



# Gradient-Free Boundary Tracking

Zhong Hu  
*August 31, 2007*  
UCLA

*Faculty Supervisor: Todd Wittman*

*This research is supported by NGA NURI grant HM1582-06-BAA-004*



## Boundary Tracking

- We want to segment a region of interest in a (hypespectral) image.
- Classical gradient-based segmentation methods like active contours and snakes look at all pixels in the image.
- Ideally, the number of pixels checked would be proportional to the length of the boundary.
- The solution is to “walk” the boundary.
- We propose a gradient-free boundary tracking algorithm inspired by the robotic vehicle tracking algorithm developed by (*Kemp-Bertozzi-Marthaler 2004*).

# Boundary Tracking Algorithm

Input: Angular velocity  $w$ , tracking velocity  $v$

Ask the user to input a point  $(x,y)$  from the image

if  $(x,y)$  is inside the boundary

set  $d = 1$

else

set  $d = -1$

Initialize  $\theta = 0$

For fixed number of iterations

Set  $\theta = \theta + d * w$

$x = x + v * \cos \theta$

$y = y + v * \sin \theta$

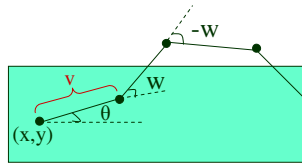
if we have made a full circle

$v = 2 * v$

if we cross the boundary

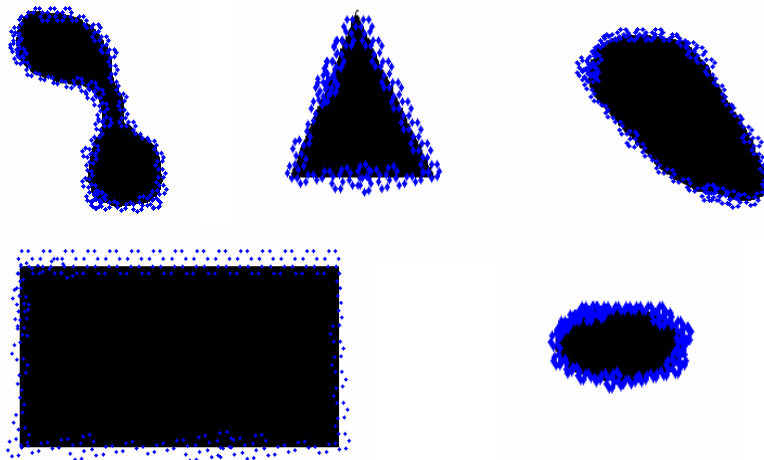
Store the boundary point

$d = -d$



*(Kemp-Bertozzi-Marthaler 2004)*

## Track the basic shapes



# Angle Correction

(Jin-Bertozzi 2007)

The goal is to detect as many boundary points as possible while minimizing the number of pixels tested.

$$\text{Percent Pixels on Boundary (PPB)} = \frac{\# \text{ boundary points detected}}{\text{Total \# pixels tested}}$$

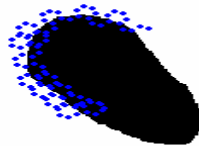
**Angle Correction:** After we cross the boundary, update  $\theta$  by

$$\theta = \theta + d \frac{w(t_2 - t_1) - 2\theta_{ref}}{2}$$

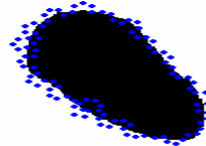
where  $t_1, t_2$  are times of last two boundary crossings and  $\theta_{ref}$  is a parameter.

Without angle correction

With angle correction



Speed  $V=5, w=1, PPB=0.21, 100$  iterations



Speed  $V=5, w=1, PPB=0.42, 100$  iterations

# CUSUM Filters

(Jin-Bertozzi 2007)

If there is substantial noise in the image, our tracking algorithm fails to track the boundary. To overcome this, we apply two independent cumulative sum (CUSUM) filters:

$$U(t) = \begin{cases} 0 & \text{if } t = 0 \\ \min(\bar{U}, \max(0, A(x, y) - B - c_u + U)) & \text{if } t > 0 \end{cases}$$

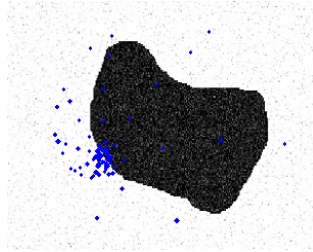
$$L(t) = \begin{cases} 0 & \text{if } t = 0 \\ \max(\bar{L}, \min(0, A(x, y) - B + c_l + L)) & \text{if } t > 0 \end{cases}$$

Where  $\bar{u}$  and  $\bar{l}$  are the accumulation threshold,  $A$  is the image,  $A(x,y)$  is the intensity at location  $(x,y)$ ,  $B$  is the threshold for the image  $A$ , and  $c_u$  and  $c_l$  are the "dead-zone" parameters.

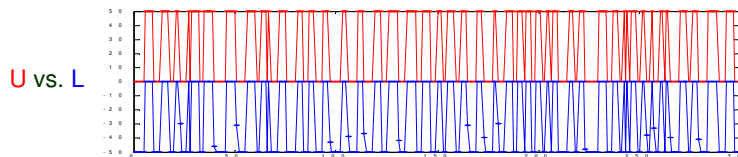
## CUSUM Filters

The left image shows the failure of the original algorithm to trace the boundary with Gaussian noise variance 0.1. The image on the right shows the results using CUSUM filters by choosing  $\bar{U}=50$ ,  $\bar{L}=-50$ ,  $c_u=2$ ,  $c_l=2$ , and  $B=100$  to the image with Gaussian noise variance 0.1.

Without CUSUM



With CUSUM



## CUSUM Filters

Even with a noise-free image, CUSUM helps to trace the boundary better.

With CUSUM filters



V=3, PPB=0.297, 150 iterations

Without CUSUM filters



V=3, PPB=0.43, 150 iterations

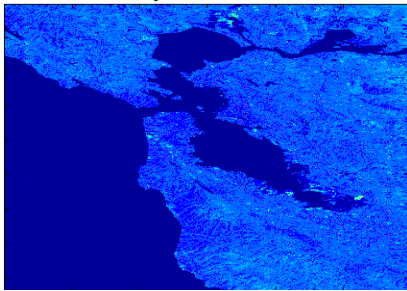
## Tracking San Francisco Bay

- The Normalized Difference Vegetation Index is used to differentiate natural from man-made regions.
- It is calculated at every pixel using just the Near Infrared (NIR) and Red bands.

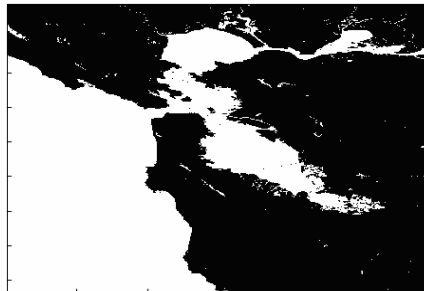
$$\text{NDVI} = \frac{\rho_{nir} - \rho_{red}}{\rho_{nir} + \rho_{red}} \quad (\text{Rouse et. al. 1974})$$

- Thresholding NDVI appropriately also does a good job telling land from water.

A Near Infrared band in San Francisco Bay

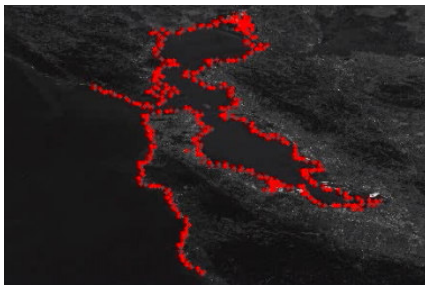


NDVI of San Francisco Bay

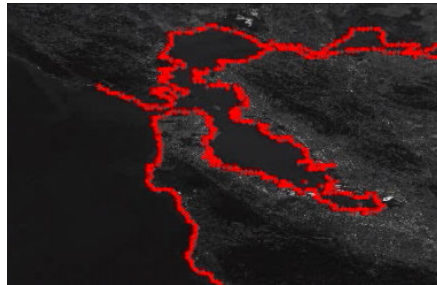


## Tracking San Francisco Bay

The smaller V traces the boundary more accurately, but it takes a longer time to run.



V=25, Runtime=20.2 sec



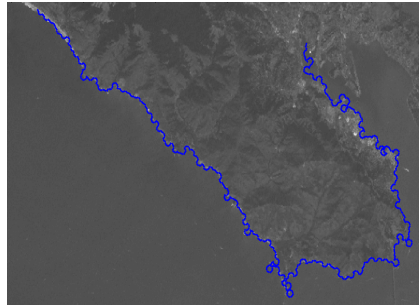
V=15, Runtime=38.8 sec

## CUSUM on San Francisco Bay

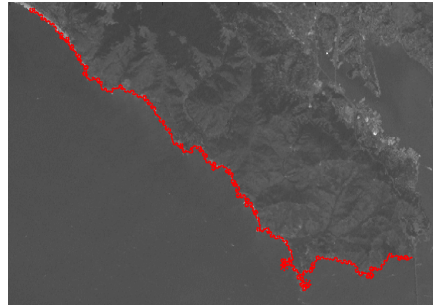
Perform the algorithm on a small portion of the San Francisco Bay (500x600) and use small  $V$

Using the method of CUSUM filters with the right parameters seems to trace the boundary much better than the method without the CUSUM filters.

800 iterations with CUSUM,  $V=3$ ,  
PPB = 0.33



800 iterations without CUSUM,  $V=3$ ,  
PPB = 0.42



## Project Goals

### Summer project:

- Implement the boundary tracking algorithm.
- Find that angle correction and CUSUM works better
- Used to track water on hyperspectral images

### Future work:

- Look for new boundary tracking algorithms
- Compare different algorithms
- Try to track roads, buildings, and forests on hyperspectral images (look for detection filters)