CS6476 – Office Hours

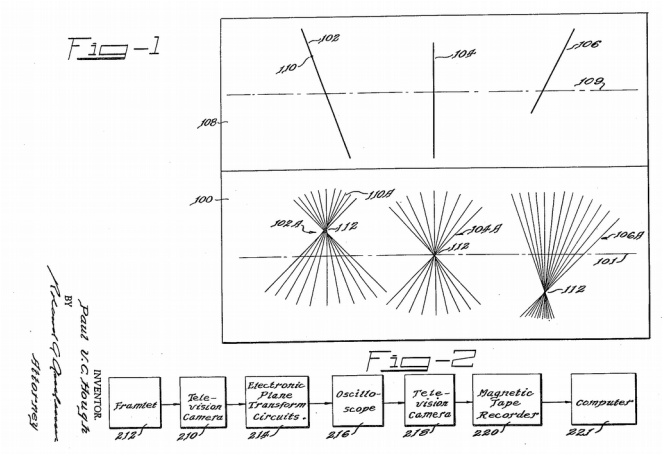
Matthew Houston
2017-09-20

Agenda

- Hough Transform (once more)
- Exploring Problem Set 3 Techniques
- Important Linear Algebra Functions
- Open Questions

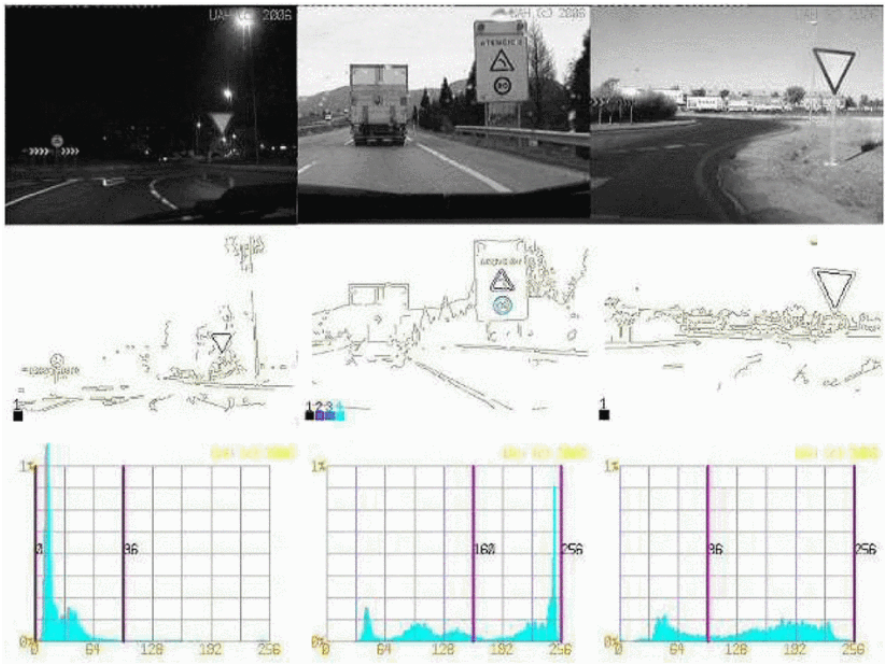
Hough Transform (once more)

- **What's the point of problem set 2?**



Hough, P.V.C. **Method and means for recognizing complex patterns**, U.S. Patent 3,069,654, Dec. 18, 1962

Hough algorithms have been used for over 40 years in computer vision systems



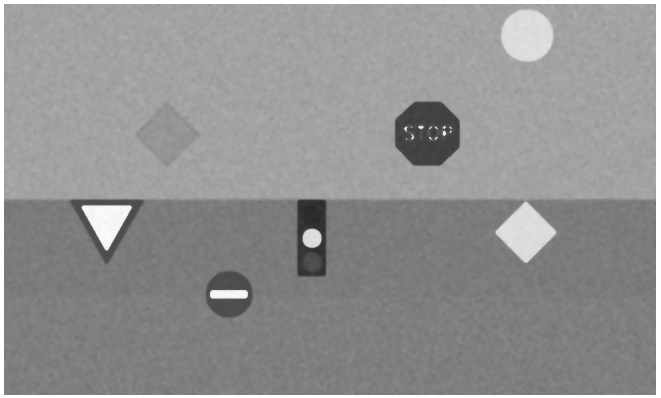
Garcia-Garrido, M.A. Sotelo, M.A. Martin-Gorostiza E. **Fast traffic sign detection and recognition under changing lighting conditions.** IEEE Intelligent Transportation Systems Conference. 2006

- **Parameter tuning is an integral part of computer vision**

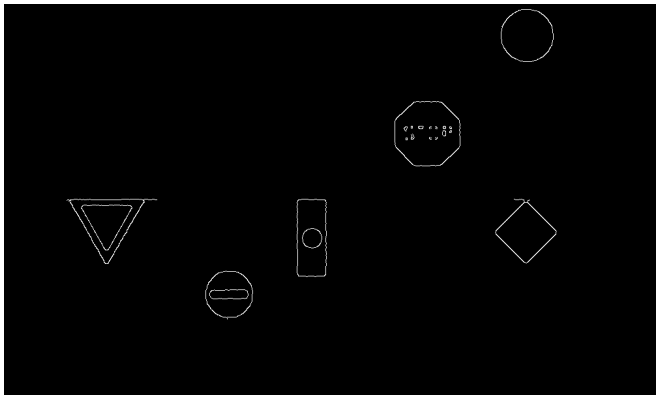
- **From problem set 2 we expect students to develop intuition for using hough tools and an understanding of their limitations**

Hough Transform (once more)

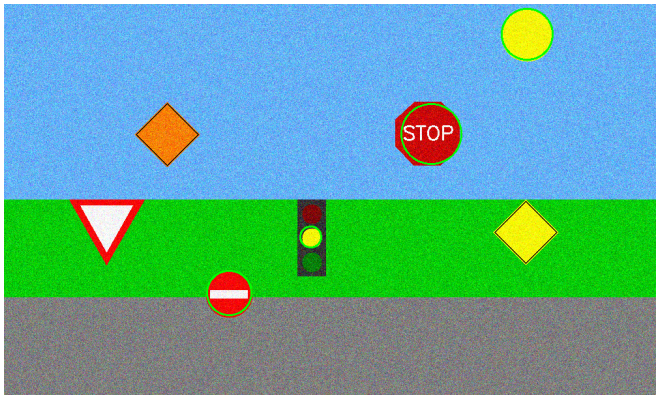
- How to get good results from Hough Circles?
  - Pay attention to the input quality (filtering is very important)
  - Remember that Hough Circles has built in Canny
  - Set parameters based on multiple input images + algorithm understanding
- Hough isn't a perfect algorithm – build algorithms expecting imperfections or use a different technique (which is a nice lead-in to PS3)



Standard gray-scale conversion doesn't show many strong circles

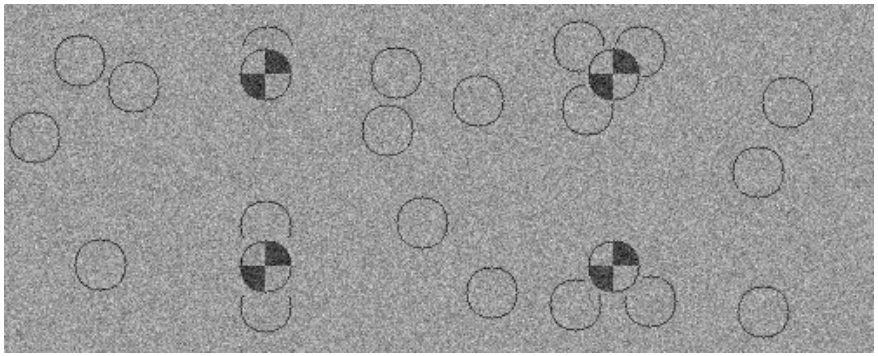


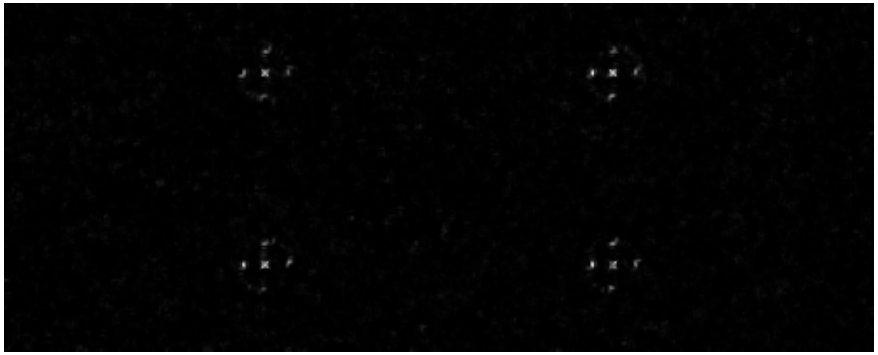
Internal Canny doesn't identify circles on the traffic light

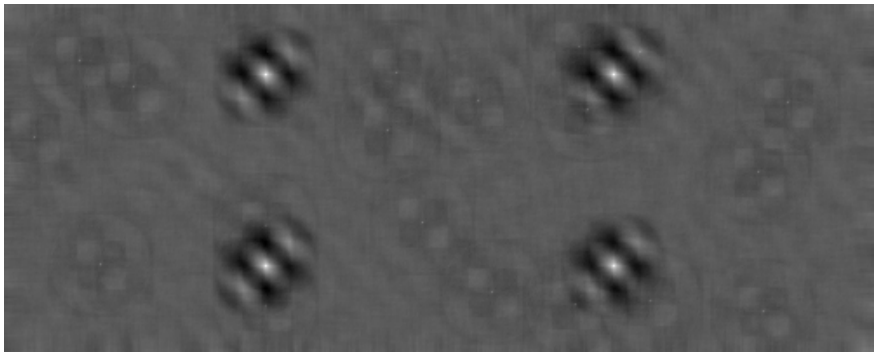


All circles identified by the internal canny are found – others are ignored

Exploring Problem Set 3 Techniques







- Images shown (not in order):
  Original, Harris,
  Minimum Eigenvalue,
  Template Matching, Canny

Useful Linear Algebra Functions

- np.linalg.solve: solves systems of equations where the number of equations matches the number of unknowns

- np.linalg.lstsq: finds least squares solutions to systems of equations

- np.linalg.svd: performs the singular value decomposition

- np.dot: multiplies two matrices

- Finally a common optimization opportunity…
  - If A is a matrix and I have a series of column vectors (x1, x2, x3) which I would like to multiply by A then these two code blocks are equivalent

```
# slow version (0.93 seconds)
A = np.random.randn(10,10)
x = np.random.randn(10,1000000)
y = np.zeros((10,1000000))
for i in xrange(0,1000000):
  y[:,i] = np.dot(A,x[:,i])
```

```
# fast version (0.026 seconds)
A = np.random.randn(10,10)
x = np.random.randn(10,1000000)
y = np.dot(A,x)
```