



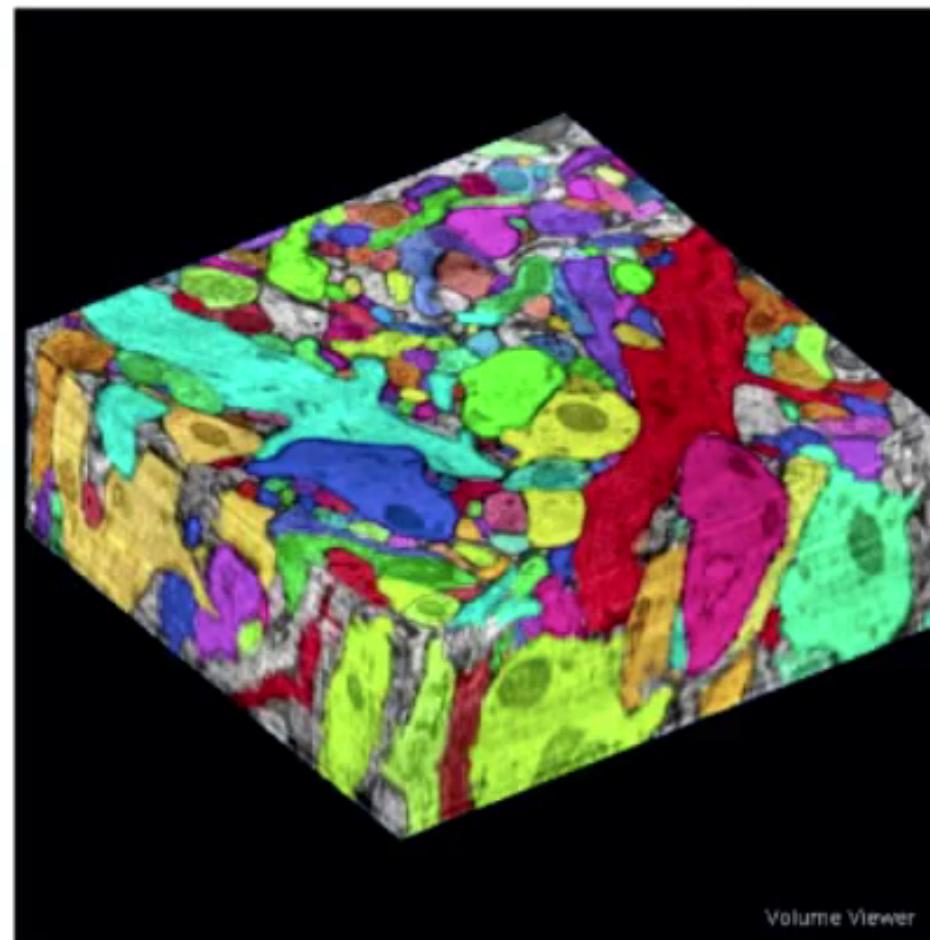
Courtesy of IEEE and IJCV



Courtesy of IEEE and IJCV



Courtesy of IEEE and IJCV



Courtesy of openconnectomeproject.org



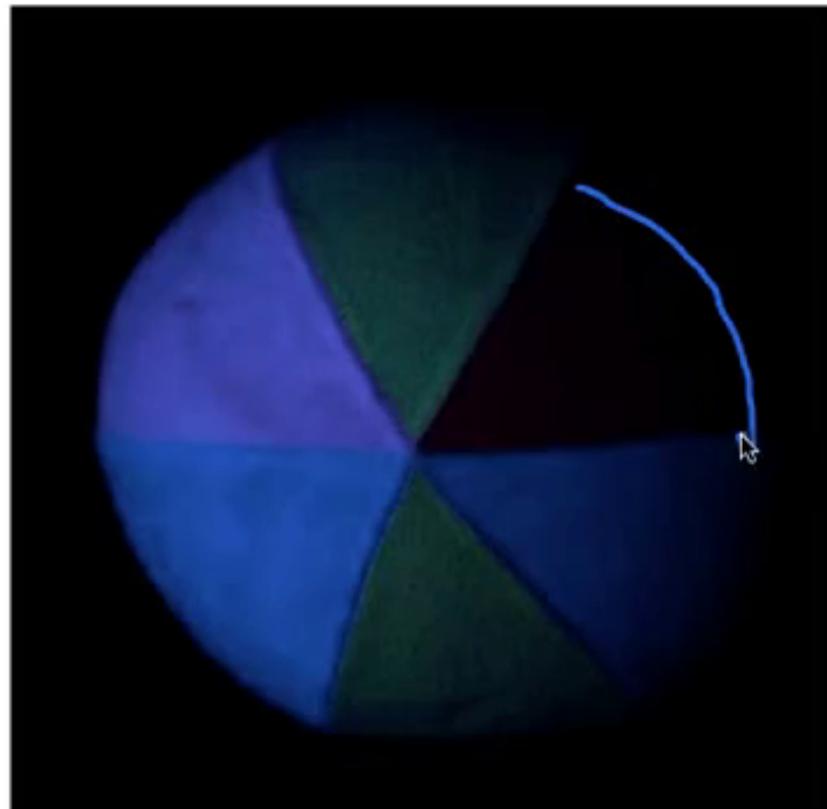
Original SIGGRAPH project videos courtesy of Artbeats (www.artbeats.com), Mike Wilbur, Jon Goldman, and Jiawen Chen



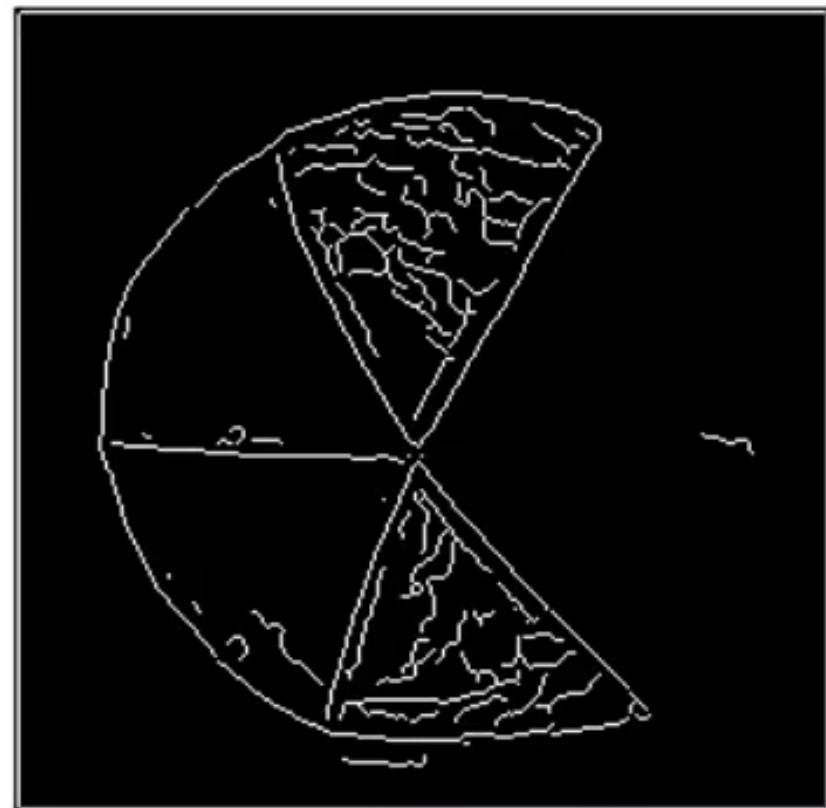
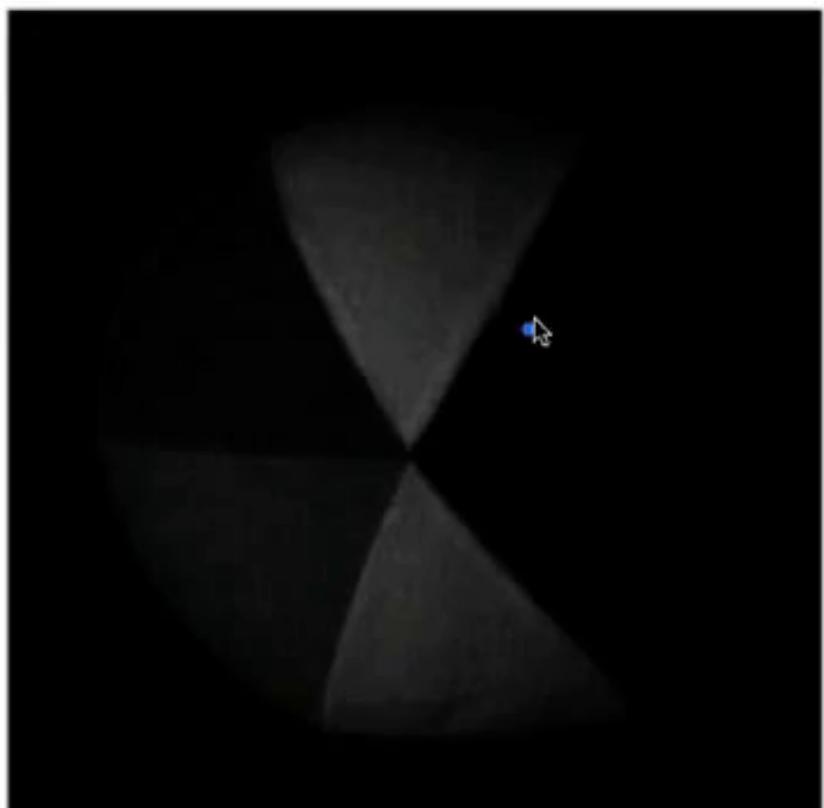
$$\begin{aligned}\nabla I &= \\ &= \left\langle \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right\rangle \\ &\quad \boxed{-1 \quad +1 \quad -1}\end{aligned}$$

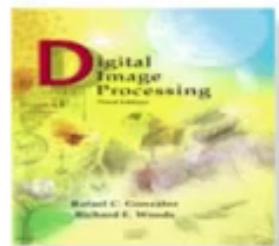


Images courtesy of ipol.im



Images courtesy of ipol.im





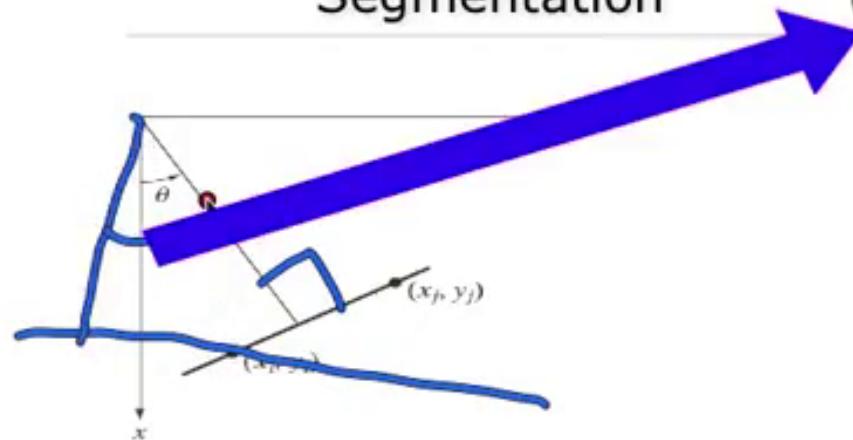
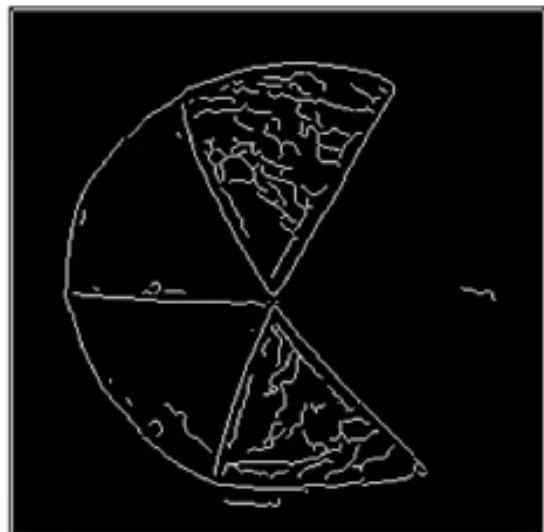
Digital Image Processing, 3rd ed.

Gonzalez & Woods

www.ImageProcessingPlace.com

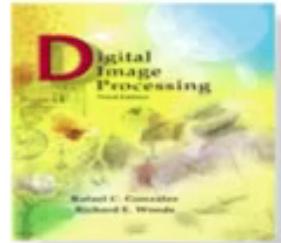
Chapter 10 Segmentation

Angle measured to x axis.



$$\rho = x \cos \theta + y \sin \theta$$





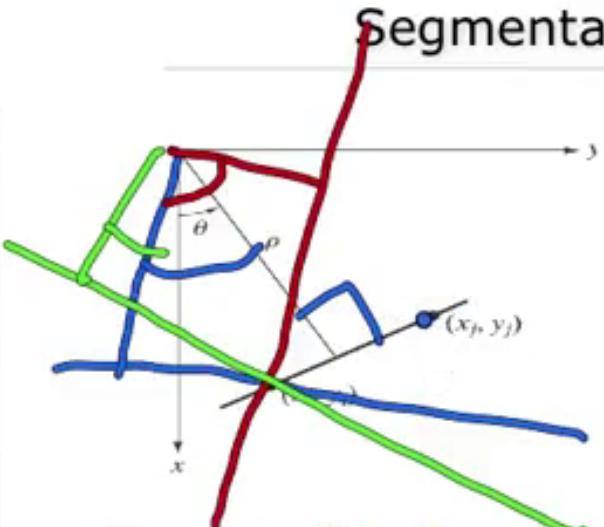
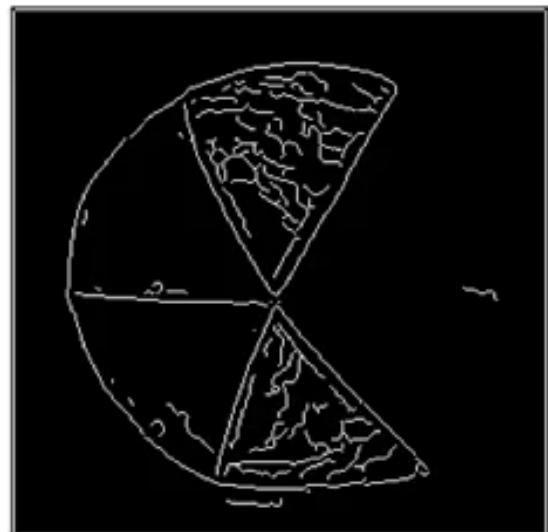
Digital Image Processing, 3rd ed.

Gonzalez & Woods

www.ImageProcessingPlace.com

Chapter 10

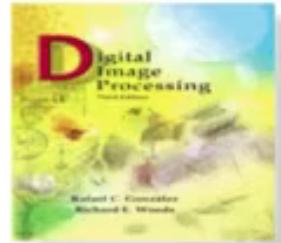
Segmentation



$$\rho = x \cos \theta + y \sin \theta$$

$$(\rho, \theta)$$





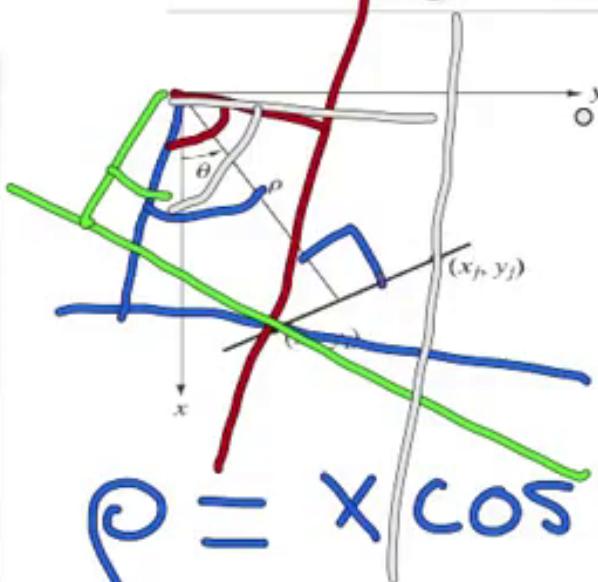
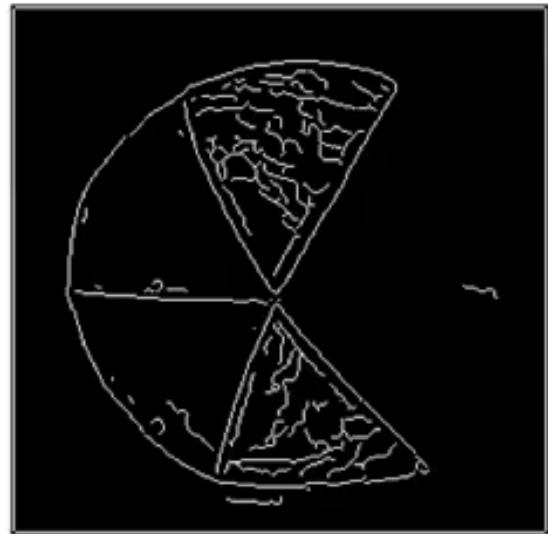
Digital Image Processing, 3rd ed.

Gonzalez & Woods

www.ImageProcessingPlace.com

Chapter 10

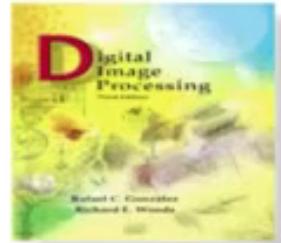
Segmentation



$$\rho = x \cos \theta + y \sin \theta$$

$$(\rho, \theta)$$





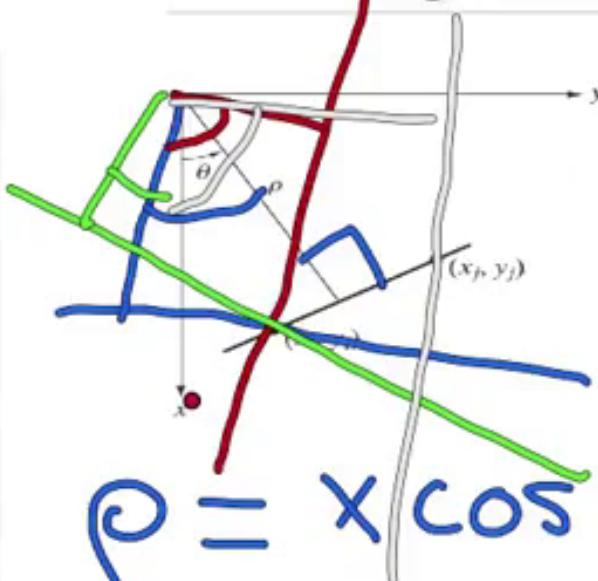
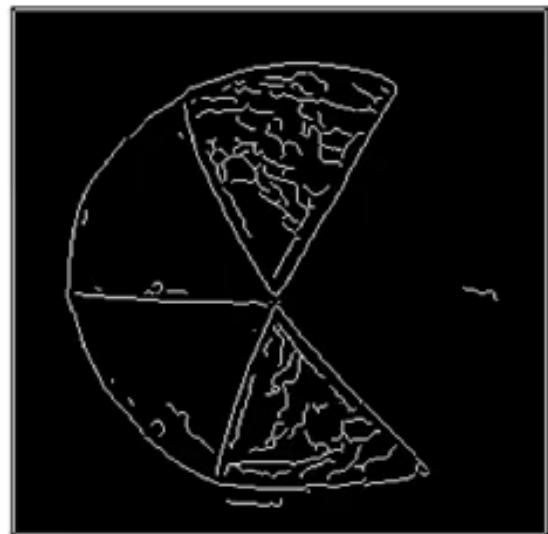
Digital Image Processing, 3rd ed.

Gonzalez & Woods

www.ImageProcessingPlace.com

Chapter 10

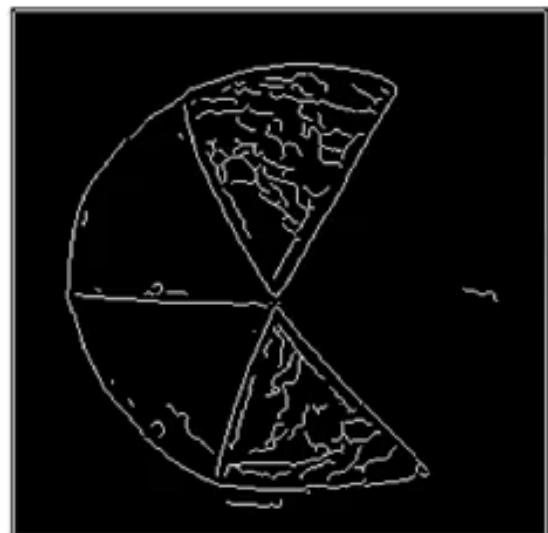
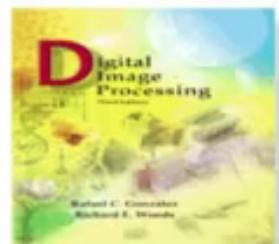
Segmentation



$$\rho = x \cos \theta + y \sin \theta$$

$$(\rho, \theta) \quad (\rho, \theta)$$





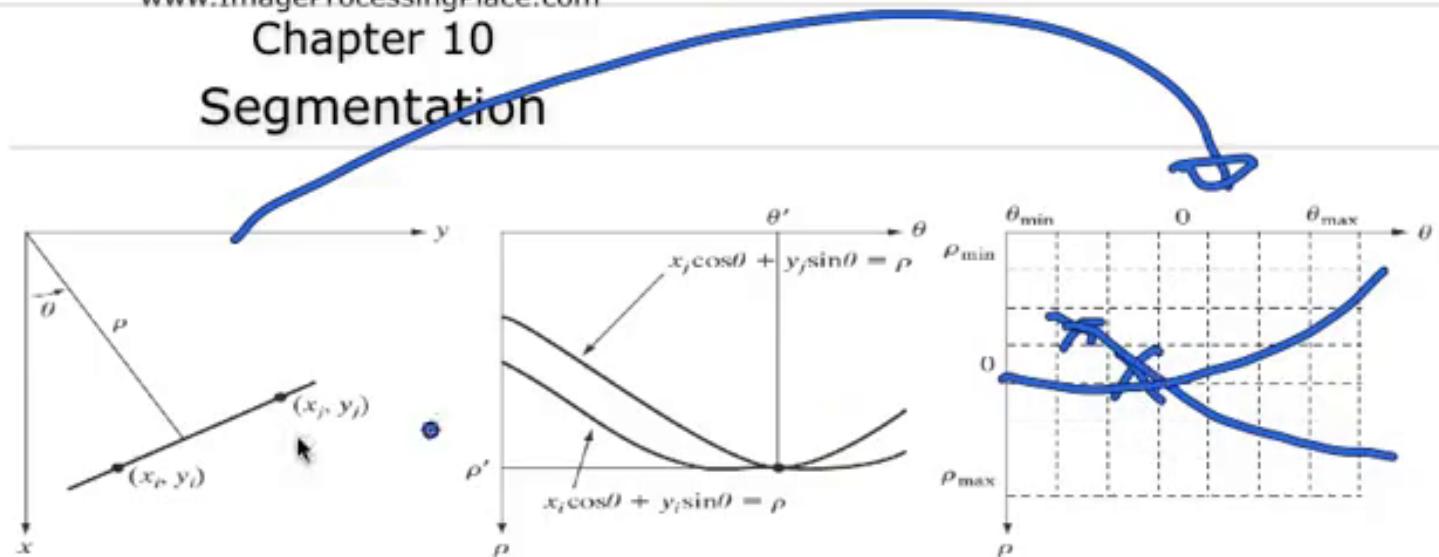
Digital Image Processing, 3rd ed.

Gonzalez & Woods

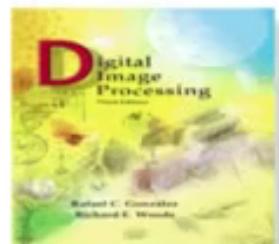
www.ImageProcessingPlace.com

Chapter 10

Segmentation



$$\rho = x \cos \theta + y \sin \theta$$



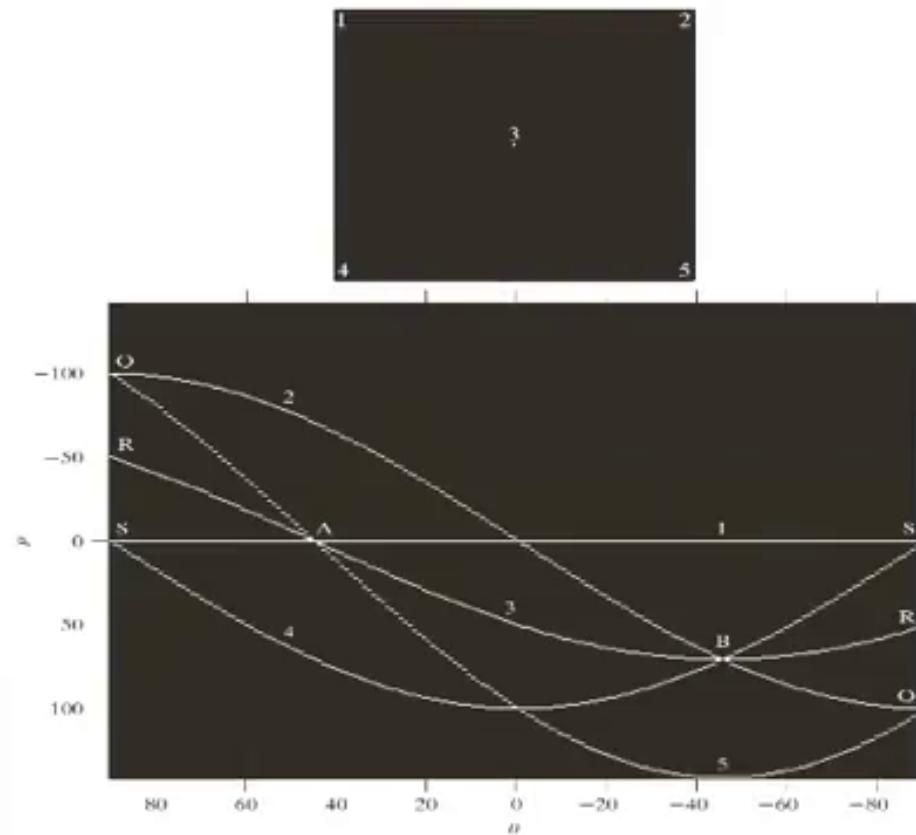
Digital Image Processing, 3rd ed.

Gonzalez & Woods

www.ImageProcessingPlace.com

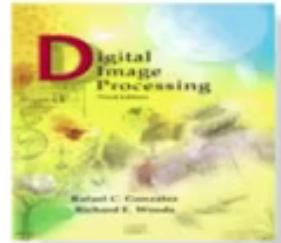
Chapter 10

Segmentation



a
b

FIGURE 10.33
(a) Image of size 101×101 pixels, containing five points.
(b) Corresponding parameter space. (The points in (a) were enlarged to make them easier to see.)



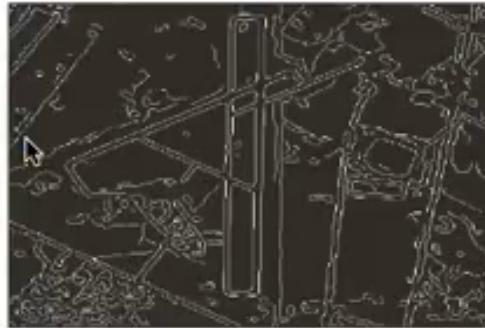
Digital Image Processing, 3rd ed.

Gonzalez & Woods

www.ImageProcessingPlace.com

Chapter 10

Segmentation





File Folder: /Users/guillermo_sapiro/Documents/MATLAB ...

Command Window

MATLAB? Watch this [Video](#), see [Demos](#), or read [Getting Started](#).

```
len = norm(lines(k).point1 - lines(k).point2);
if ( len > max_len)
    max_len = len;
    xy_long = xy;
end
end

% highlight the longest line segment
plot(xy_long(:,1),xy_long(:,2),'LineWidth',2,'Color','cyan');

See also hough and houghpeaks.
```

Reference page in Help browser
[doc houghlines](#)

```
>>
I = imread('circuit.tif');
rotI = imrotate(I,33,'crop');
BW = edge(rotI,'canny');
[H,T,R] = hough(BW);
imshow(H,[],'XData',T,'YData',R,'InitialMagnification','fit');
xlabel('\theta'), ylabel('\rho');
axis on, axis normal, hold on;
P = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:))));
x = T(P(:,2));
y = R(P(:,1));
plot(x,y,'s','color','white');

% Find lines and plot them
lines = houghlines(BW,T,R,P,'FillGap',5,'MinLength',7);
figure, imshow(rotI), hold on
max_len = 0;
for k = 1:length(lines)
    xy = [lines(k).point1; lines(k).point2];
    plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

    % plot beginnings and ends of lines
    plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
    plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');

    % determine the endpoints of the longest line segment
    len = norm(lines(k).point1 - lines(k).point2);
    if ( len > max_len)
        max_len = len;
        xy_long = xy;
    end
end
```

Select a file to view details

fx >>

Name

- BW
- H
- I
- P
- R
- T
- ball
- e
- im
- k
- len
- lines
- max_len
- radii
- rotI
- x
- xy

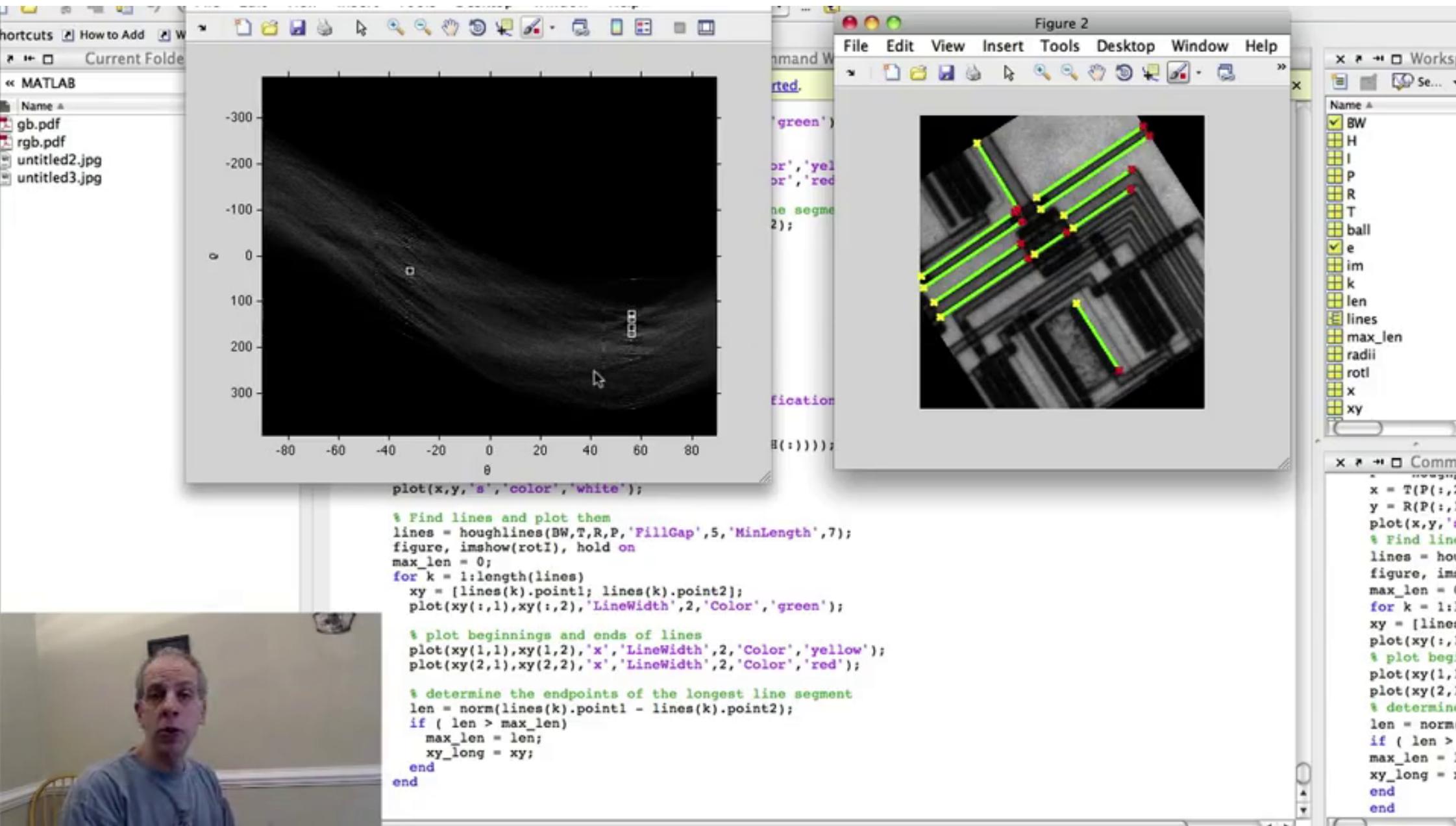
Comm

```
x = T(P(:,2));
y = R(P(:,1));
plot(x,y,'s','color','white');

% Find lines and plot them
lines = houghlines(BW,T,R,P,'FillGap',5,'MinLength',7);
figure, imshow(rotI), hold on
max_len = 0;
for k = 1:length(lines)
    xy = [lines(k).point1; lines(k).point2];
    plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

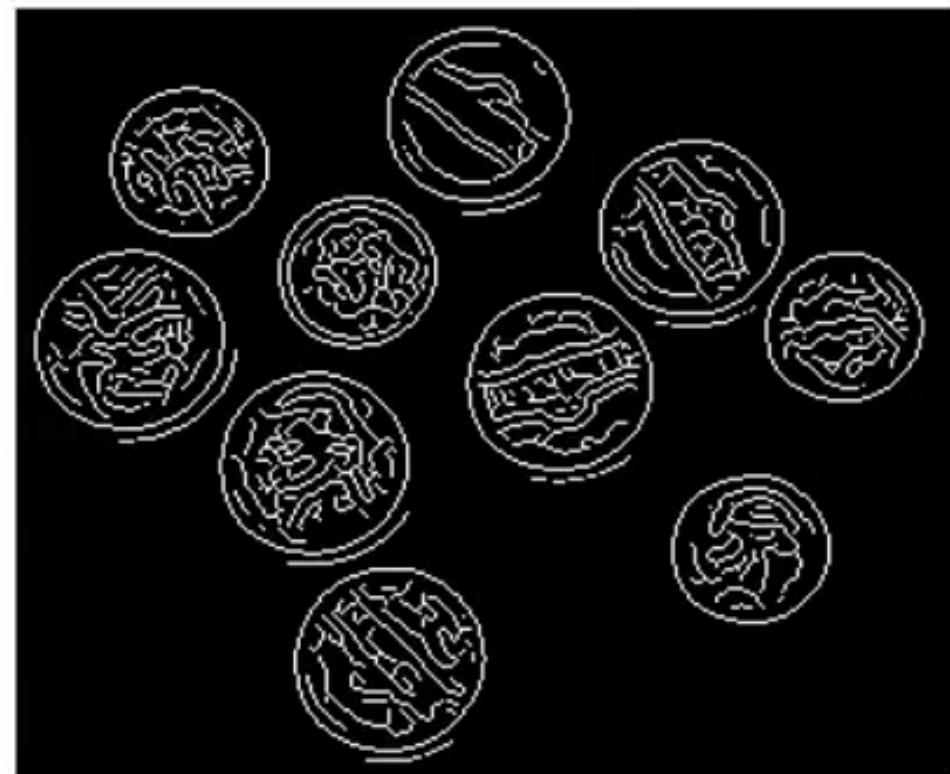
    % plot beginnings and ends of lines
    plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
    plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');

    % determine the endpoints of the longest line segment
    len = norm(lines(k).point1 - lines(k).point2);
    if ( len > max_len)
        max_len = len;
        xy_long = xy;
    end
end
```



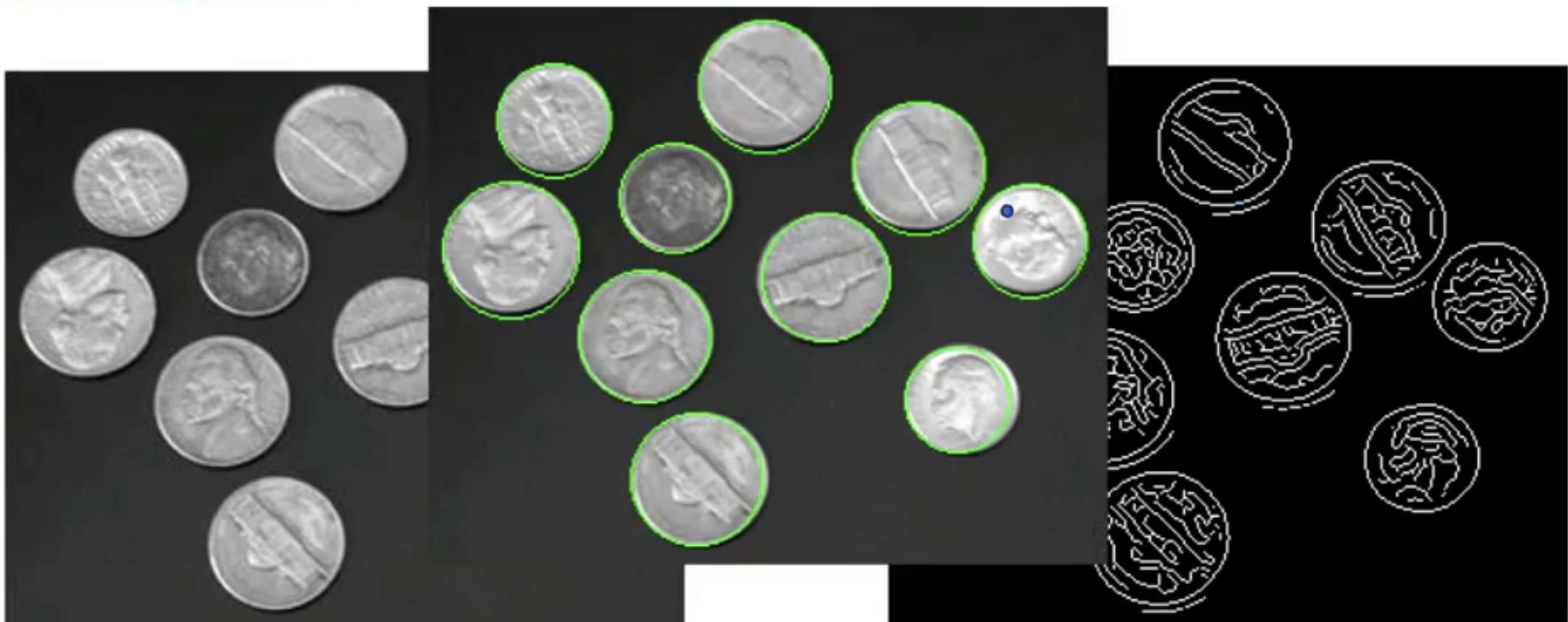


What About Circles?

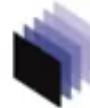




What About Circles?



Images courtesy of D. Young and Mathworks



LSD: a Line Segment Detector

Rafael Grompone von Gioi, Jérémie Jakubowicz, Jean-Michel Morel, Gregory Randall

[article](#) [demo](#) [archive](#)

published · 2012-03-24

reference · Grompone von Gioi, Rafael, Jérémie Jakubowicz, Jean-Michel Morel, and Gregory Randall. "LSD: a Line Segment Detector." *Image Processing On Line* 2012 (2012). <http://dx.doi.org/10.5201/ipol.2012.gjmr-lsd>

- full text manuscript: PDF high-res. [?]
- source code: ZIP

Communicated by Lionel Moisan

Demo edited by Rafael Grompone

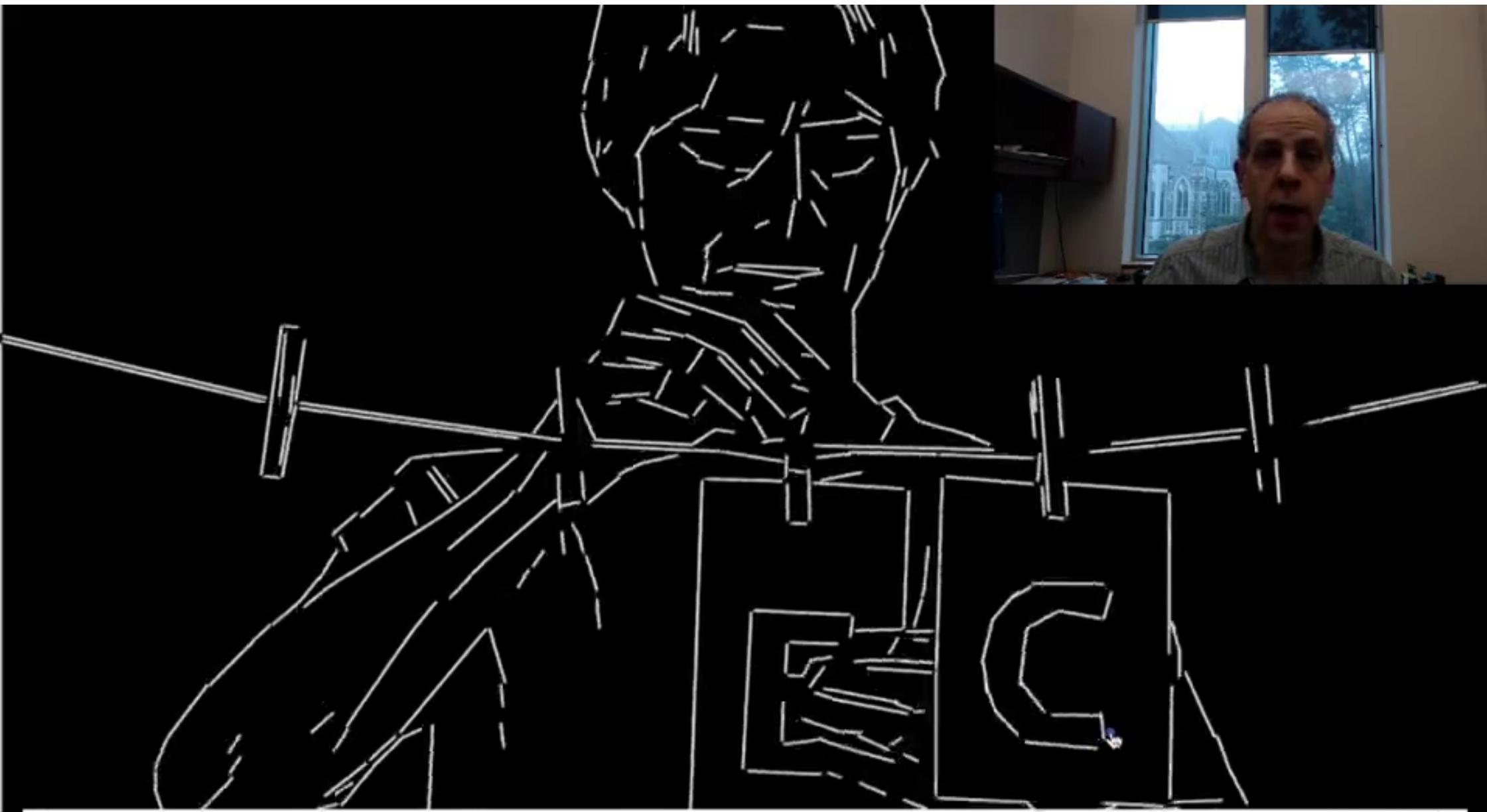
Abstract

LSD is a linear-time Line Segment Detector giving subpixel accurate results. It is designed to work on any digital image without parameter tuning. It controls its own number of false detections: On average, one false alarms is allowed per image. The method is based on Burns, Hanson, and Riseman's method, and uses an a-contrario validation approach according to Desolneux, Moisan, and Morel's theory. The version described here includes some further improvement over the one described in the original article.

Supplementary Material

- sample video: MP4 [?]





00:13 00:43



LSD: a Line Segment Detector

[article](#) [demo](#) [archive](#)

Please cite the reference article if you publish results obtained with this online demo.

The image was converted to gray level values.

Run the algorithm:

Or you can run it after selecting a subimage by clicking two opposite corners of the subimage.



article demo archive

Please cite the reference article if you publish results obtained with this online demo.

Run again?

Result

698 Line Segments were detected. The algorithm ran in 0.22s.

You can download the result in EPS format, in SVG format, or an ASCII file (see description below).

output

input



Result

847 Line Segments were detected. The algorithm ran in 0.28s.

You can download the result in EPS format, in SVG format, or an ASCII file (see description below).

output

input



Result

847 Line Segments were detected. The algorithm ran in 0.28s.

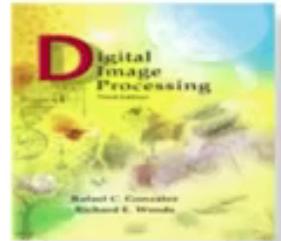
You can download the result in EPS format, in SVG format, or an ASCII file (see description below).

output



input





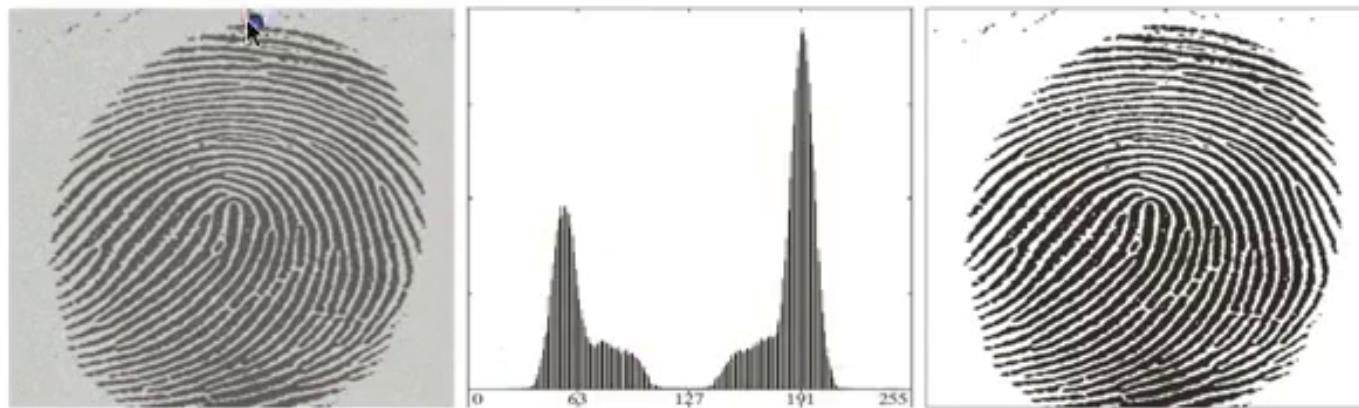
Digital Image Processing, 3rd ed.

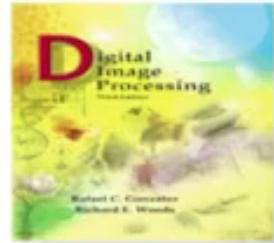
Gonzalez & Woods

www.ImageProcessingPlace.com

Chapter 10

Segmentation





Digital Image Processing, 3rd ed.

Gonzalez & Woods

www.ImageProcessingPlace.com

Chapter 10

Segmentation

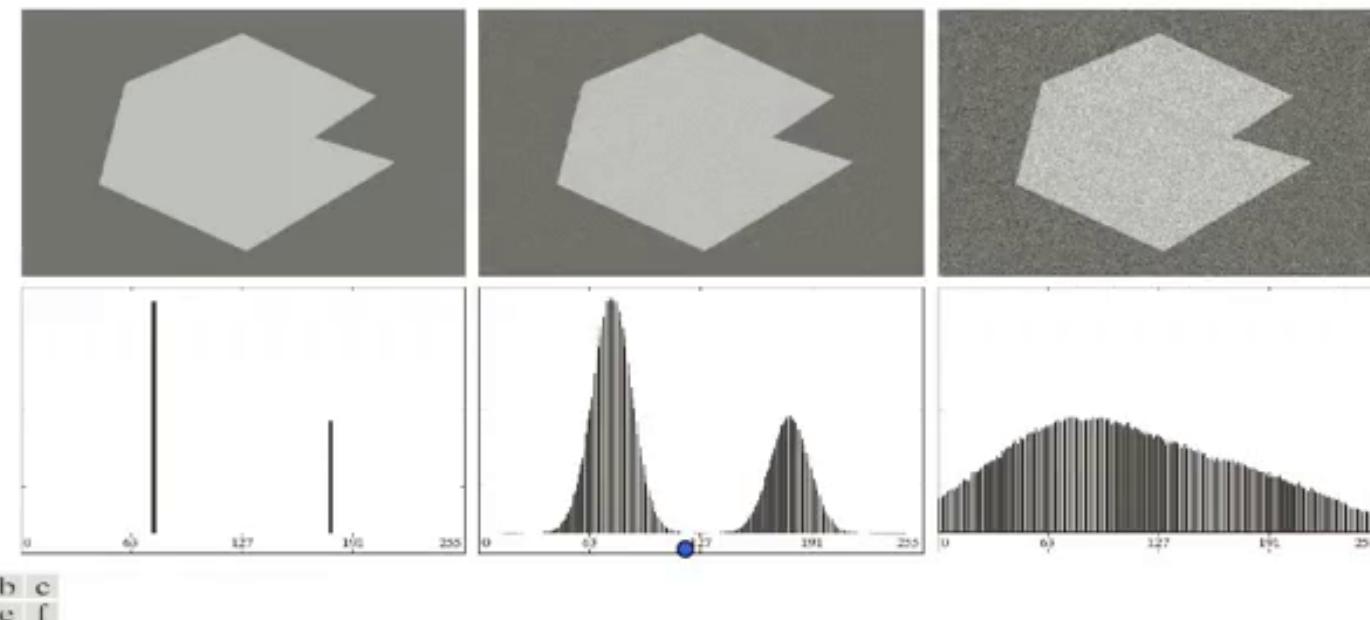
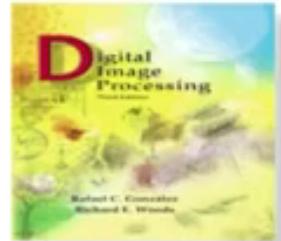


FIGURE 10.36 (a) Noiseless 8-bit image. (b) Image with additive Gaussian noise of mean 0 and standard deviation of 10 intensity levels. (c) Image with additive Gaussian noise of mean 0 and standard deviation of 50 intensity levels. (d)–(f) Corresponding histograms.



Digital Image Processing, 3rd ed.

Gonzalez & Woods

www.ImageProcessingPlace.com

Chapter 10

Segmentation

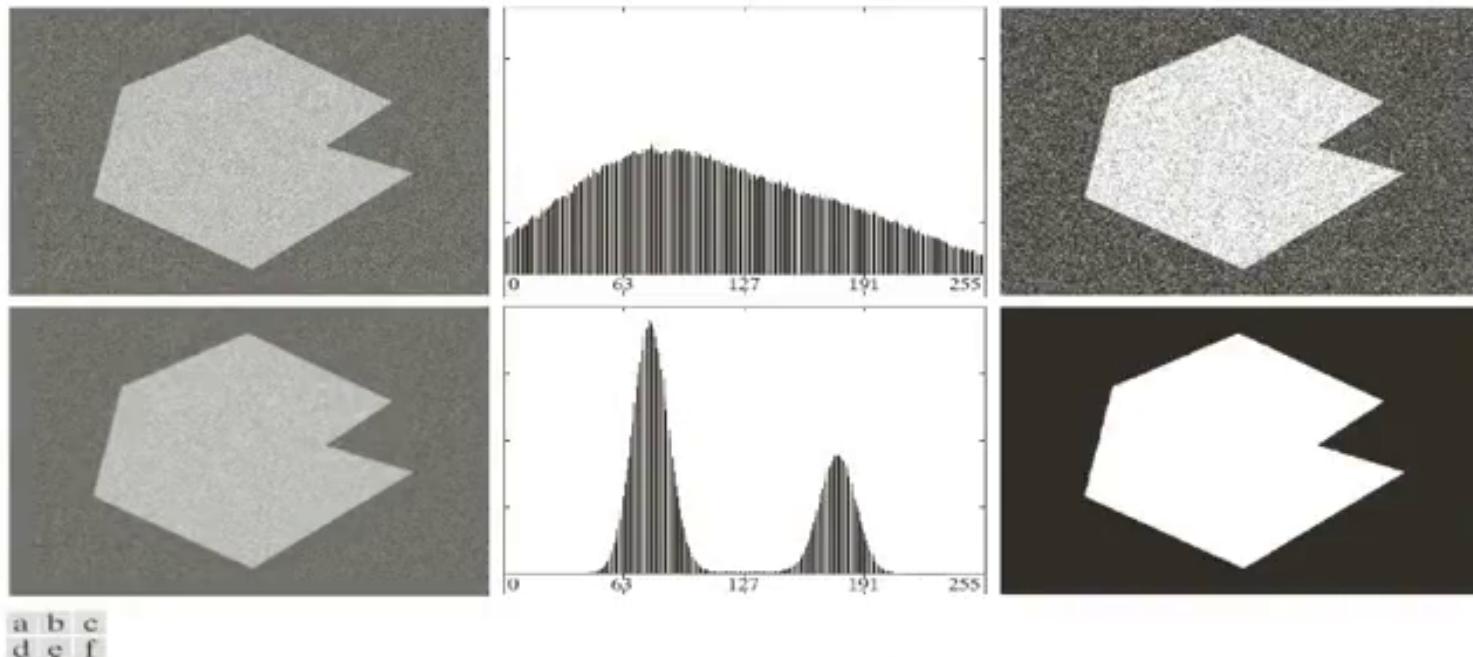


FIGURE 10.40 (a) Noisy image from Fig. 10.36 and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a 5×5 averaging mask and (e) its histogram. (f) Result of thresholding using Otsu's method.

Minimize the *weighted within-class variance*:



$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

$$q_1(t) = \sum_{i=1}^t P(i)$$

$$q_2(t) = \sum_{i=t+1}^I P(i)$$

$$\mu_1(t) = \sum_{i=1}^t \frac{iP(i)}{q_1(t)}$$

$$\mu_2(t) = \sum_{i=t+1}^I \frac{iP(i)}{q_2(t)}$$

$$\sigma_1^2(t) = \sum_{i=1}^t [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^I [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)}$$

Minimize the *weighted within-class variance*:



$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

$$\sigma^2 = \underbrace{\sigma_w^2(t)}_{\text{Within-class}} + \underbrace{q_1(t)[1 - q_1(t)][\mu_1(t) - \mu_2(t)]^2}_{\text{Between-class}}$$

Current Folder: /users/guillermo_sapiro/Documents/MATLAB

Shortcuts ▾ How to Add ▾ What's New

Current Folder

« MATLAB Name ▾

- gb.pdf
- rgb.pdf
- untitled2.jpg
- untitled3.jpg

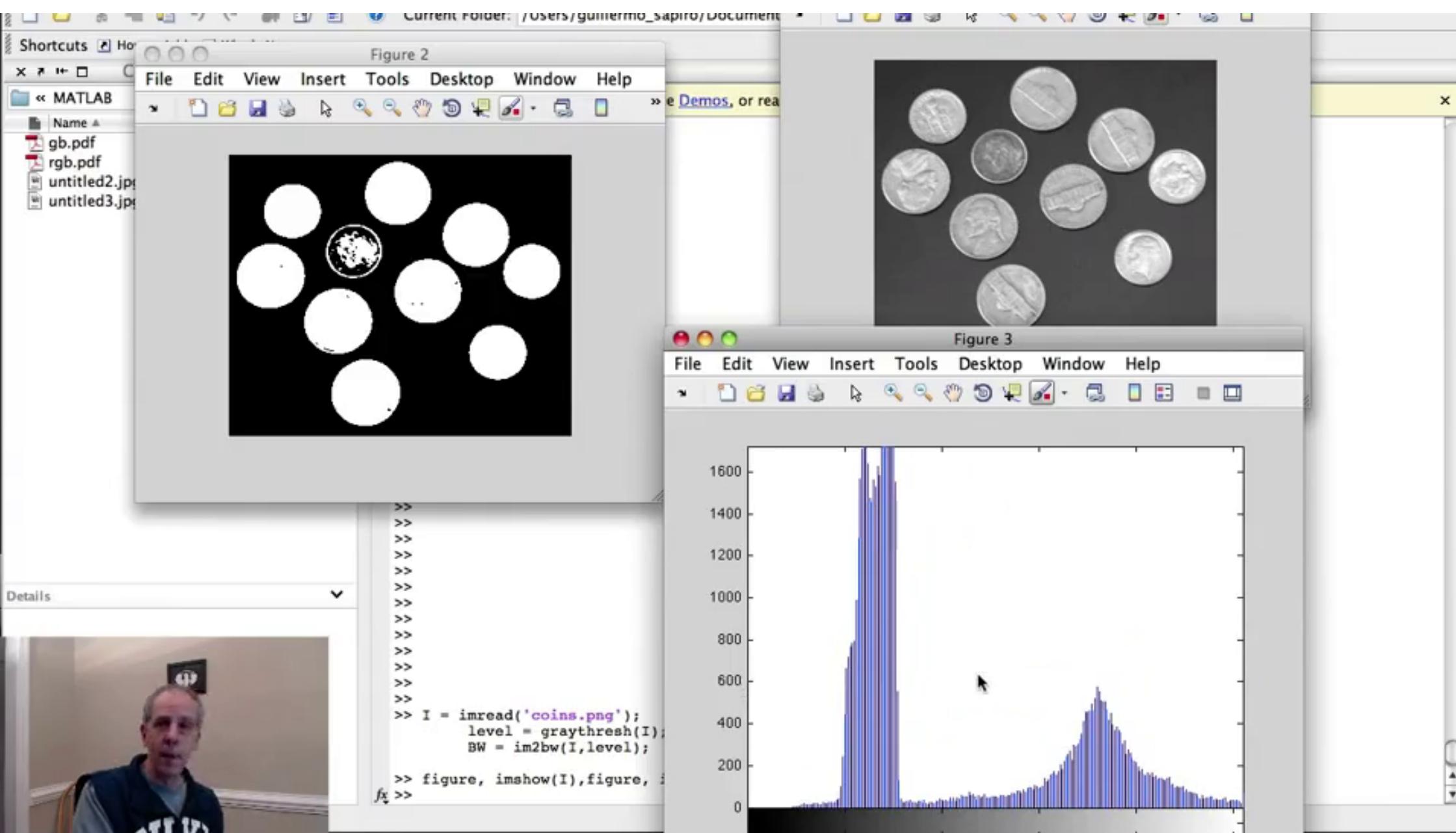
Details



Command Window

New to MATLAB? Watch this [Video](#), see [Demos](#), or read [Getting Started](#).

```
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>>
>> I = imread('coins.png');
>> level = graythresh(I);
>> BW = im2bw(I,level);
>> figure, imshow(I), figure, imshow(BW), figure, imhist(I)
```



Automatic Segmentation is Tough!



By Doolittle

Automatic Segmentation is Tough!

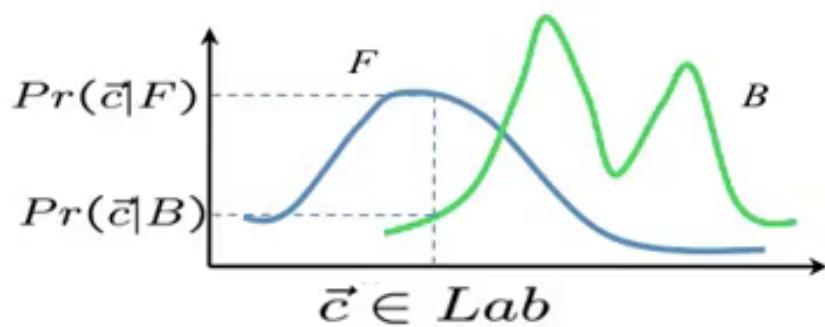


Interactive image segmentation



Step1 – Feature Distribution Estimation

- Estimate the color distribution on scribbles
- Each pixel is assigned a probability to belong to F or B:

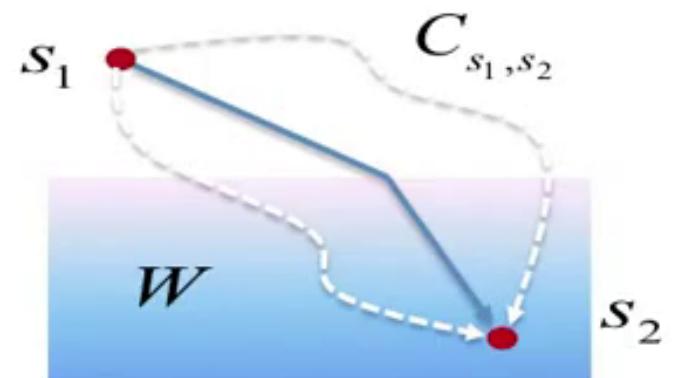


$$P_F(x) = \frac{Pr(\vec{c}_x|F)}{Pr(\vec{c}_x|F) + Pr(\vec{c}_x|B)}$$

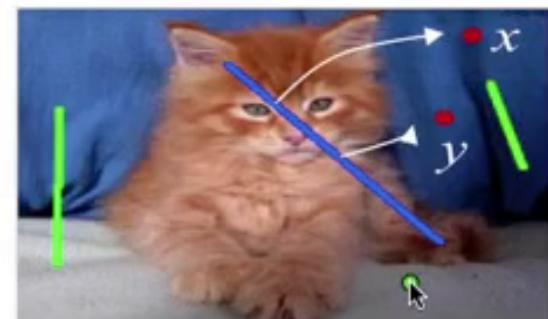
Step2 – Weighted Distance Transform

Weighted Geodesic Distance

$$d(s_1, s_2) := \min_{C_{s_1, s_2}} \int_{C_{s_1, s_2}} W ds$$



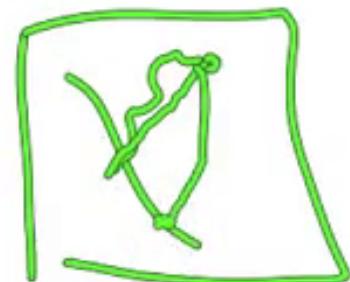
- Computed in linear time!



Weighted Distance Transform (cont'd)

$$W := |\nabla P_F(x) \cdot \vec{C}'_{s_1, s_2}(x)|$$

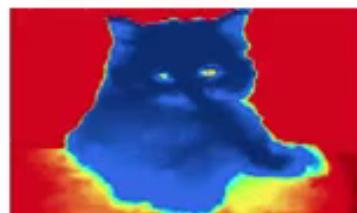
$$D_l(x) := \min_{s \in \Omega_l} d(s, x), \quad l \in \{F, B\}$$



- Pixels are classified by comparing $D_F(x)$ and $D_B(x)$



$P_F(x)$



$D_F(x)$



$D_B(x)$



Weighted Distance Transform



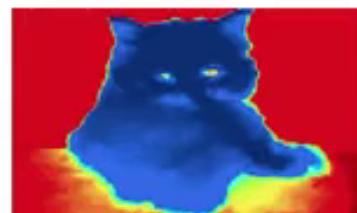
$$W := |\nabla P_F(x) \cdot \vec{C}'_{s_1, s_2}(x)|$$

$$D_l(x) := \min_{s \in \Omega_l} d(s, x), \quad l \in \{F, B\}$$

- Pixels are classified by comparing $D_F(x)$ and $D_B(x)$



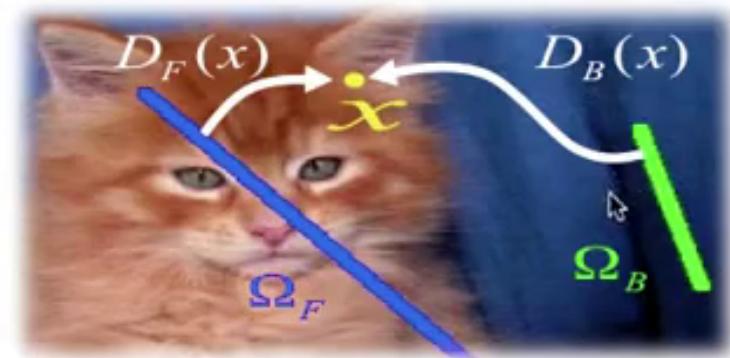
$P_F(x)$



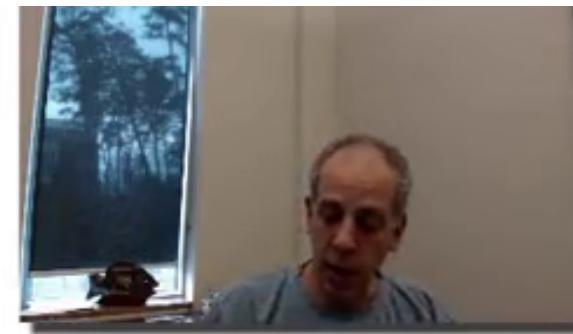
$D_F(x)$



$D_B(x)$



Weighted Distance Transform



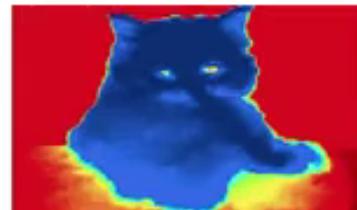
$$W := |\nabla P_F(x) \cdot \vec{C}'_{s_1, s_2}(x)|$$

$$D_l(x) := \min_{s \in \Omega_l} d(s, x), \quad l \in \{F, B\}$$

- Pixels are classified by comparing $D_F(x)$ and $D_B(x)$



$P_F(x)$



$D_F(x)$

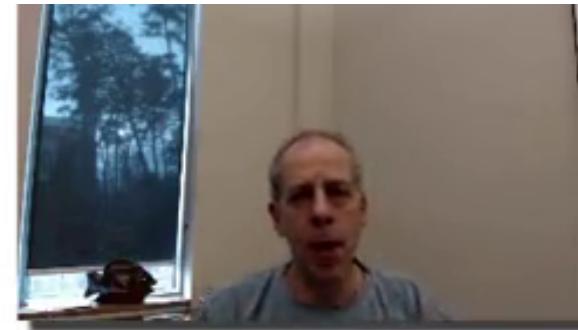


$D_B(x)$

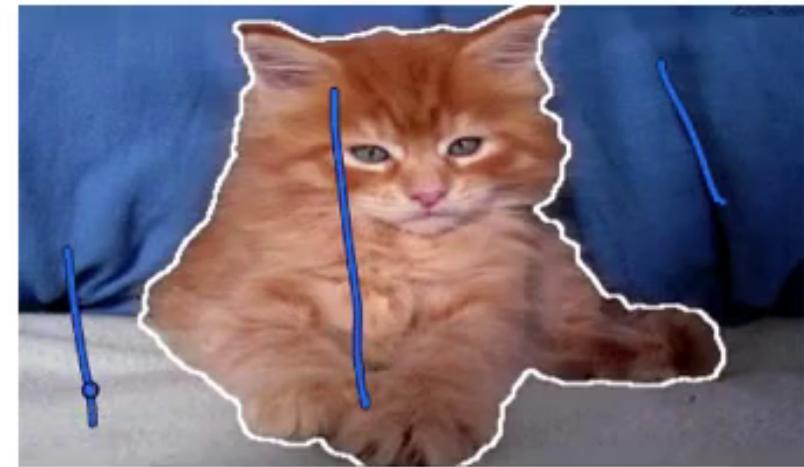


binary segmentation

Step3 – Refine



- Automatically create a narrow band and new scribbles.
 - Band boundaries serve as “new scribbles”

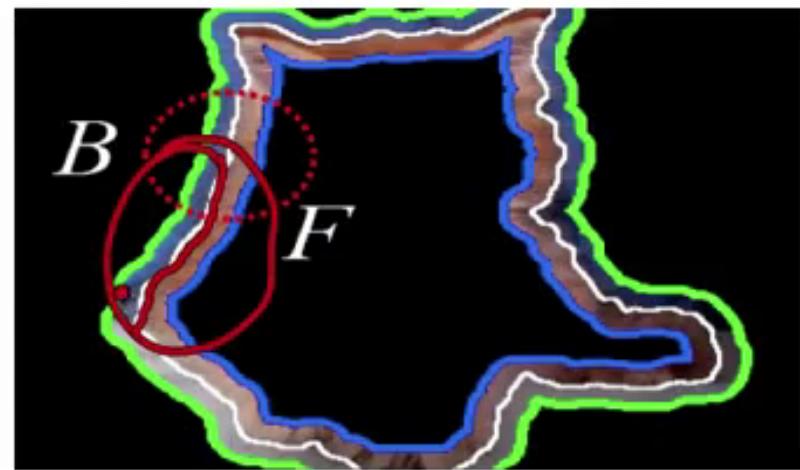


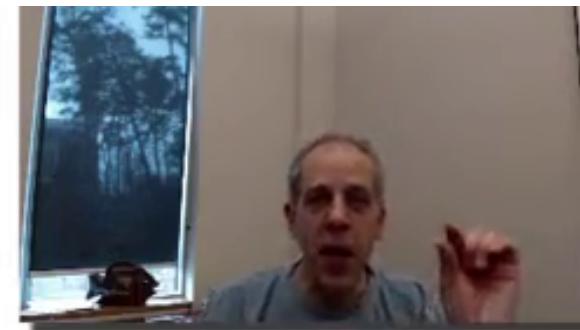


Step3 – Refine

- Automatically create a narrow band and new scribbles.
 - Band boundaries serve as “new scribbles”

P
F



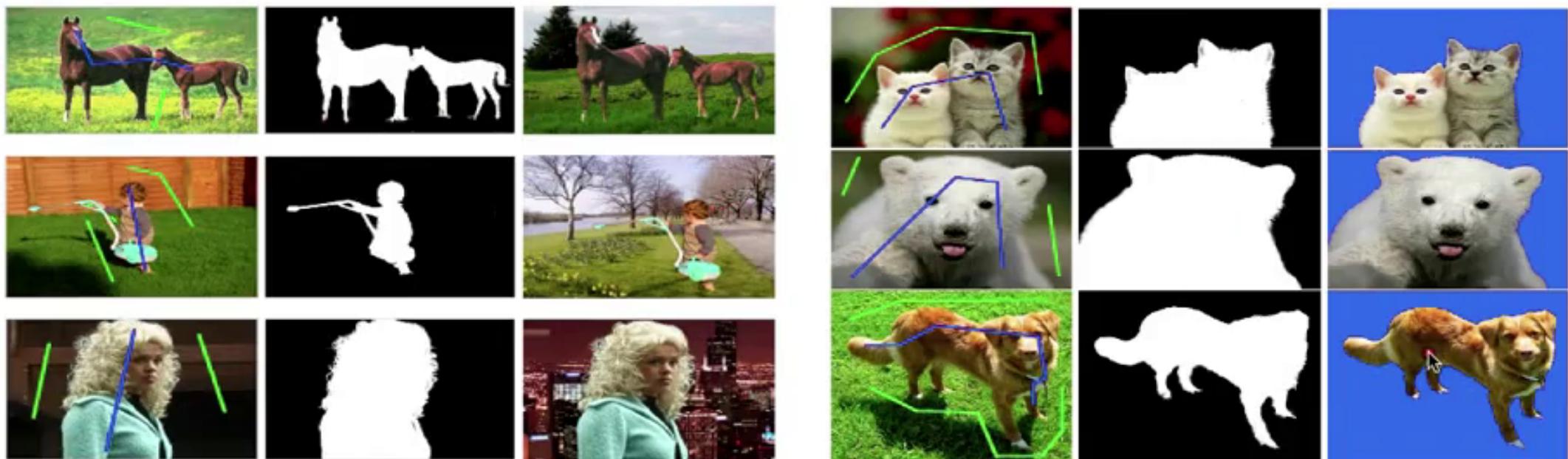
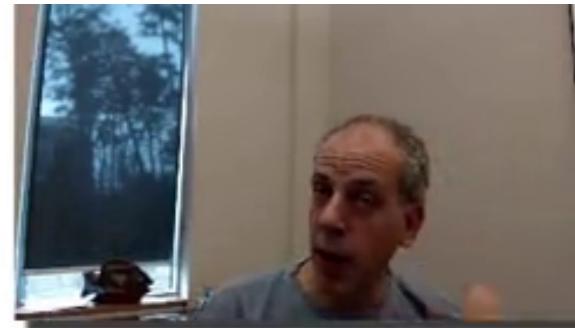


Step3 – Refine

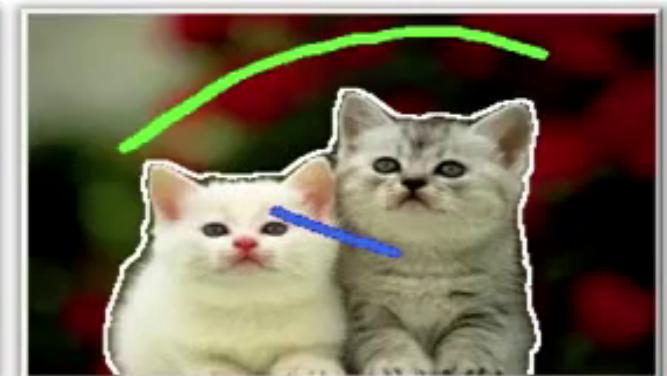
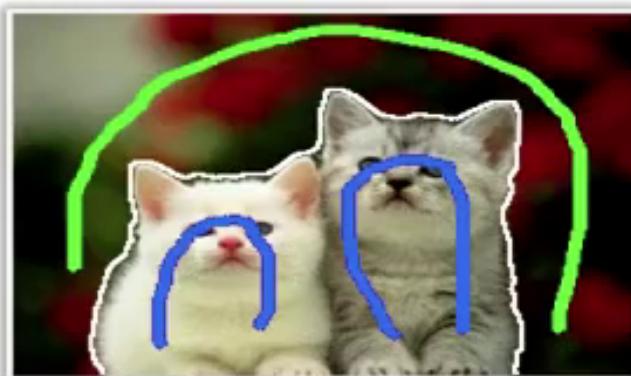
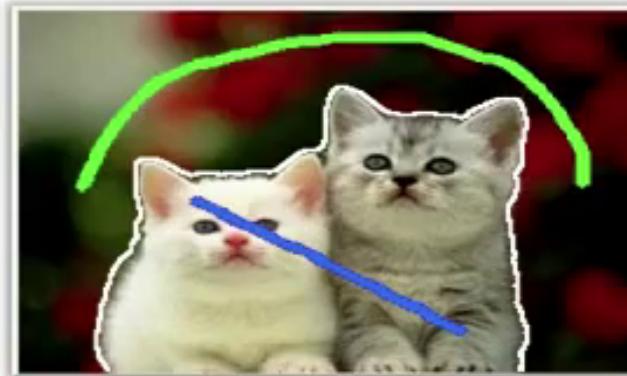
- Automatically create a narrow band and new scribbles.
 - Band boundaries serve as “new scribbles”



Examples

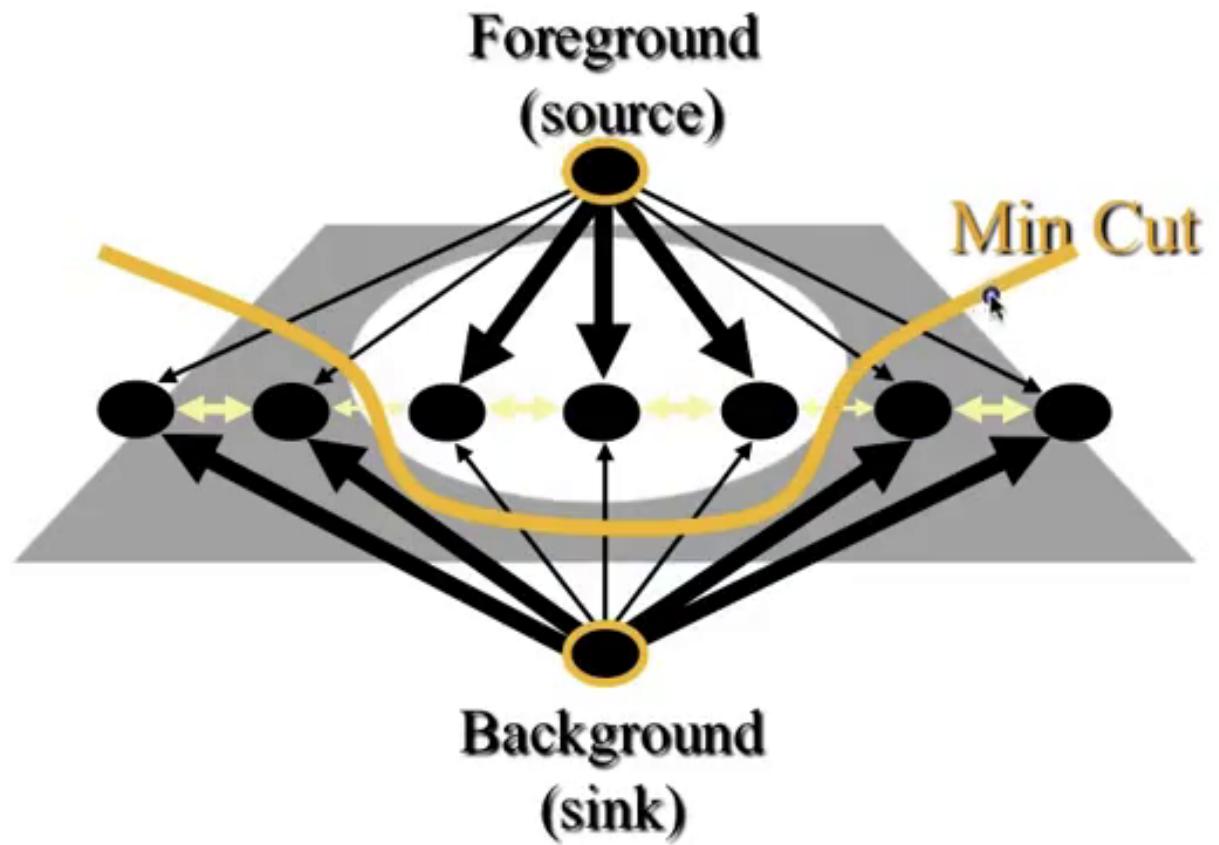
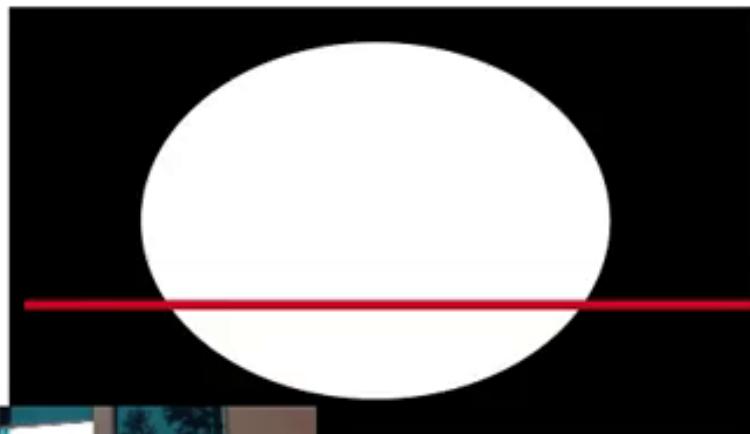


Scribble Robustness



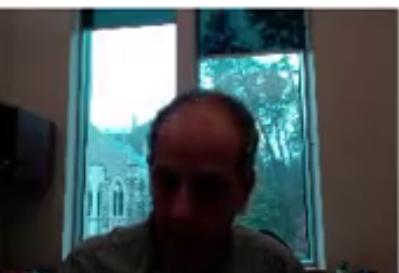
Graph Cuts - Boykov and Jolly (2001)

Image



Cut: separating source and sink; Penalty: collection of edges

Min Cut: Global minimal energy in polynomial time



Photomontage



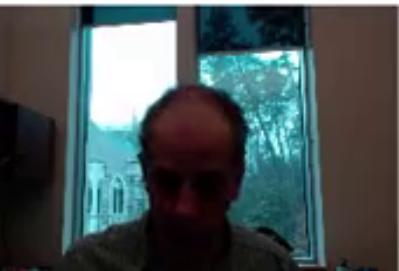
Courtesy of Carsten Rother

Photomontage



Courtesy of Carsten Rother

Photomontage



Courtesy of Carsten Rother

Photomontage



Courtesy of Carsten Rother

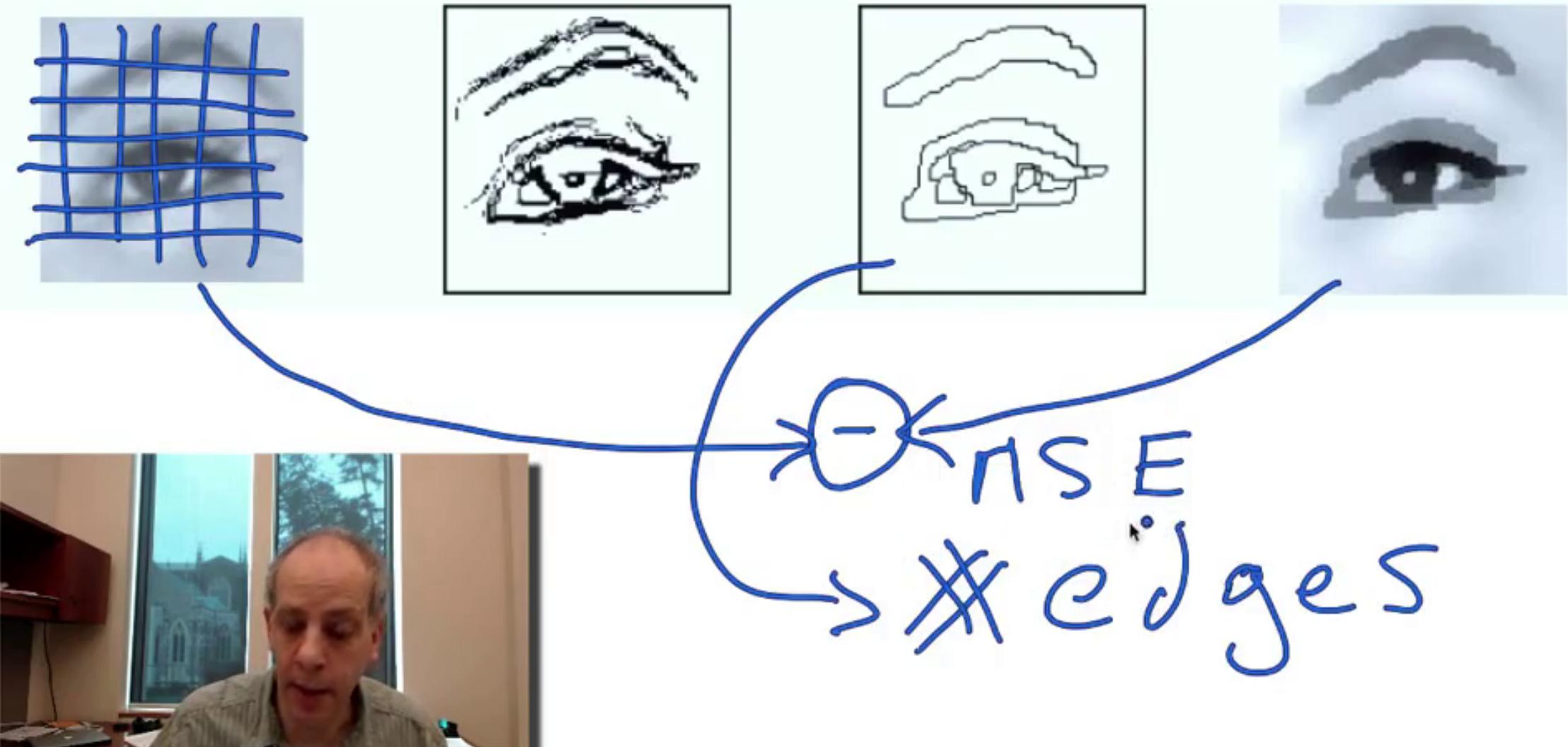


Image courtesy of Alan Yuille

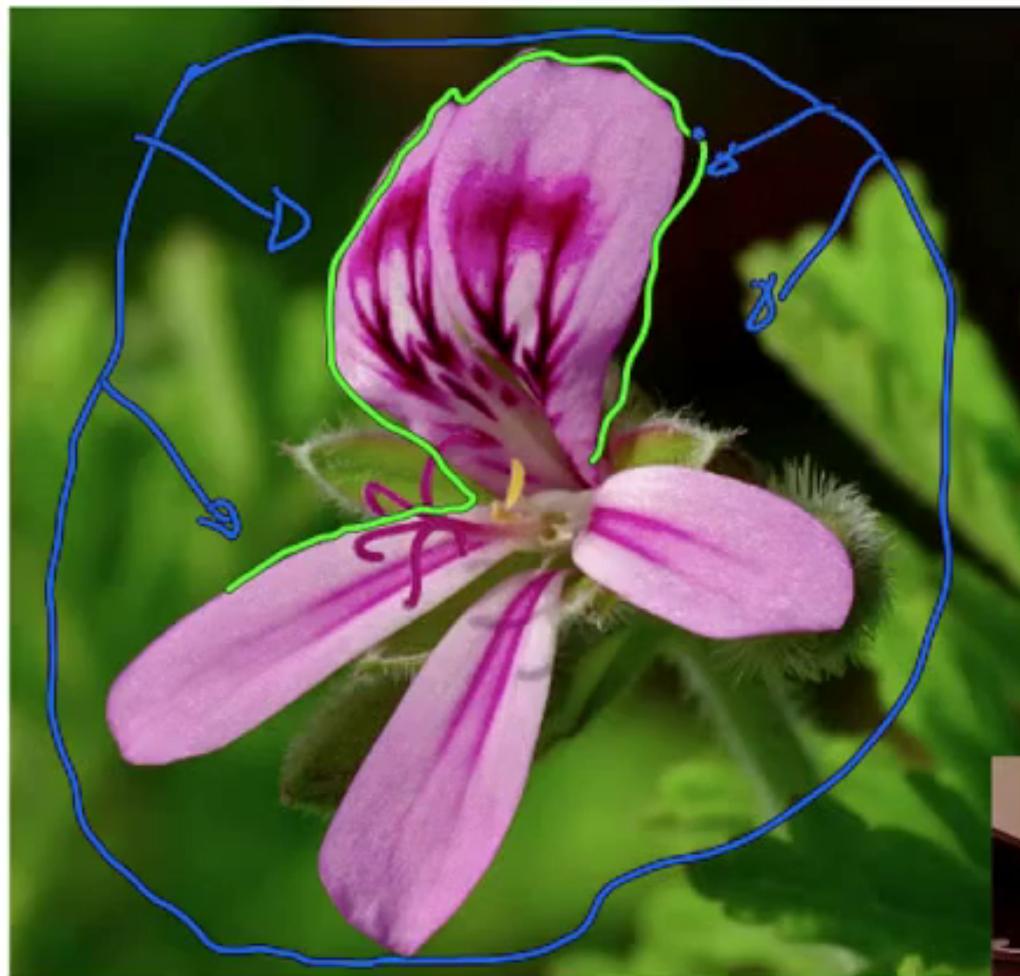
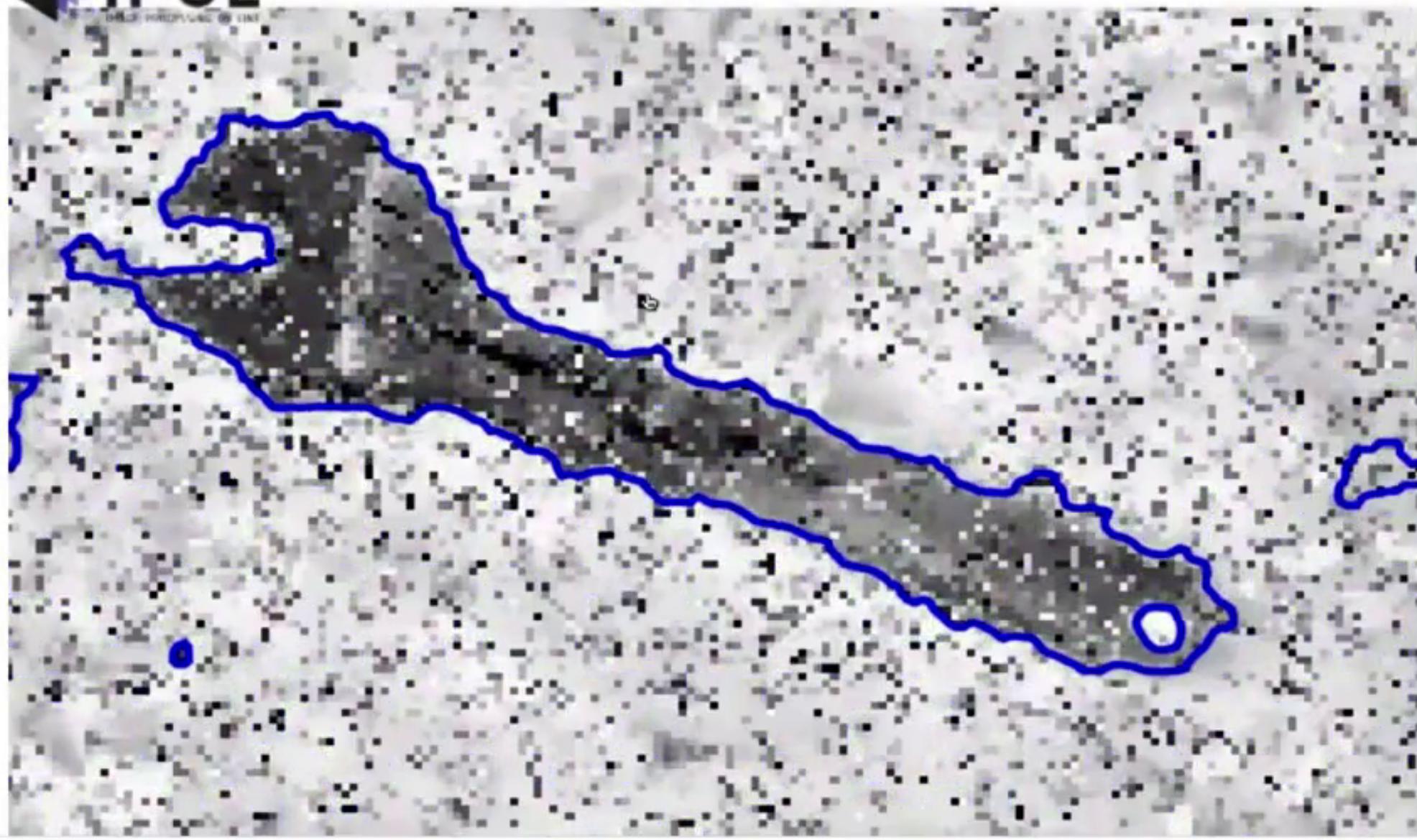
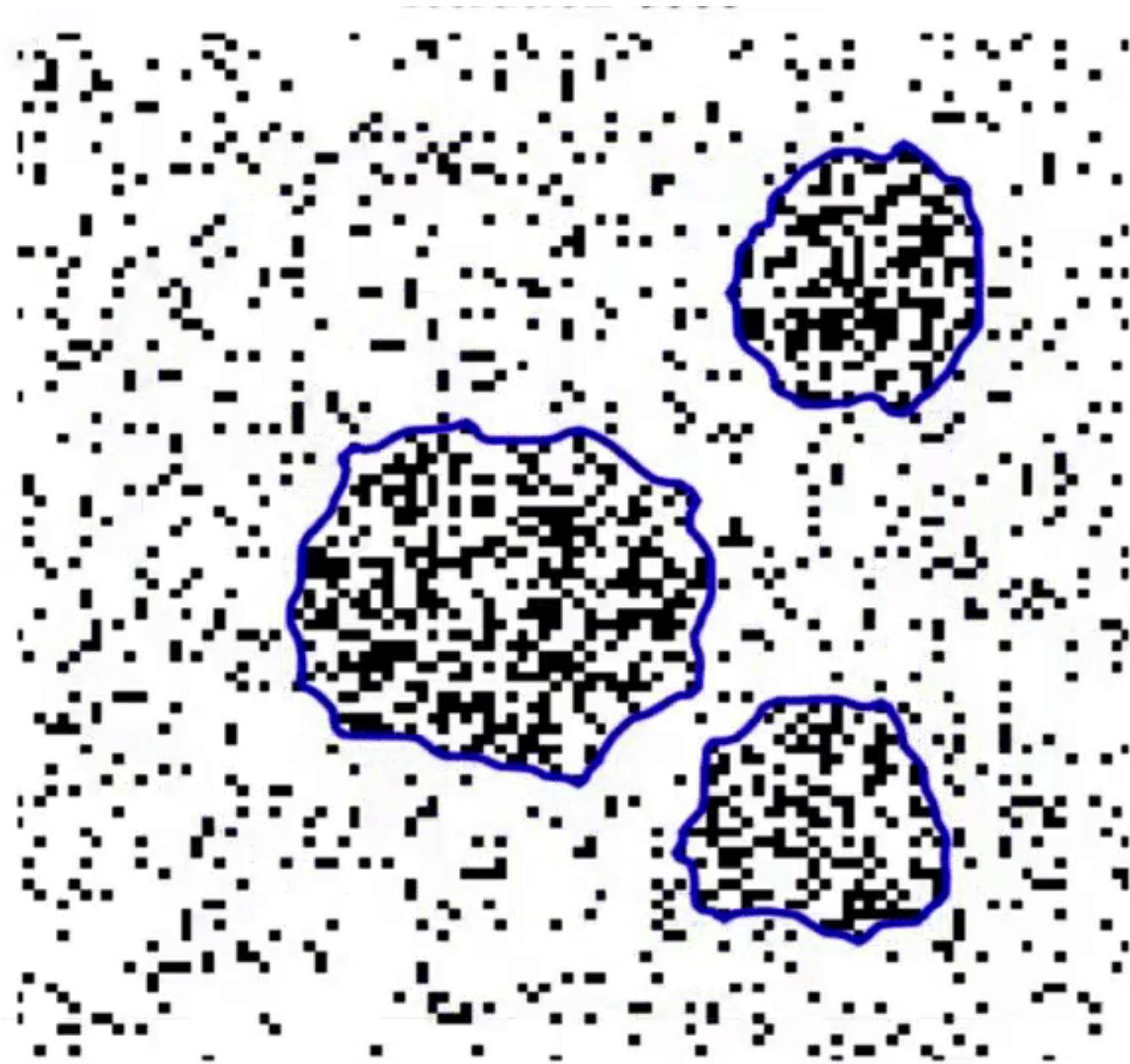


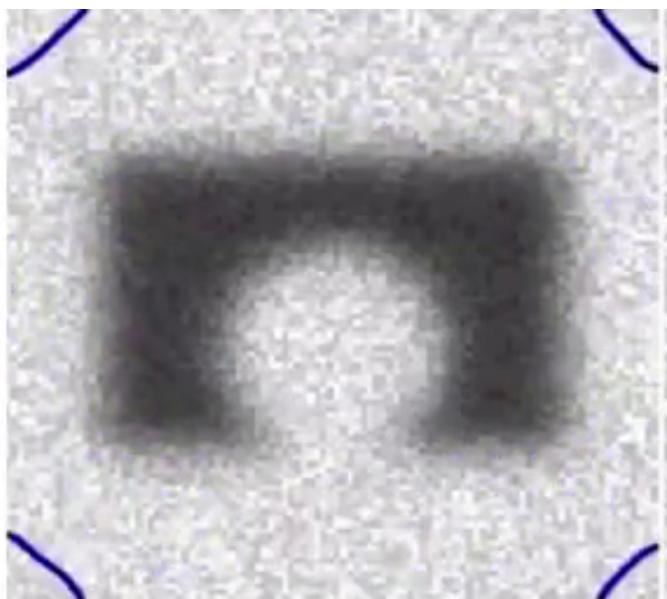
Image courtesy of ipol.im

IPUL

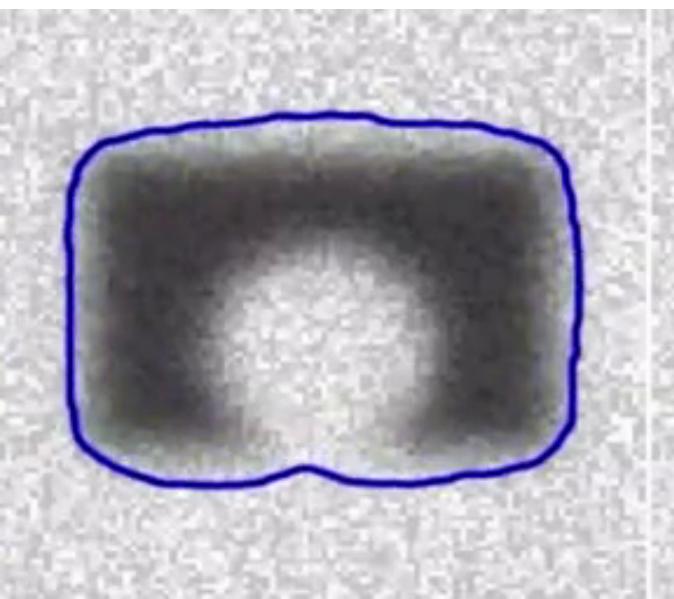
TOOLBOX



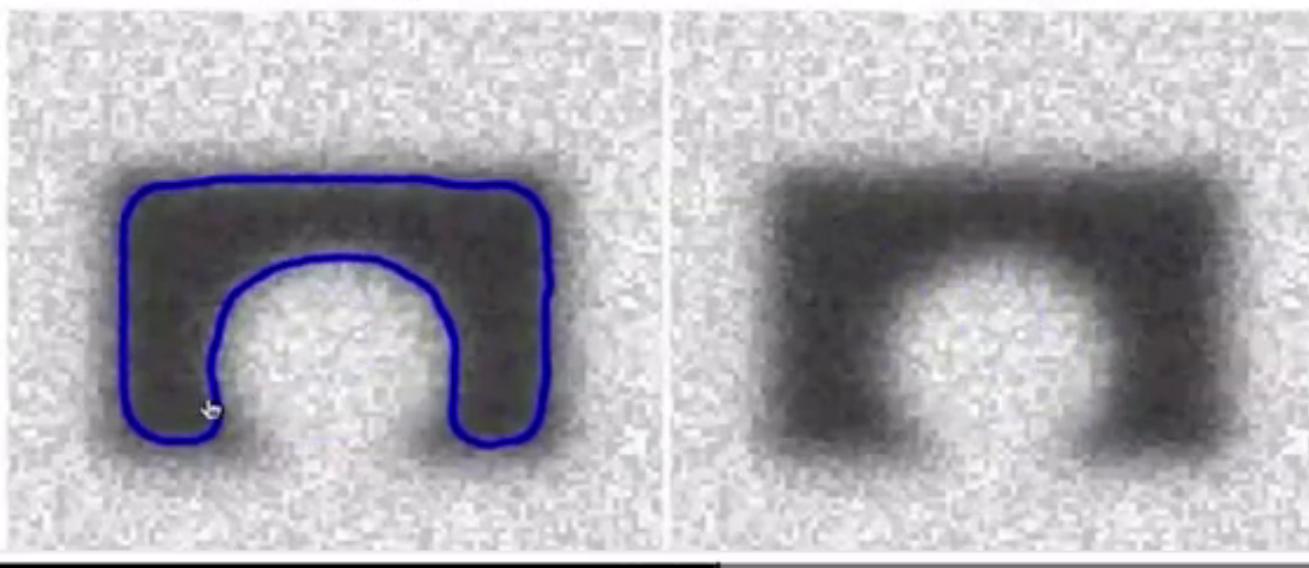


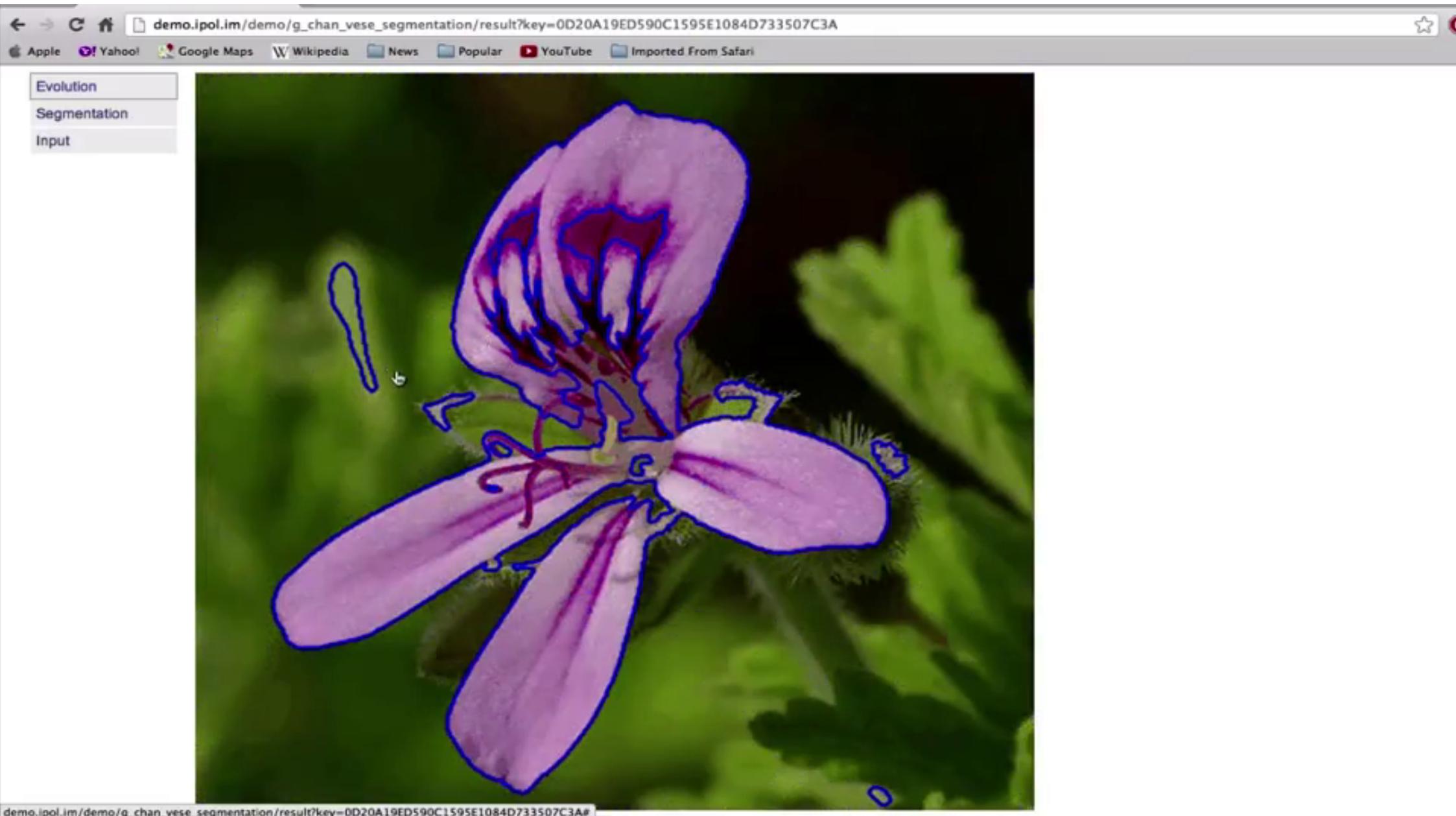


$\nu = 0.1$



$\nu = 0.3$





demo.ipol.im/demo/g_chan_vese_segmentation/result?key=0D20A19ED590C1595E1084D733507C3A#

Behind Roto Brush: Adobe's Video Segmentation



After Effects

*Images Courtesy of: ACM – SIGGRAPH and ECCV; Xue Bai
Jue Wang; David Simons; Adobe*

Image and Video Processing: From Mars to Hollywood with a Stop at the Hospital

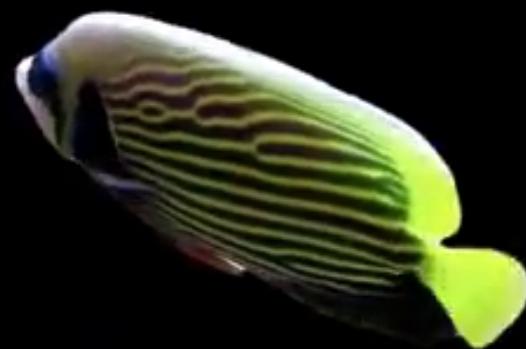
Guillermo Sapiro

Duke
UNIVERSITY



Problem: Interactive Video Segmentation

- Pixel level accuracy
- Minimal user intervention
- Interactive real-time



High-Quality Video Object Cutout



Original



Challenges



overlapping color distributions



weak boundaries



topology changes, dynamic backgrounds



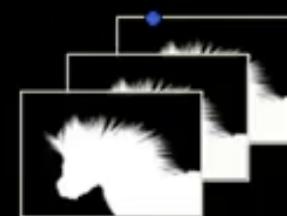
Features

- Accuracy
 - work with complicated scenes
- Robustness
 - on diverse data
- Practical workflow
 - easy to converge/interact
- Computational efficiency
 - Faster than 2 frames per second

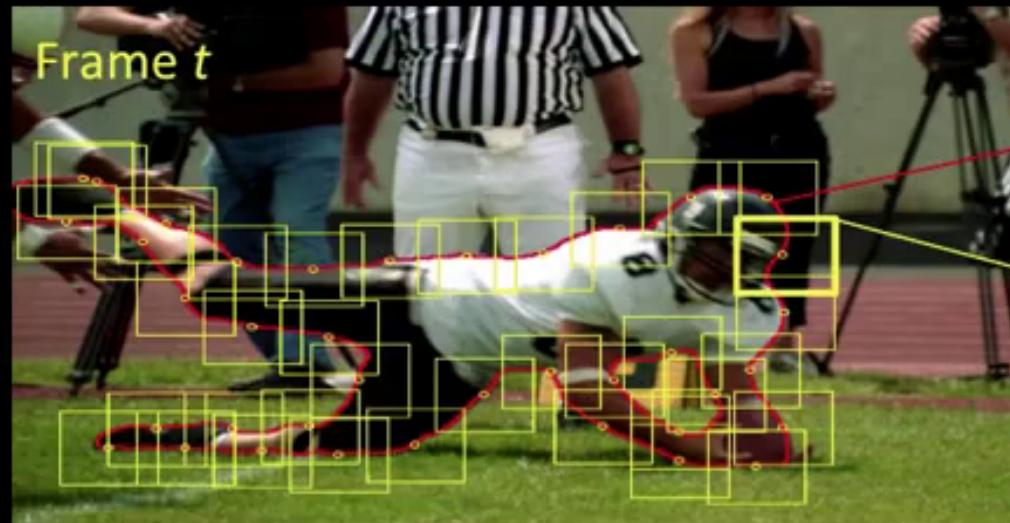


Algorithm Overview

- Localized classifiers
- Multi-frame propagation
- Local correction
- Post-processing

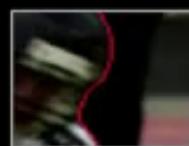


Localized Classifiers



Existing
segmentation

Local classifier
window

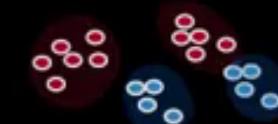


Shape Prior

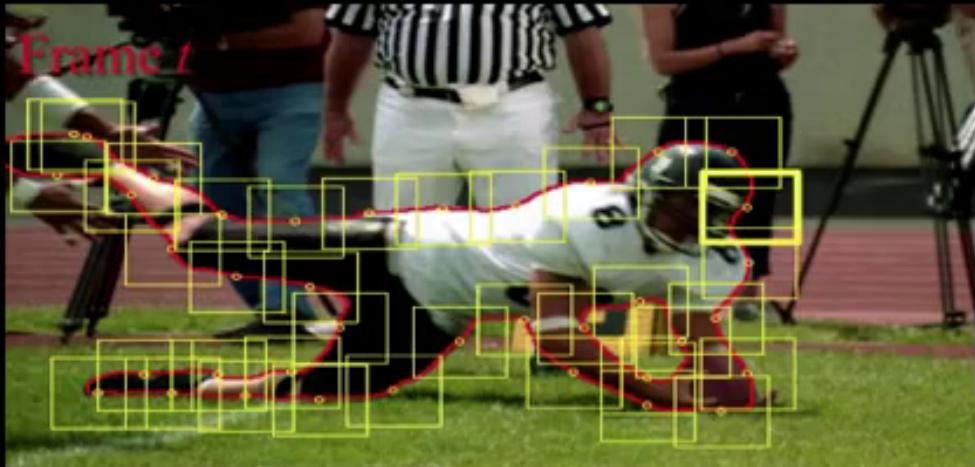


Local window i

Local classifier i



Localized Classifiers

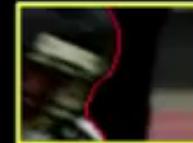


Frame $t+1$



Graph cut

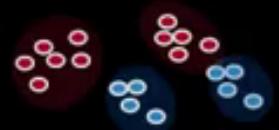
training



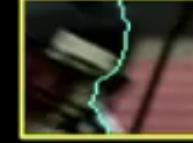
shape



GMM's



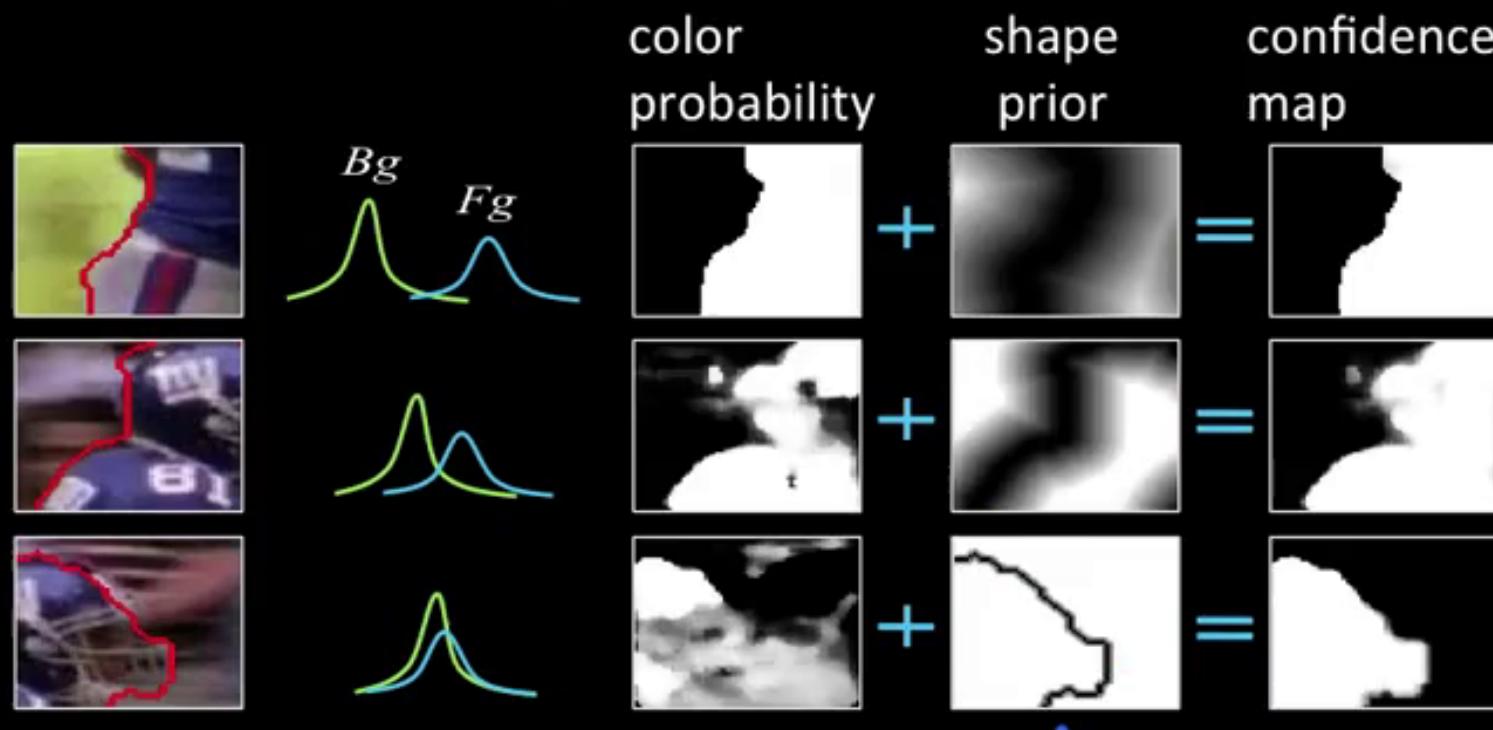
testing



Integration

Adaptive Color-Shape Integration

- If colors are separable, trust color model
- If not, trust shape prior



Single-Frame Propagation Example



frame t (segmented)



frame $t+1$ 



global color model



local color models

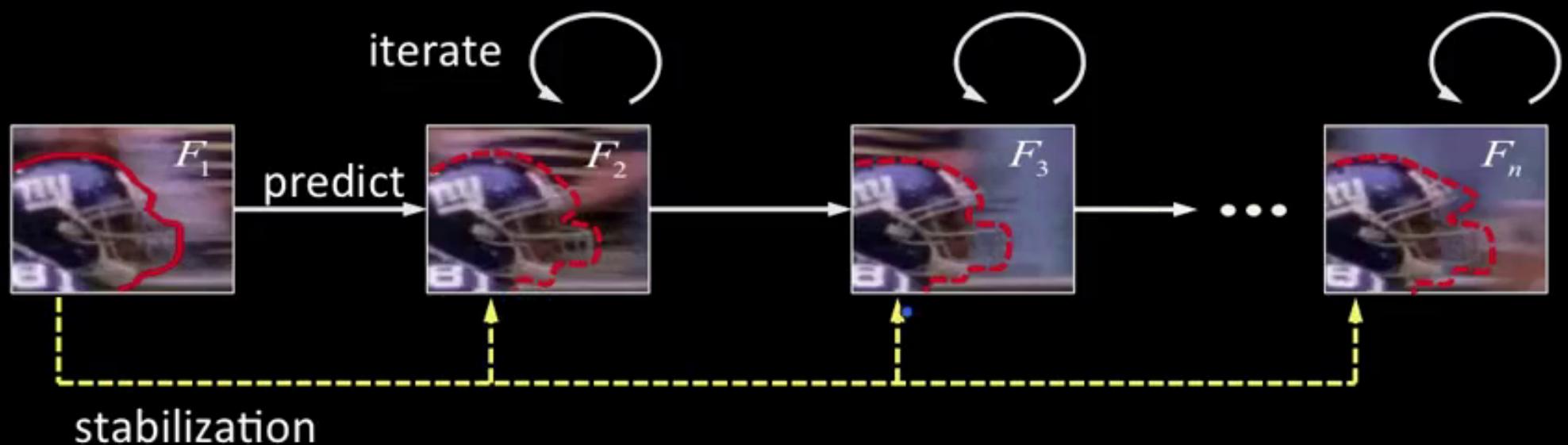


local color models +
local shape prior

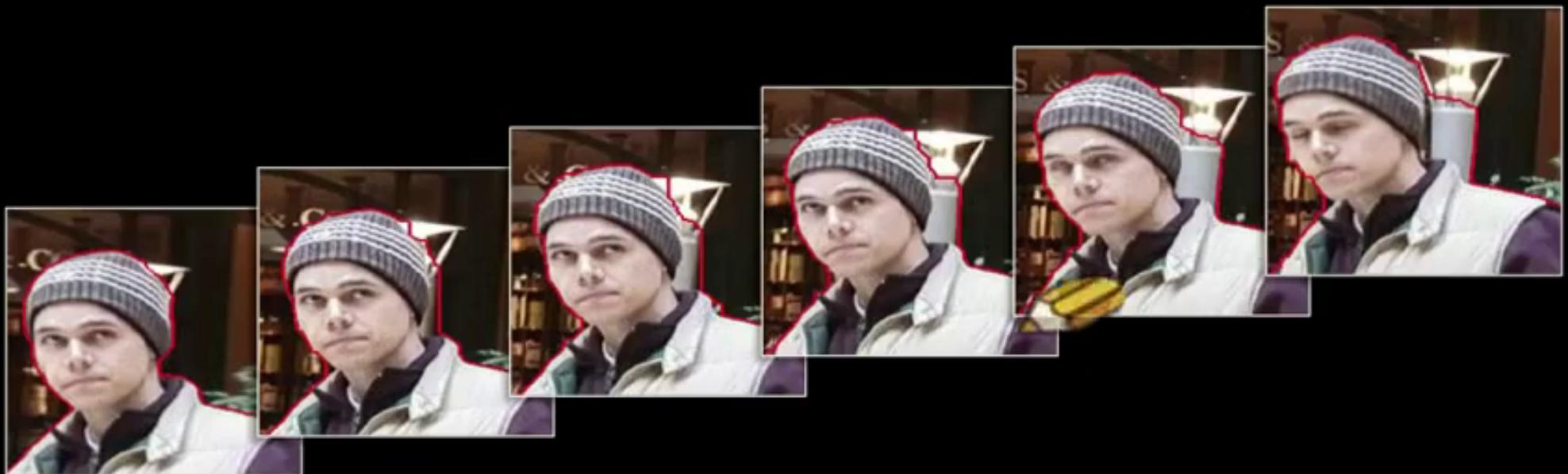


Multi-Frame Propagation

- Practical workflow
- Stabilization



Spatial-Temporal Local Correction



Spatial-Temporal Local Correction



Spatial-Temporal Local Correction

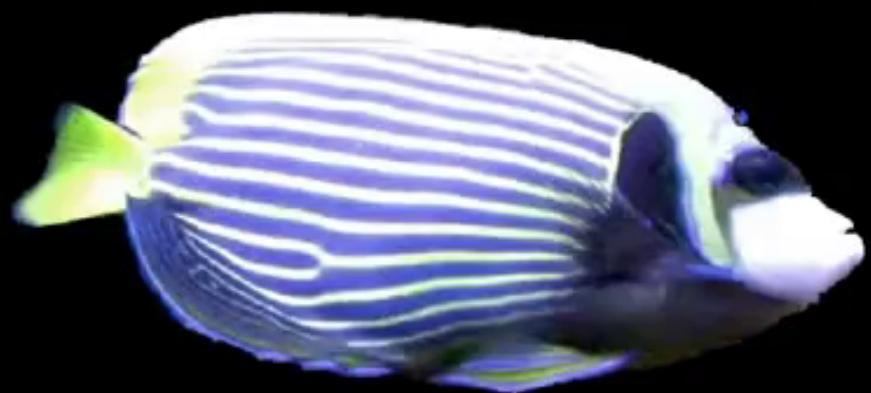


Affects only a few local windows

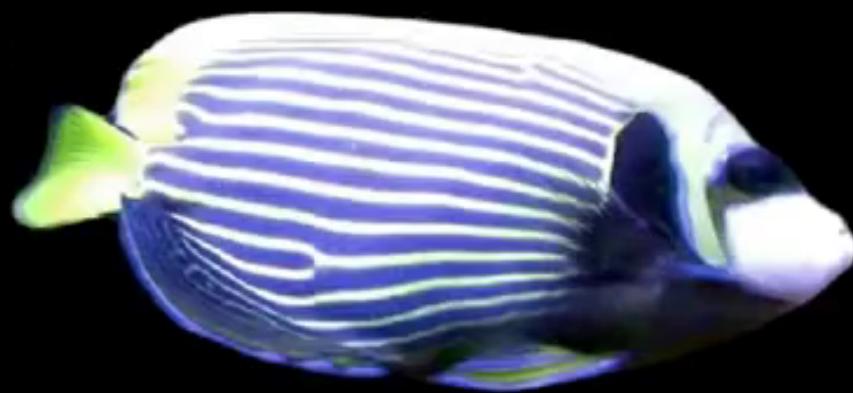
Propagates changes temporally



Post-processing



before



after



Results



720×406 185 frames

Results



720×576 50 frames

Results



original



848×480 232 frames



filter



relight



composite



Original

Composite



473×588



Original



Composite



504×380