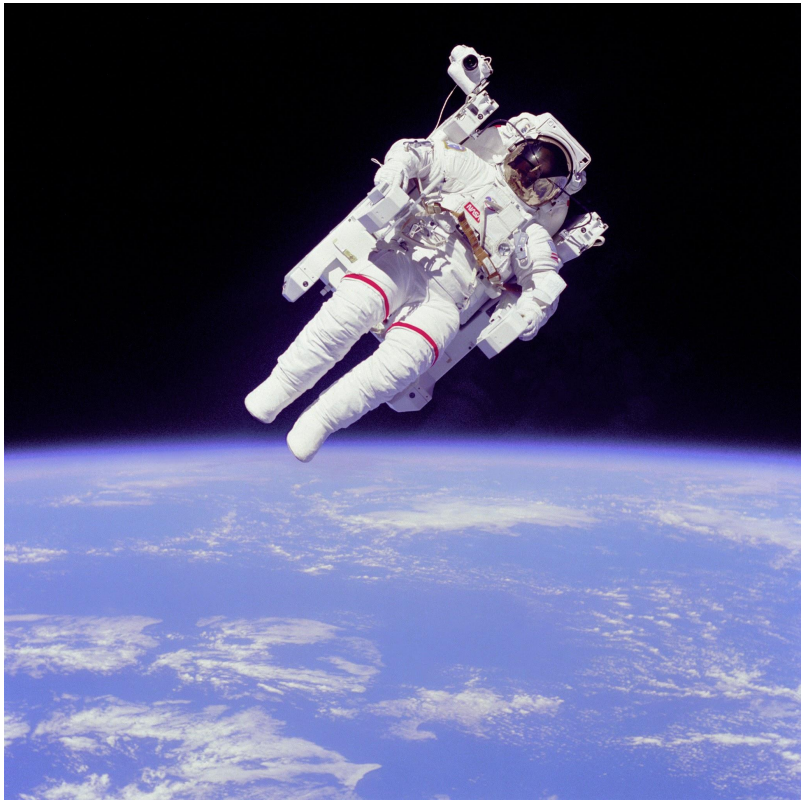
Image Segmentation

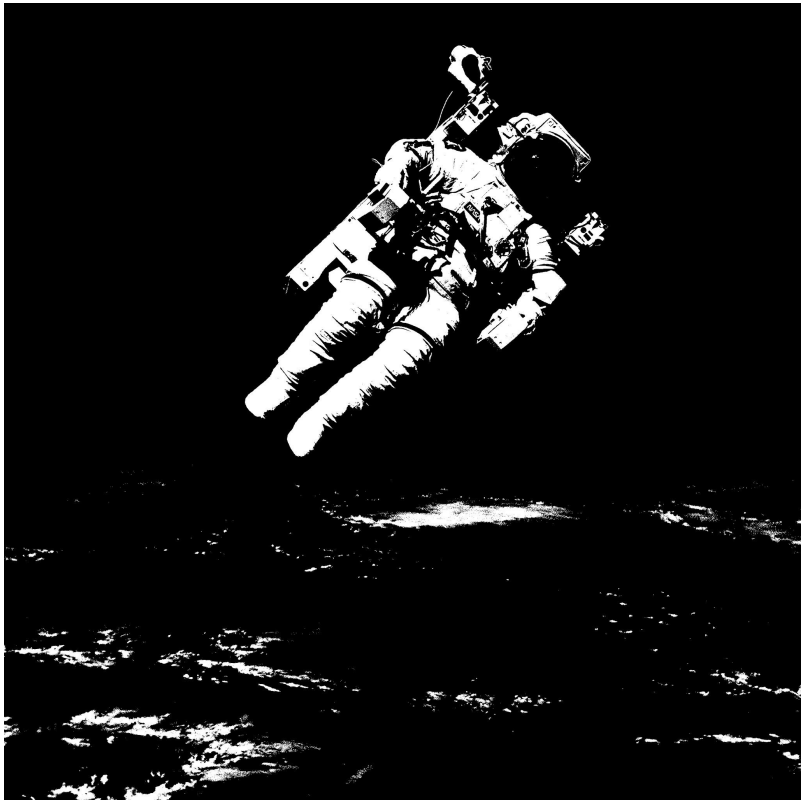
Thresholding

`cv2.threshold()`



Thresholding



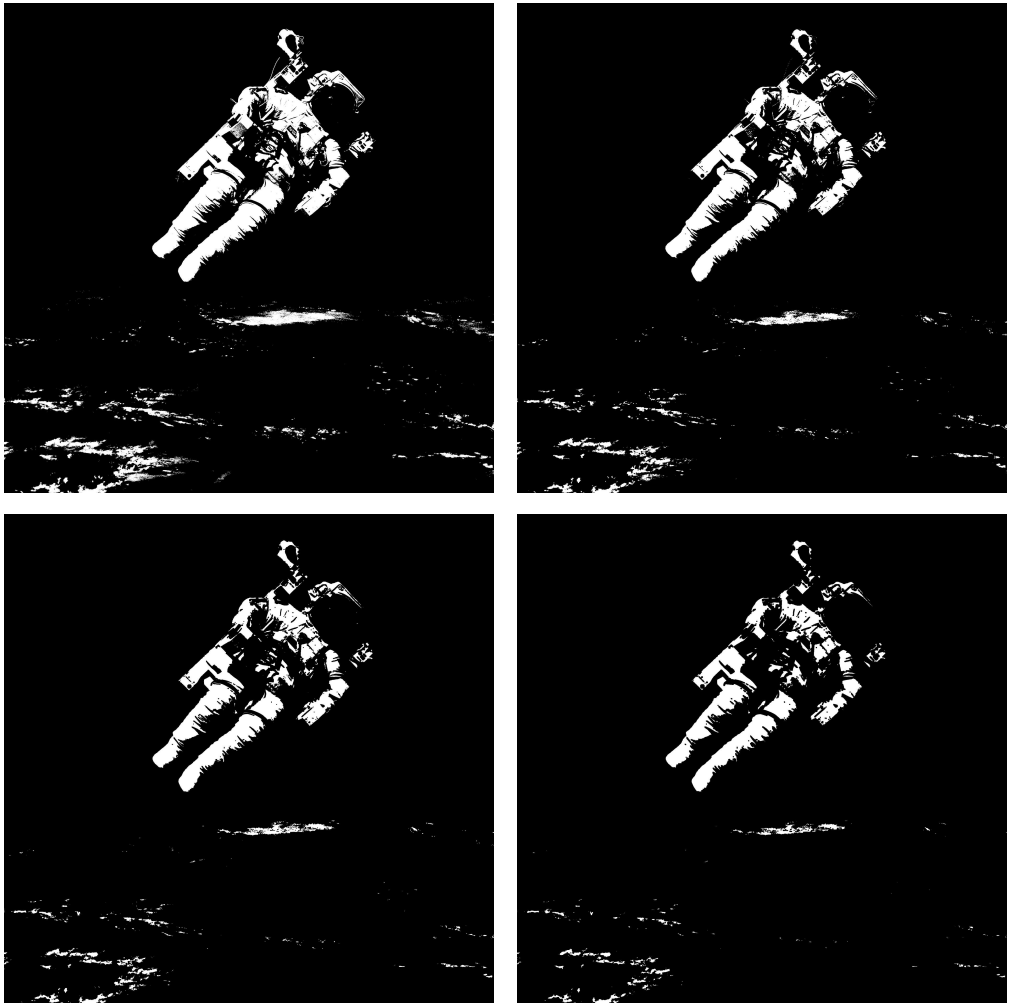


Erosion and Dilation

Grow or shrink
regions in an image

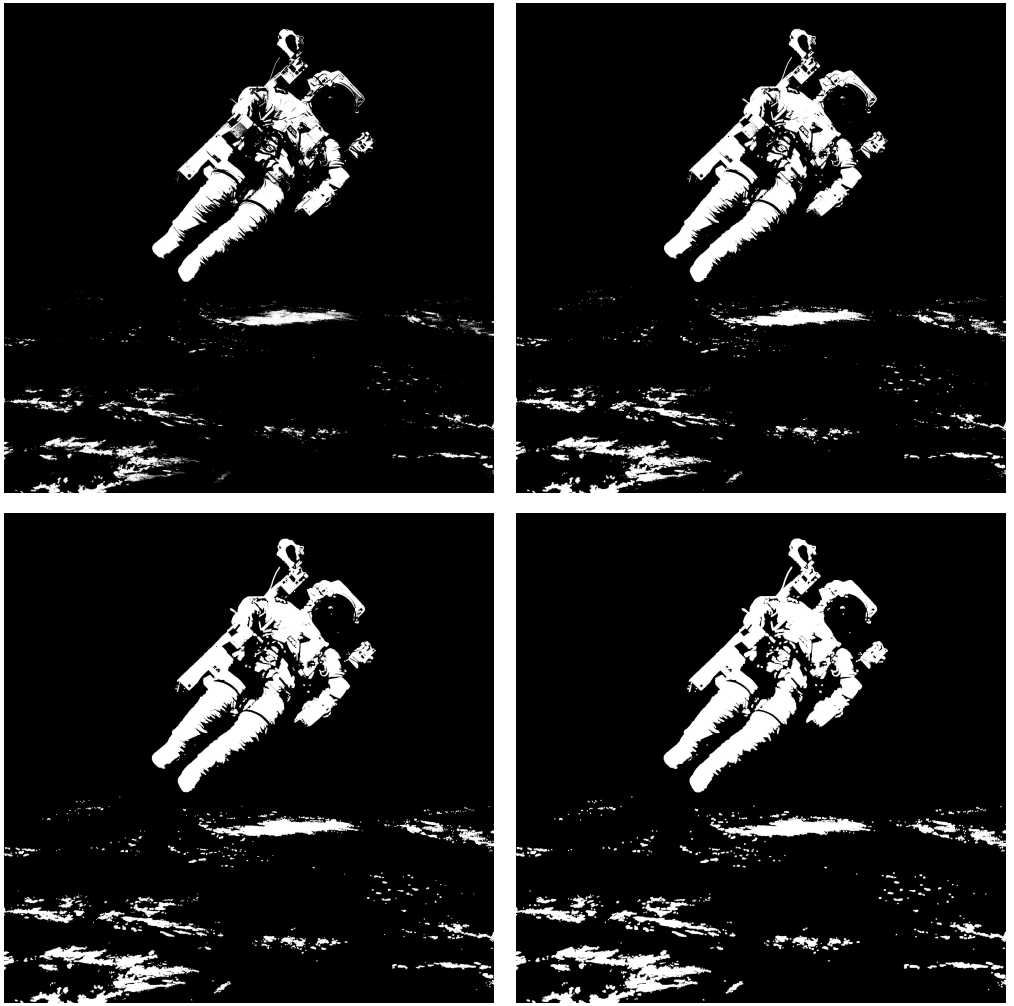
Erosion

`cv2.erode()`



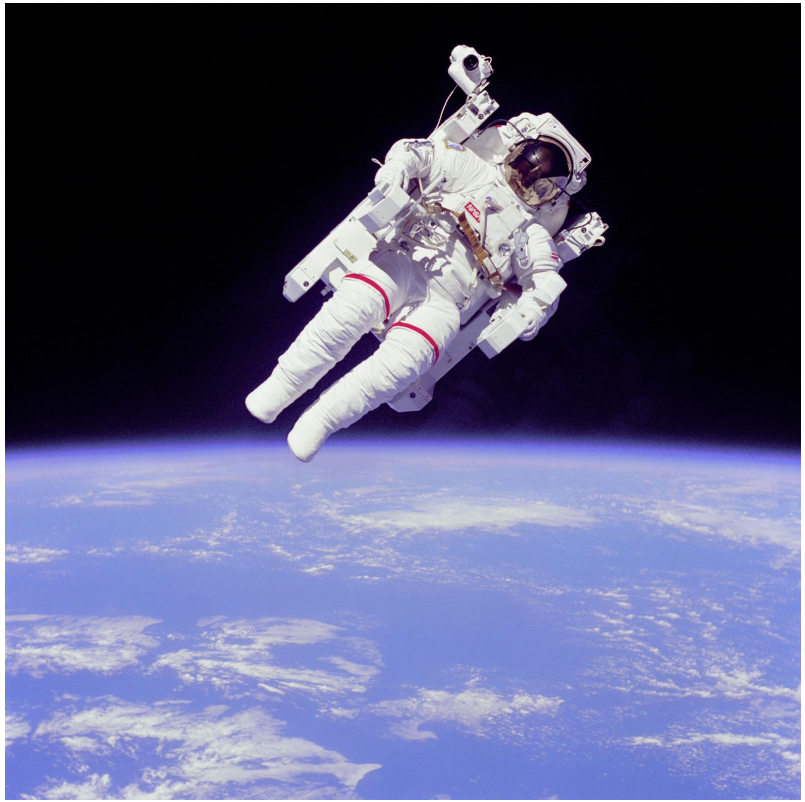
Dilation

`cv2.dilate()`

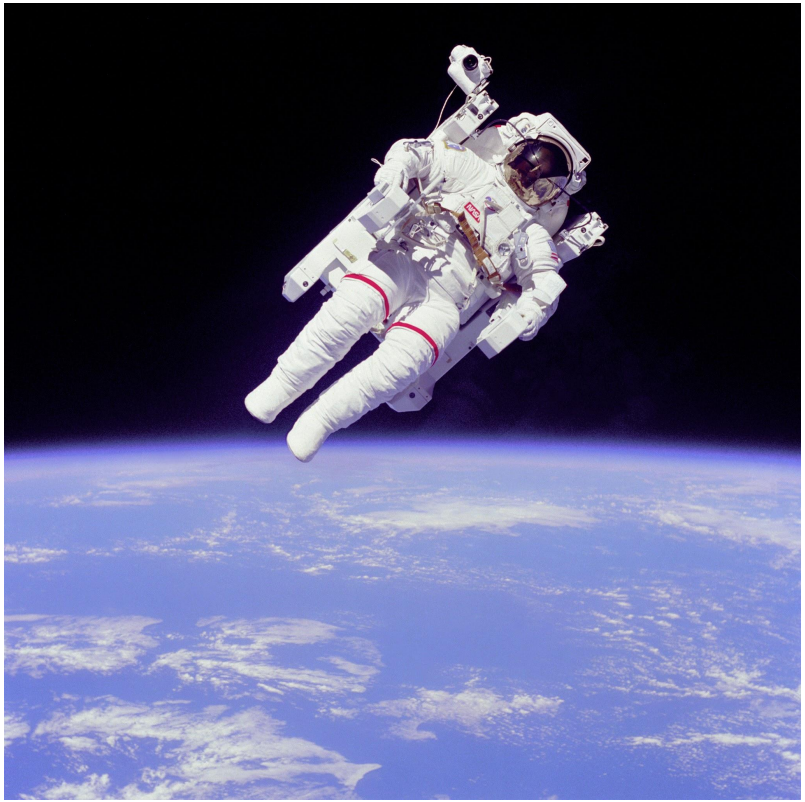


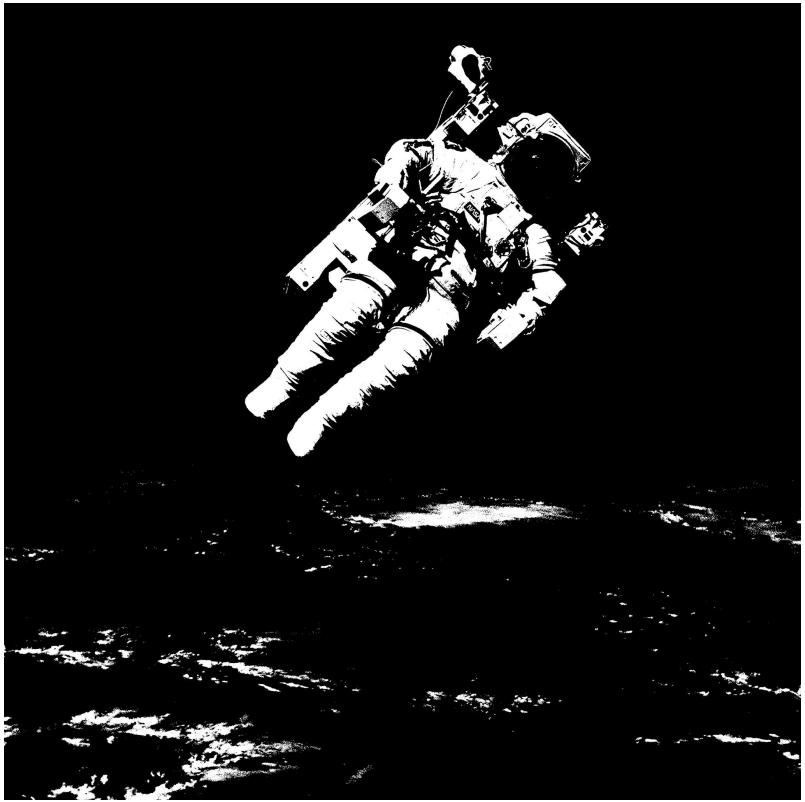
Contours

`cv2.findContours()`

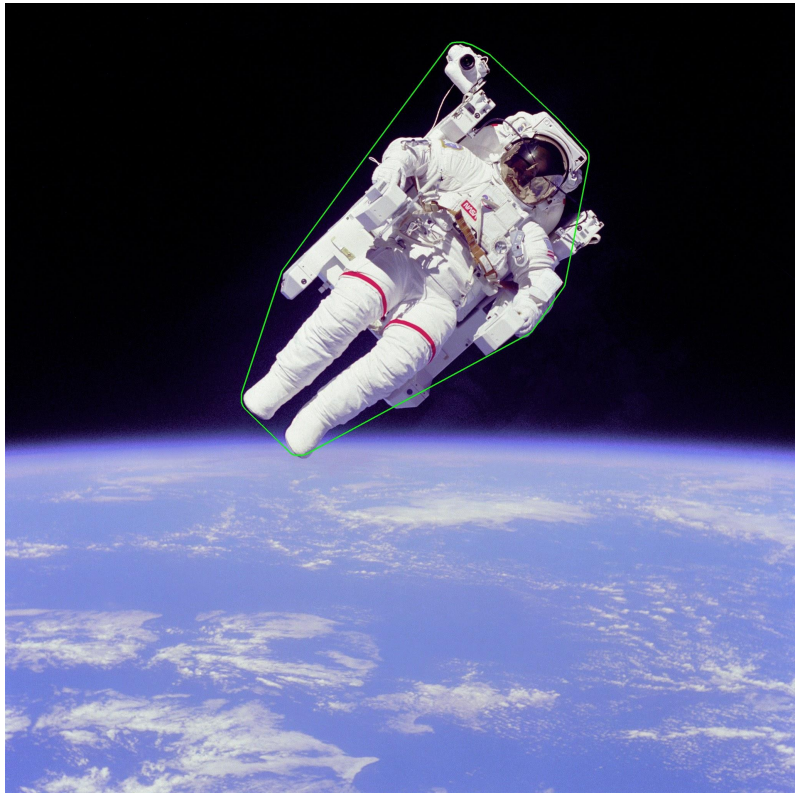


Contours





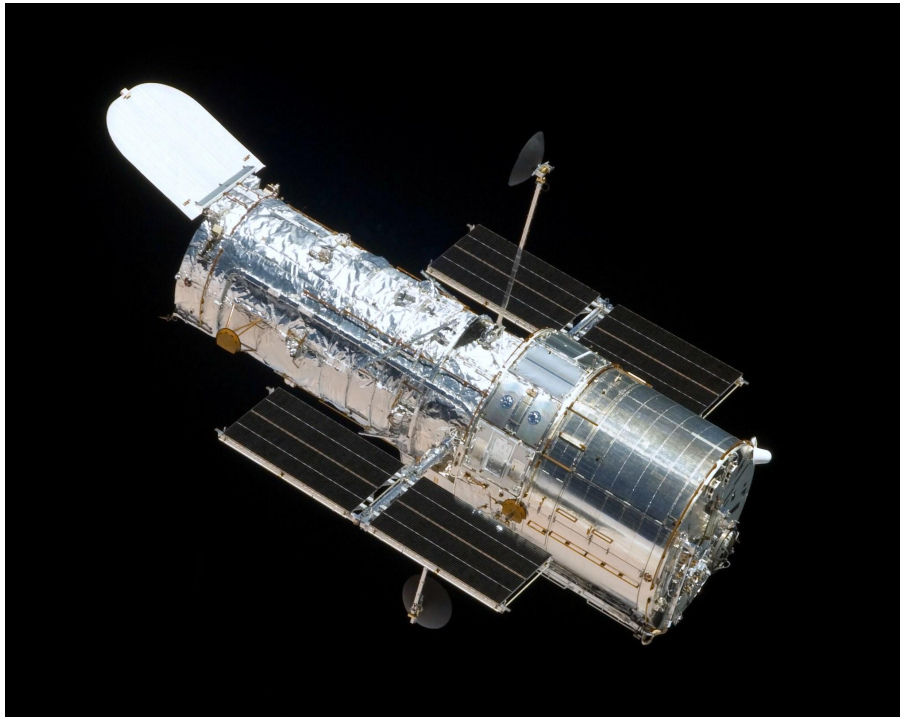
Contours

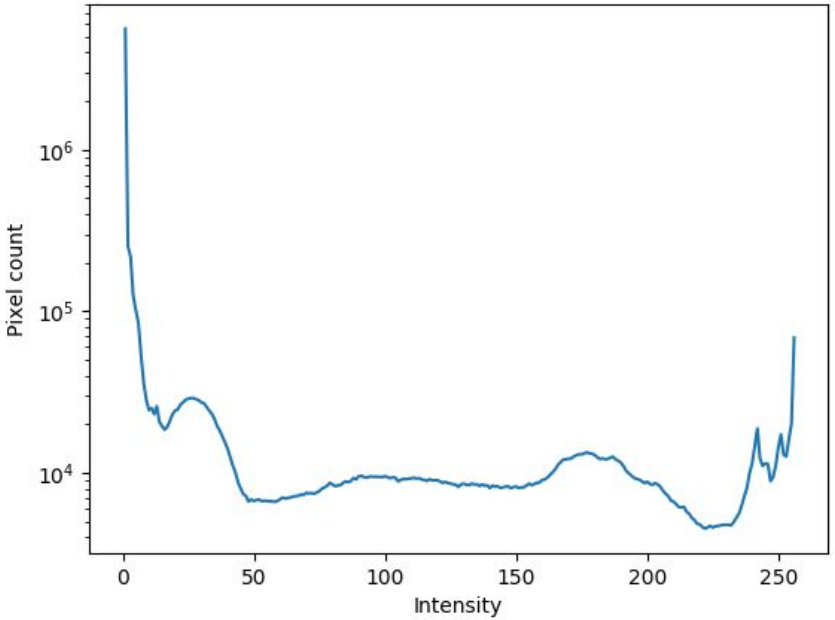


Histograms

Histograms

`numpy.histogram()`



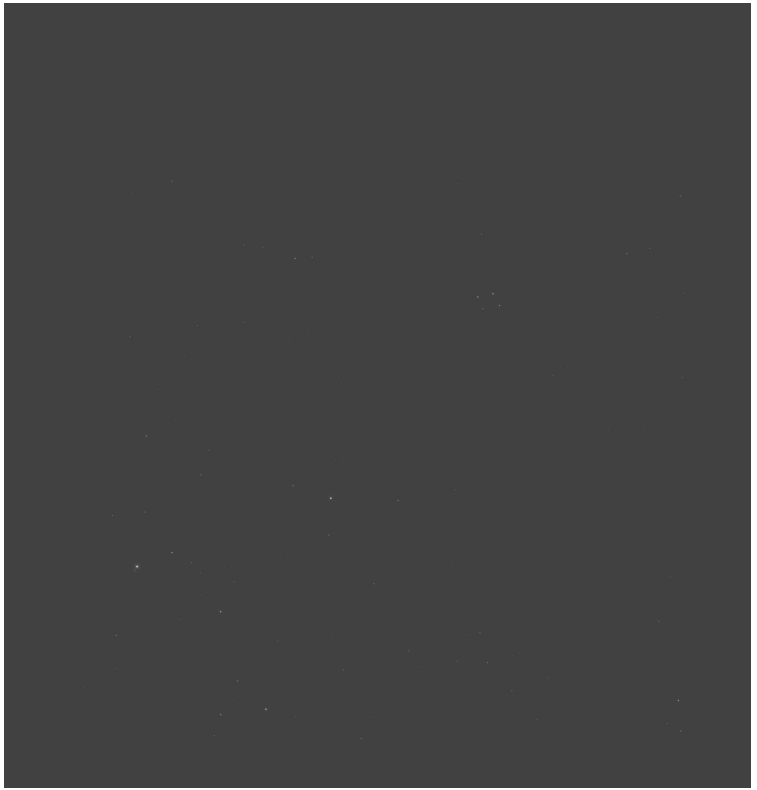


Histogram Equalization

This is not a typical Hubble image

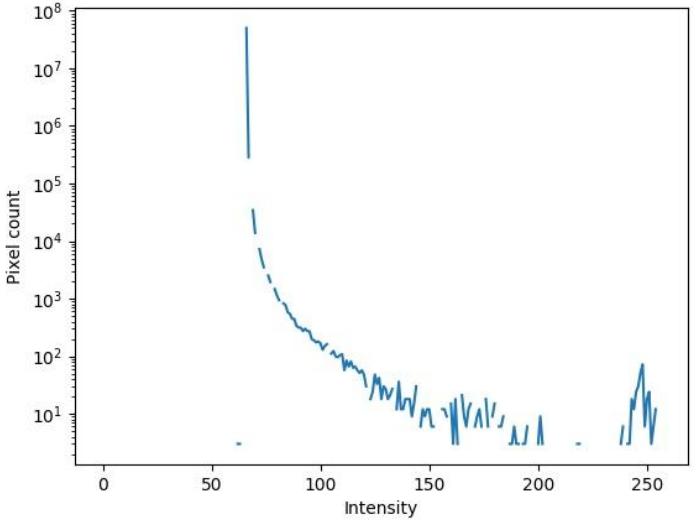
Histogram Equalization

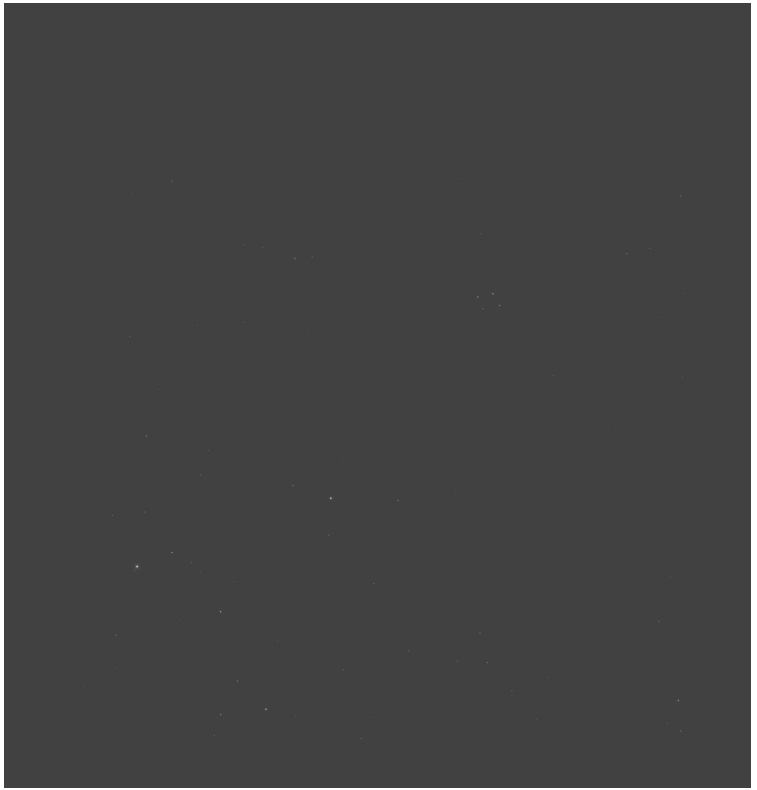
This is a typical Hubble image



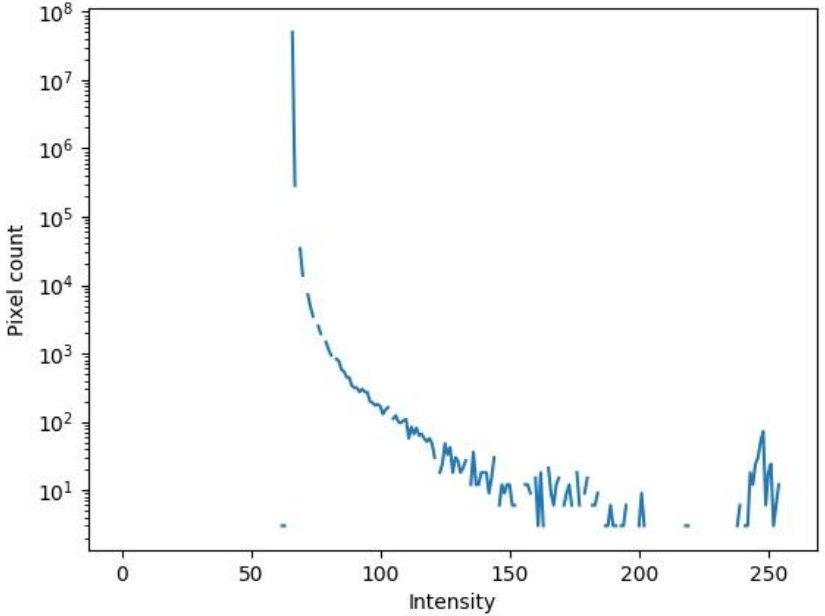
Histogram Equalization

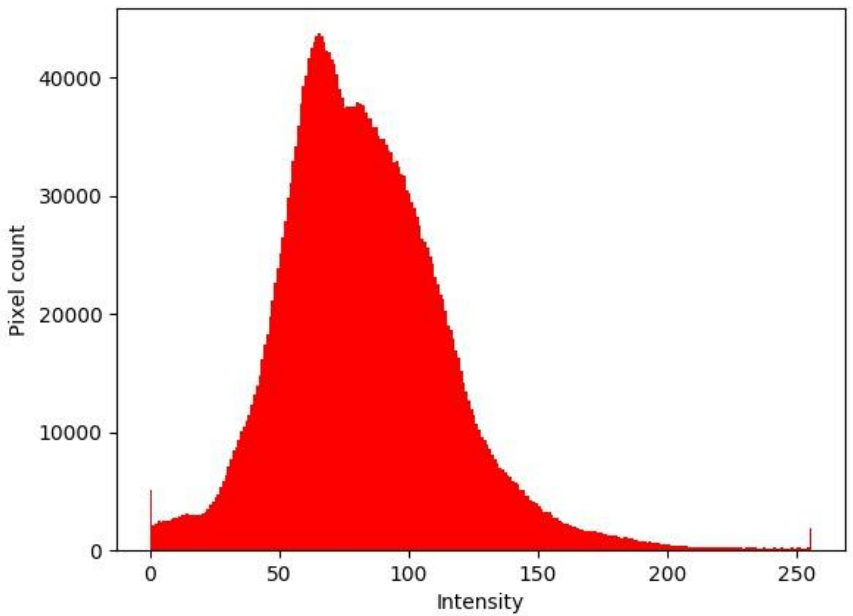
This is a typical Hubble image



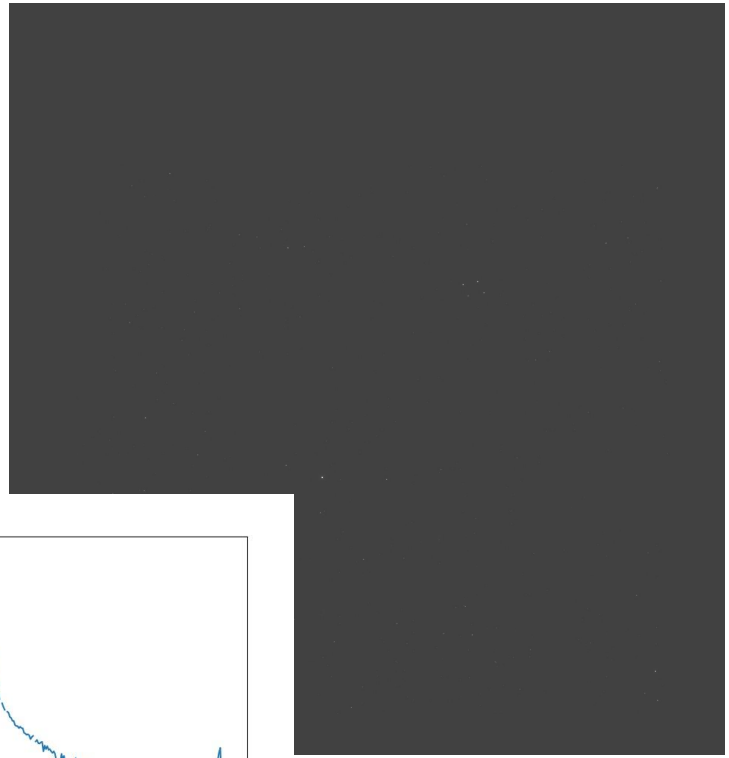


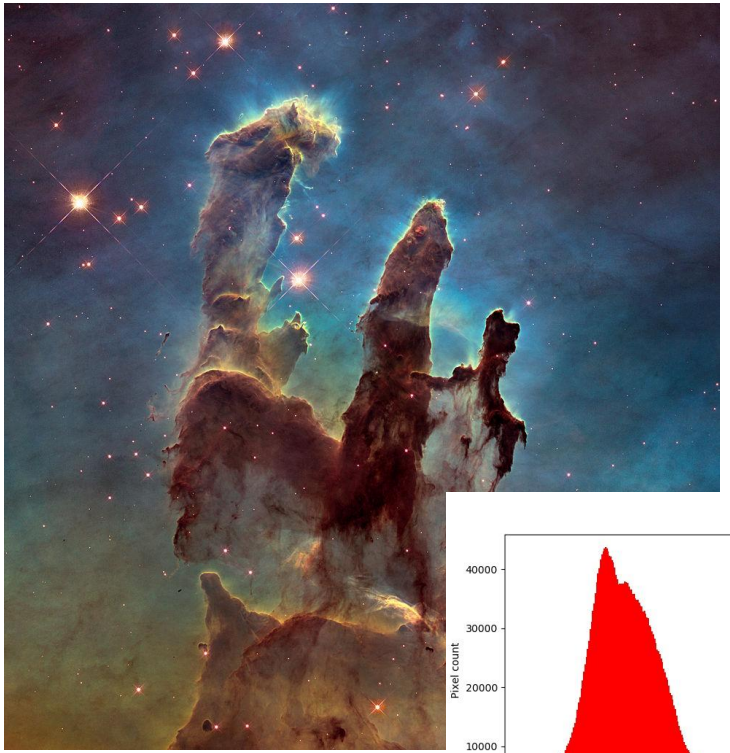
Histogram Equalization

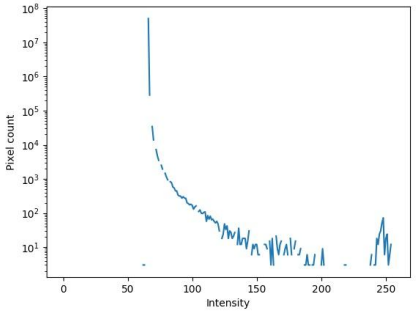




Histogram Equalization







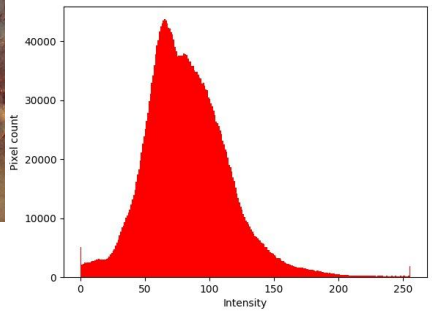


Image Registration

Panoramas

Panoramas

Stacking

Stacking

Sources of noise:

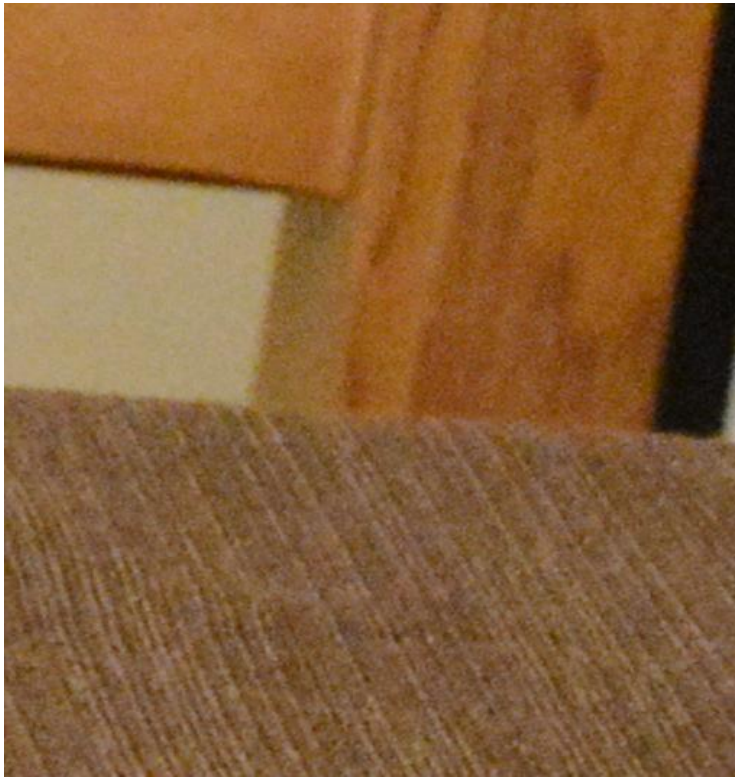
- Cosmic rays
- Atmospheric distortion
- Lens imperfections
- Sensor imperfections
- Thermal noise
- Readout noise
- Clouds
- Airplanes
- Pedestrians
- People blinking

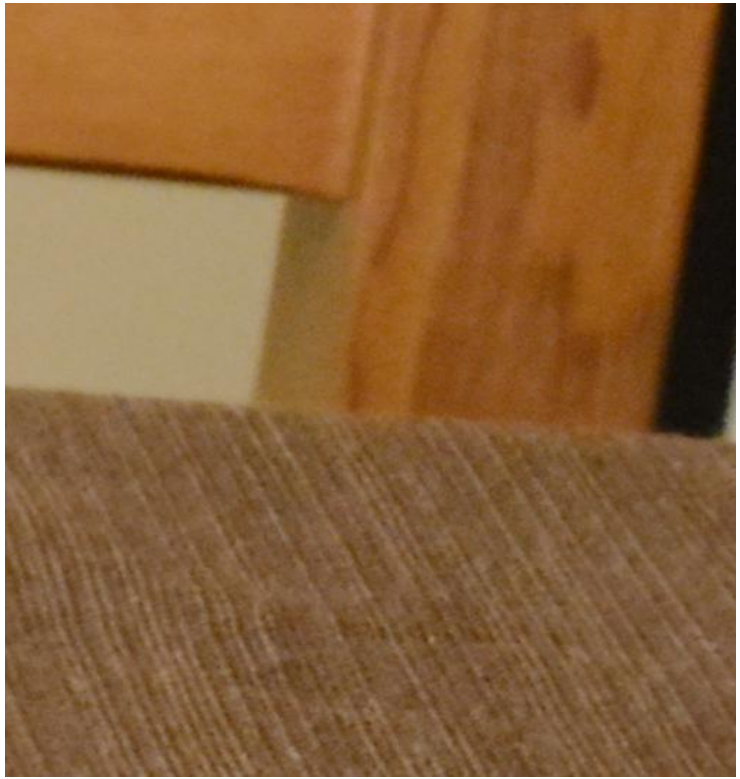
Stacking






Stacking





Computer Vision

- Object detection
- Object recognition
- Face recognition
- Reverse image search
- Duplicate detection
- OCR
- QR codes
- Photogrammetry
- Neural nets

Object Detection

Feature matching

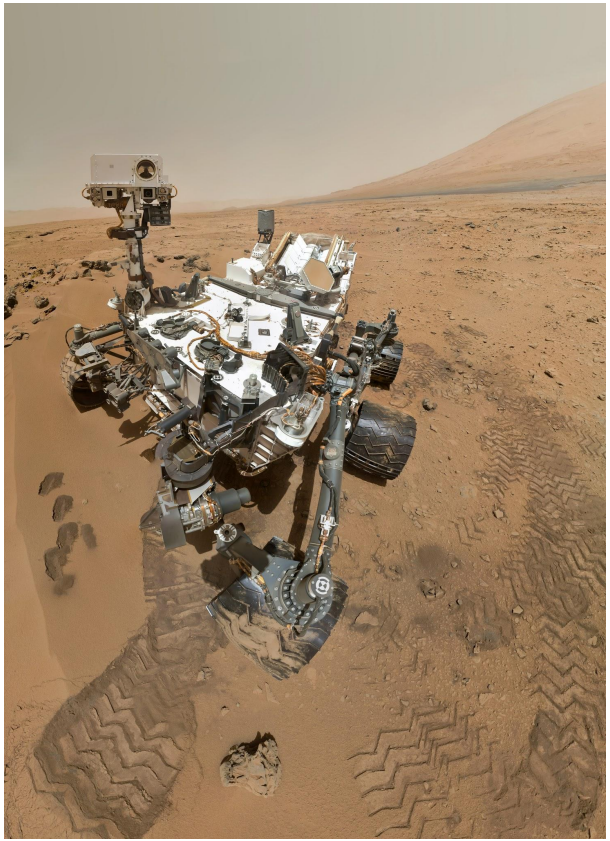
Histogram backprojection

Haar cascades*

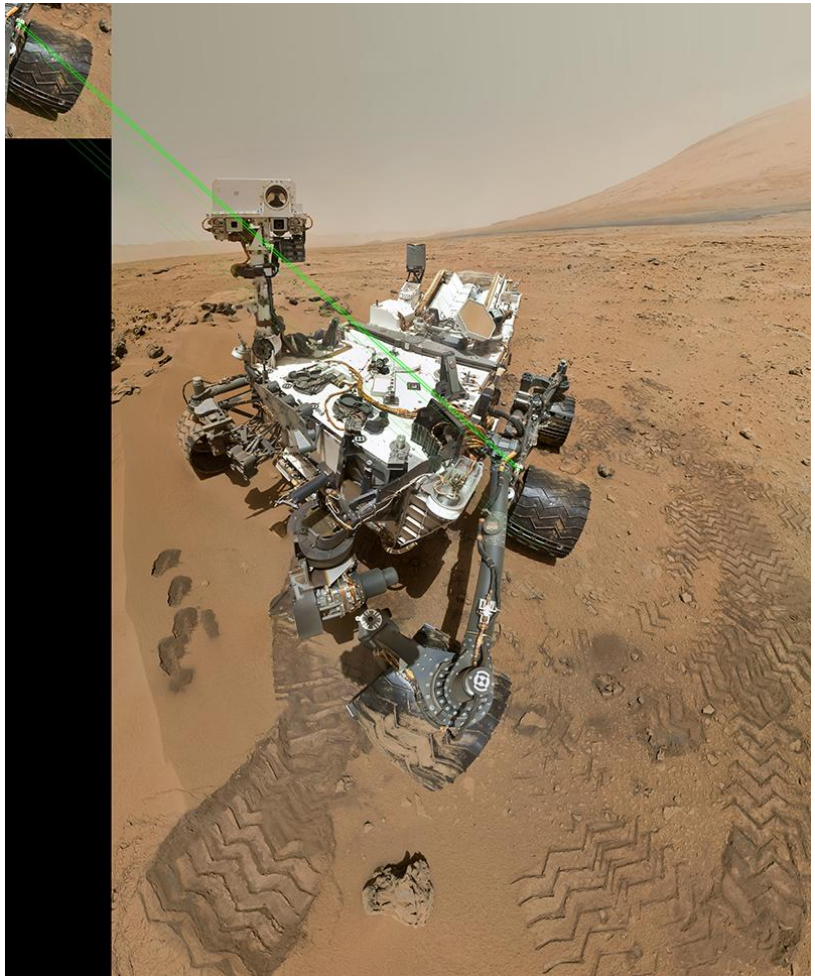
Neural nets*

Feature Matching

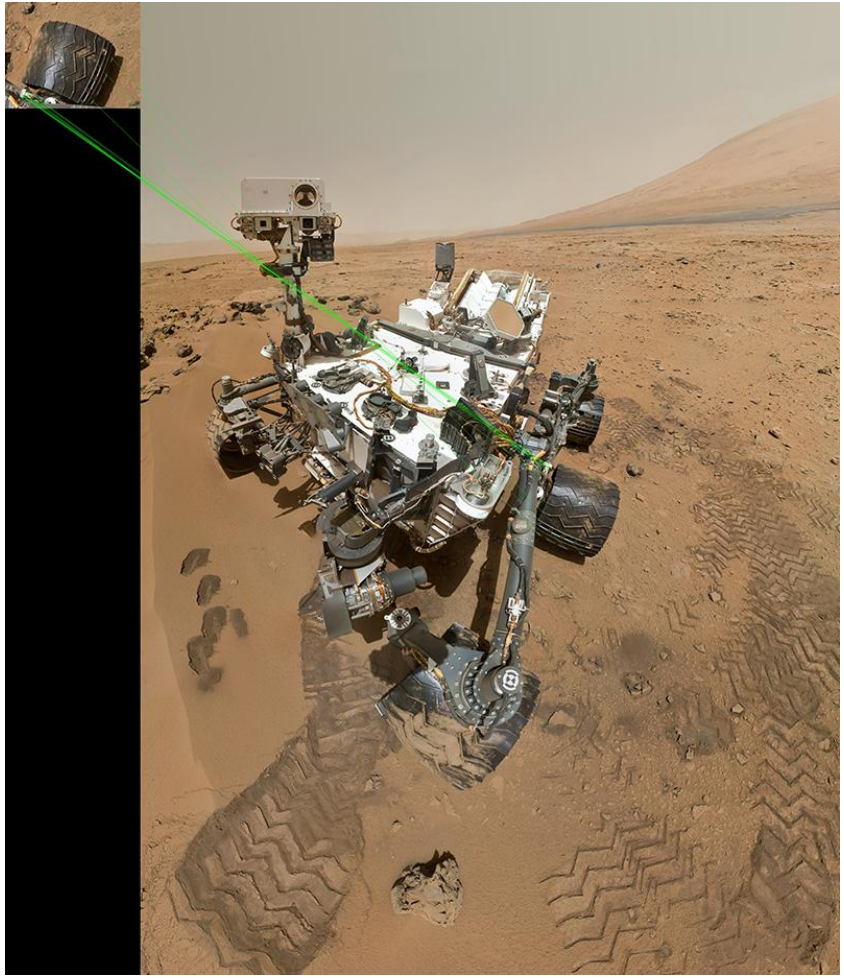
Feature Matching



Feature Matching



Feature Matching



Object Recognition

Cascade classifier using known characteristics

Neural nets using ??? characteristics

Face Recognition

Haar cascades

Most human faces share some characteristics:

- Forehead lighter than eyes
- Eyes are darker than cheekbones
- Ears extend out to the sides
- Nose is about halfway between top of head and chin
- Etc. etc.

Face Recognition

`cv2.CascadeClassifier()`

Face Recognition

Reverse Image Search

Astrometry.net

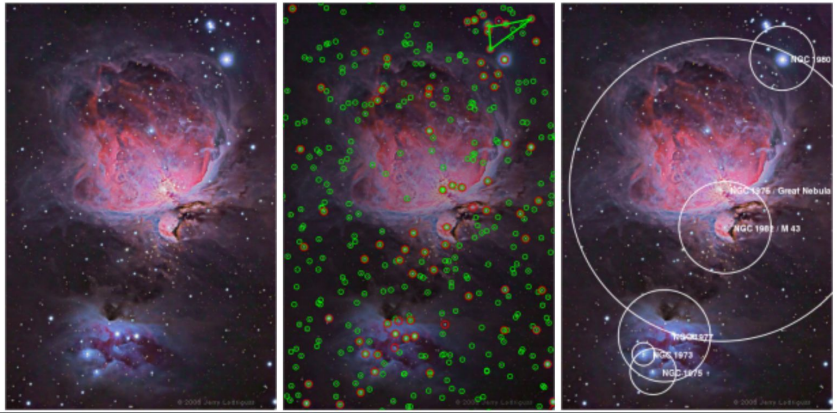
Astrometry.net

home | project summary | people | gallery | news | related links | bibliography | data | use | download | forum

Gallery of Solved Images

In the images below, the red circles are stars our algorithm automatically detects in the image, and the green circles are stars from our master index which appear in the query image. Nebulae, constellations and other objects can be automatically overlayed on the image after it has been solved.

A shot of the Great Nebula, by Jerry Lodriguss (c.2006), from astropix.com



Astrometry.net

https://www.flickr.com/photos/astronoctografo/33008725153

Astrometry.net

Locate the brightest stars

Find relations between them

Search a catalog of known relations

https://www.flickr.com/photos/astronoctografo/33008725153

Astrometry.net

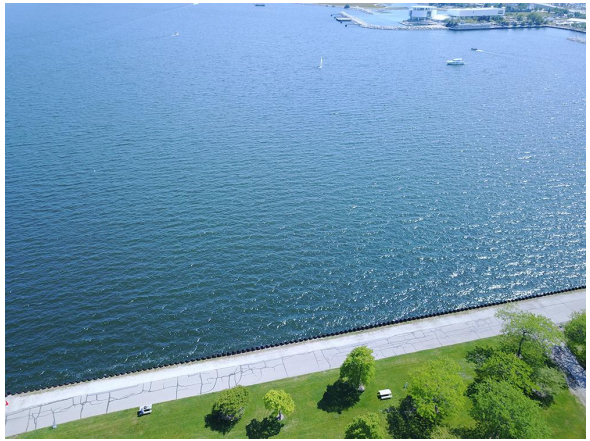
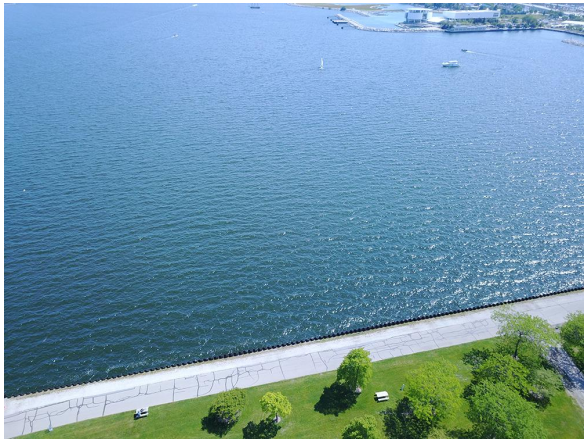
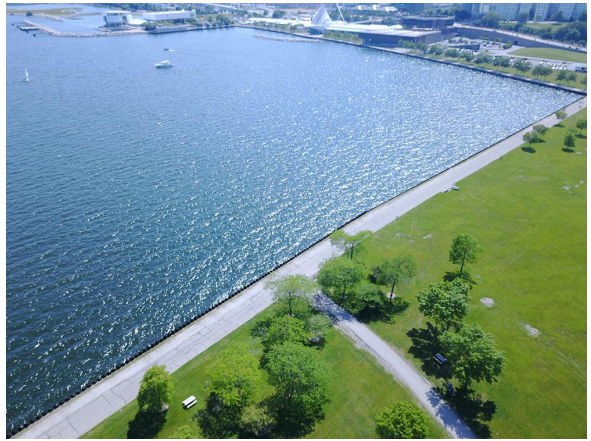
Duplicate Detection

Perceptual Hashes

Low-avalanche hash functions

dhash - Difference hash

dhash

dhash

Convert to grayscale

Resize to postage stamp

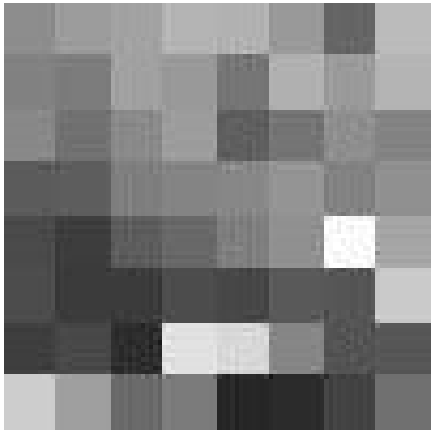
Calculate the difference between adjacent pixels

dhash

Convert to grayscale

Resize to postage stamp

Calculate the difference between adjacent pixels

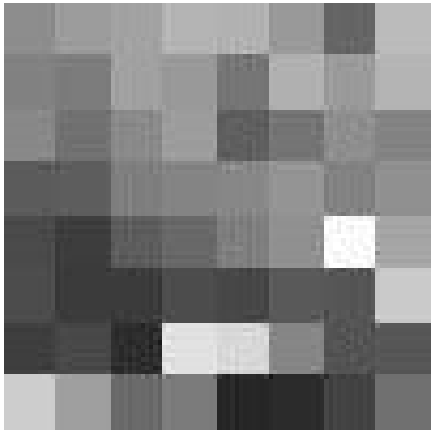


dhash

Convert to grayscale

Resize to postage stamp

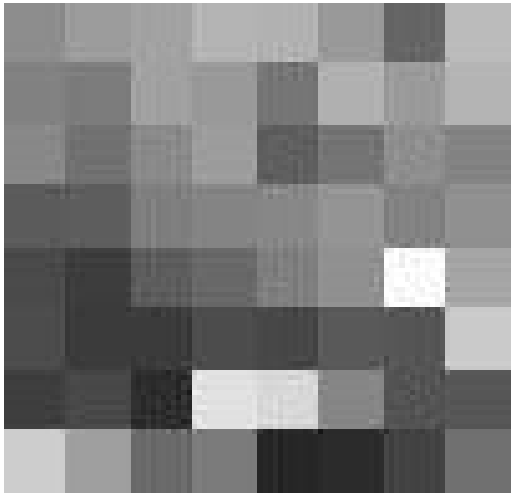
Calculate the difference between adjacent pixels



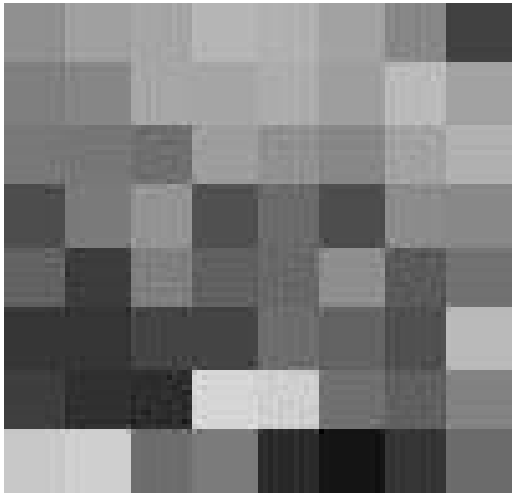
0b1001111111111111111111111111111
1111111111111100010111110100001

0x9ffffffffffe2fa1

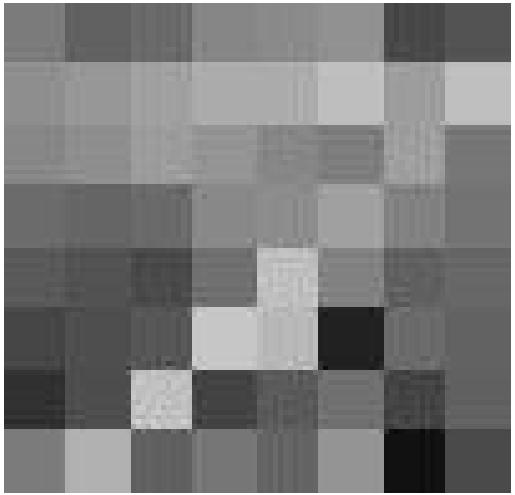
dhash



0x9fffffffffe2fa1



0x9fffffffffe2fa1



0x0d0f3f1f8f469391

dhash

Allows for fuzzy matching as well - Hamming distance

```
  0b10011111111111111111111111111111111111111111110001011111010000 1
 -0b00001101000011110011111100011111100011110100011010010011100100 01
                                                                    26
```

```
$ pip install imagehash
```

```
imagehash.dhash
```

Optical Character Recognition

```
pytesseract.image_to_string()
```

Optical Character Recognition

“United States”

QR Codes

https://foxrow.com/assets/toinfinity.pdf

Photogrammetry

Physical measurements from imagery

Photogrammetry

Maps

Orthomosaics

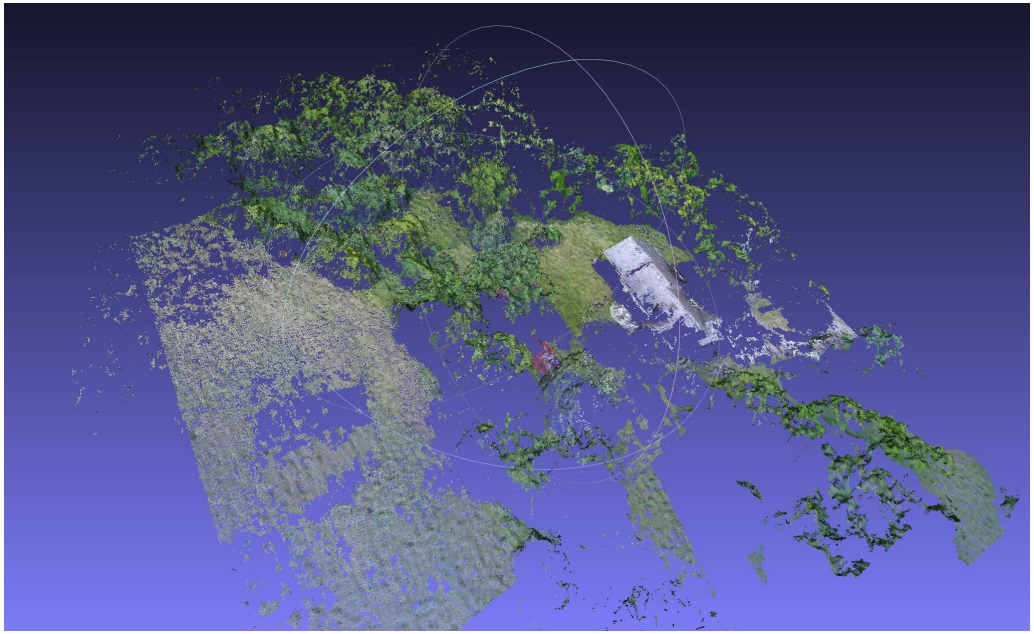
Point clouds

Contour lines

Length/Area/Volume measurement

Terrain classification

Photogrammetry



Neural Networks

Image Classification - VGG16, VGG19, ResNet

Object Recognition - YOLO, Faster R-CNN

OCR

Much more!

Image Classification

What's in an image?

grass, outdoor, people, large, field, park, group, sitting, table, man, standing, grassy, cake, crowd, display, ball, riding, horse, air, umbrella

Object Recognition

To Infinity and Beyond

How can you use it?

```
load DJI_0241.jpg
highlight car truck person
save

load pano/DJI_0002.jpg pano/DJI_0003.jpg
pano/DJI_0004.jpg pano/DJI_0005.jpg
stitch
save
```

How can you use it?

OpenCV - opencv.org

Google, Azure, AWS

Astropy - astropy.org

Deep Sky Stacker - deepskystacker.free.fr

Hugin - hugin.sourceforge.net

OpenDroneMap - github.com/OpenDroneMap

```
load DJI_0241.jpg
highlight car truck person
save

load pano/DJI_0002.jpg pano/DJI_0003.jpg
pano/DJI_0004.jpg pano/DJI_0005.jpg
stitch
save
```

DIL

Drone Imaging Language

https://github.com/foxrow/dil

```
206     def show(self):
207         pano_html = os.path.abspath(os.path.join(self.results_dir.name, 'index.html'))
208         webbrowser.open(pano_html)
209
210     def save(self):
211         for result in os.listdir(self.results_dir.name):
212             shutil.copy(os.path.join(self.results_dir.name, result), '.')
213
214     def execute_command(self, command):
215         if command[0] == 'load':
216             self.load(command[1:])
217         elif command[0] == 'highlight':
218             self.highlight(command[1:])
219         elif command[0] == 'stitch':
220             self.stitch_pano()
221         elif command[0] == 'show':
222             self.show()
223         elif command[0] == 'save':
224             self.save()
225         elif command[0] == '':
226             pass
227         elif command[0].startswith('#'):
228             pass  # comment line
229         else:
230             print('unrecognized command: {}'.format(command[0]))
231
232     def execute(self):
233         for command in self.commands:
234             self.execute_command(command)
235
236
237 if __name__ == '__main__':
238     with open(sys.argv[1]) as f:
239         program = Program(f.read())
240         program.execute()
```


car
0.73
car
0.53
car
0.75
car
0.67
person
0.83

Languages

1.

2.

3.

Languages

1. Don’t have to be complex
2.
3.

Datetime formatting

Python's `strftime` directives

Note: Examples are based on `datetime.datetime(2013, 9, 30, 7, 6, 5)`

Code	Meaning	Example
`%a`	Weekday as locale's abbreviated name.	`Mon`
`%A`	Weekday as locale's full name.	`Monday`
`%w`	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	`1`
`%d`	Day of the month as a zero-padded decimal number.	`30`
`%-d`	Day of the month as a decimal number. (Platform specific)	`30`
`%b`	Month as locale's abbreviated name.	`Sep`
`%B`	Month as locale's full name.	`September`
`%m`	Month as a zero-padded decimal number.	`09`

http://strftime.org

Regular expressions

```
pat = re.compile(r"""
 \s*                 # Skip leading whitespace
 (?P<header>[^:]+)   # Header name
 \s* :               # Whitespace, and a colon
 (?P<value>.*?)      # The header's value -- *? used to
                     # lose the following trailing whitespace
 \s*$                # Trailing whitespace to end-of-line
""", re.VERBOSE)
```

https://docs.python.org/3.6/howto/regex.html

String formatting mini-language

```
format_spec     ::=  [[fill]align][sign][#][0][width][grouping_option][.precision][type]
fill            ::=  <any character>
align           ::=  "<" | ">" | "=" | "^"
sign            ::=  "+" | "-" | " "
width           ::=  integer
grouping_option ::=  "_" | ","
precision       ::=  integer
type            ::=  "b" | "c" | "d" | "e" | "E" | "f" | "F" | "g" | "G" | "n" | "o" | "s" | "x"
```

```
>>> '{:+f}; {:+f}'.format(3.14, -3.14)  # show it always
'+3.140000; -3.140000'
>>> '{: f}; {: f}'.format(3.14, -3.14)  # show a space for positive numbers
' 3.140000; -3.140000'
>>> '{:-f}; {:-f}'.format(3.14, -3.14)  # show only the minus -- same as '{:f}; {:f}'
'3.140000; -3.140000'
```

https://docs.python.org/3/library/string.html#formatspec

Languages

1. Don’t have to be complex
2. Trade generality for simplicity
3.

Simplicity

```
load

highlight

stitch

show

save
```

Languages

1. Don’t have to be complex
2. Trade generality for simplicity
3. Are interfaces

Ryan Fox

ryan@foxrow.com

@ryan_fox

foxrow.com

github.com/foxrow/dil