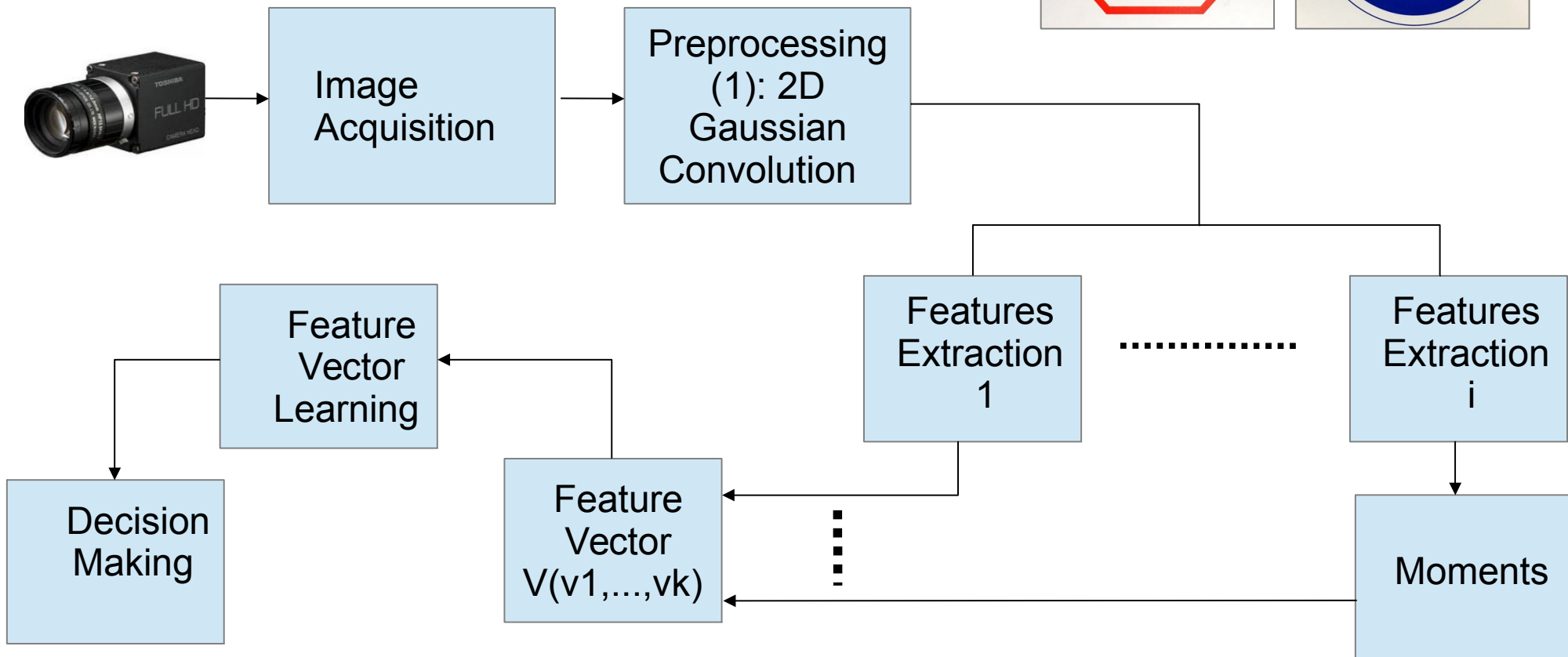


lec5-1-floodfill-2018-2-22.ppt

Harry Li, Ph.D.

Intro to Moments for Objects Recognition

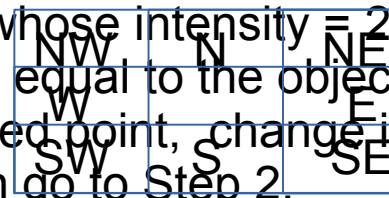
Objectives: Develop a technical to detect different shaped, different color, different size objects



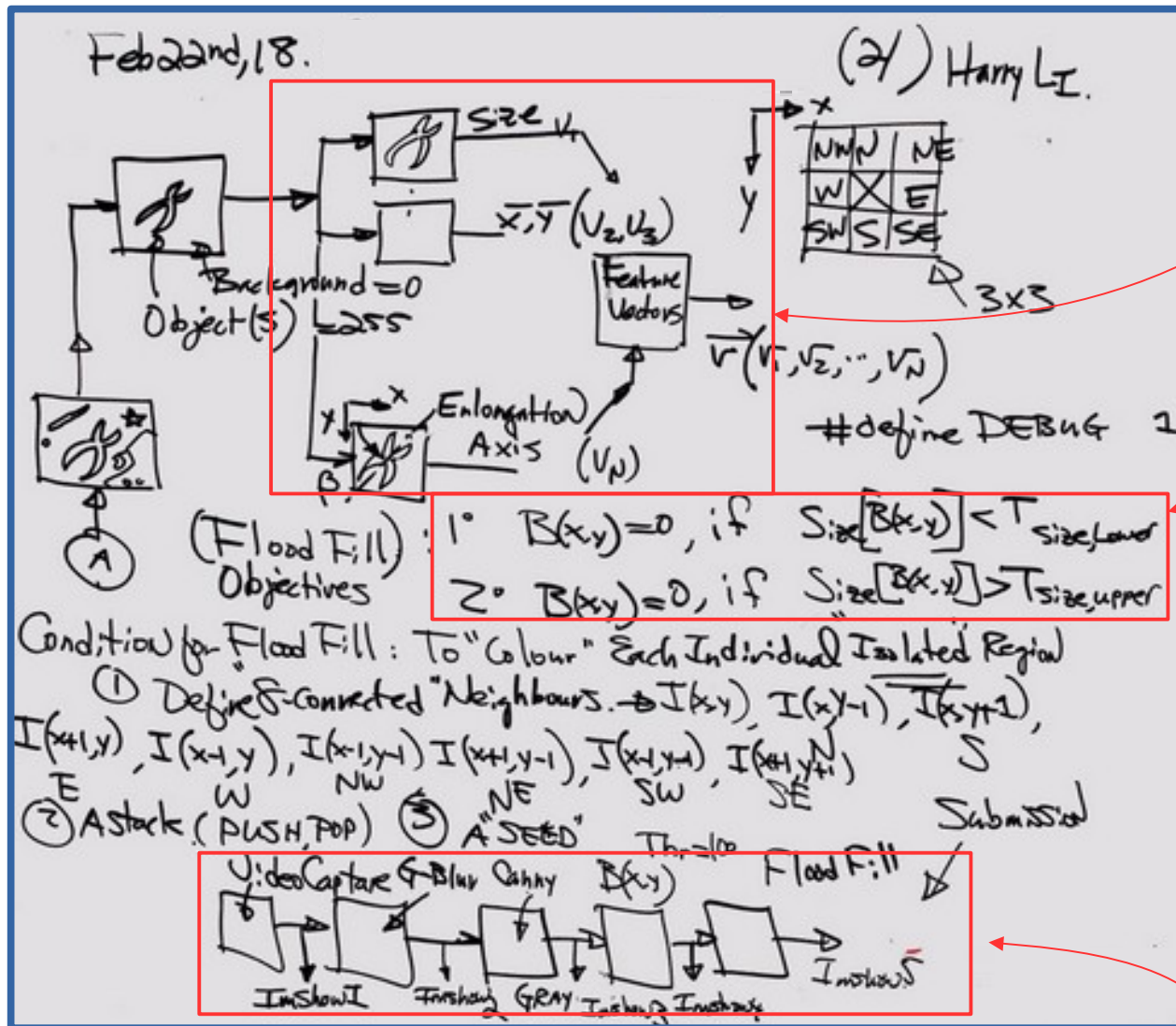
Flood Fill Algorithm

Note: Define 8-connected neighbors, N, NE, NW, S, SE, SW, E, W;

1. Find a “seed” point from an object (e.g., whose intensity = 255), change it to new color (label), and save this seed location (x,y);
2. Then visit N to see if it is the same color as the previous color of the seed (before seed color changed to “1”), if yes, then push its location (x,y) into a stack, and then change its color to the same new color (label) as the new color of the seed to indicate this pixel being visited; continue this process till all the 8-connected neighbors were visited;
3. Then pop up (x,y) of a saved pixel from the stack, treat this pixel as a new seed point, go to Step 2, and continue.
4. If the stack is empty then it means this connected region has been colored, so go back to step 1, read the seed location (x,y) and continue to scan the image in a left to right, top to bottom fashion. If no new pixel whose intensity = 255 is found, then flood fill is finished. Otherwise till the next pixel equal to the object (its intensity = 255) is found, and make this pixel as a new seed point, change it to new color (label), and save this seed location (x,y); Then go to Step 2.



Preprocessing: Binarization Before Flood Fill



Preprocessing steps: (1) histogram equalization, (2) Gaussian; (3) Canny edge detection; (4) binarization; (5) flood fill filtering; Then (6) moments computation

Hand Calculation Example

Feb 22nd, 18.

Harry Li 3/

Example:

	x	
y	0	100 100 255
	0	0 0 100
	0	100 0 0
	0	20 20 20 0

$I(x,y)$ ① Binarize the Image.

Given $B(x,y) = \begin{cases} 255 & \text{if } I(x,y) \geq \text{Thr} (=40) \\ 0 & \text{o/w} \end{cases} \dots (1)$

Step 1. Scan @ (0,0)

Left 2 Right, TAB.

Flood Fill to "Glowr"

	x	
y	0	255 255 0
	0	0 0 255
	0	255 0 0
	0	0 0 0

$B(x,y)$

@ (1,0), $B(x,y) = 255 \rightarrow$ "Seed" \rightarrow Change colour to "1".

Push $(x,y) = (1,0)$ into A Stack.

Step 2. Visit "N" pixel, outside, done

Visit "NE" outside, No Action.

Visit "E", check its Intensity, $=255$, Change to 1, Push $(x=2, y=0) \rightarrow$ Continue for the Rest, No Action

Step 3

Pop from Stack, $(x=2, y=0)$ check "N", outside, No Action, "NE" outside, No Action, "E", No Action, "SE" = 255, Change to 1, push $(x=3, y=1)$

"S", No Action, "SW" No Action
Check "W" = 1, A/ready Checked.
No Action.

Step 4. pop from stack, $(x=3, y=1)$ (New Seed)

Check "N" = 0, N.A.
"NE", outside, N.A., "E" outside, N.A.
"SE", outside, N.A., "S" = 0, N.A.
"SW" = 0, N.A., "W" = 0, N.A., "NW" = 1, Already Done N.A.

Step 5. Top from Stack $(x=1, y=0)$

Scan the next pixel in the fashion of L2R, TAB.

@ $(x=0, y=2) = 0$, N.A., Next, @ $(x=1, y=2) = 255 \rightarrow$ Change colour to "2", Push into the Stack,

Continue to check the Rest of the Neighbours, all equal to 0, N.A.

Step 6. pop from the stack, $(x=1, y=2)$

Check "N" = 0, N.A., "NE" = 0, N.A., "SE" = 0, N.A., "S" = 0, N.A., "SW" = 0, N.A., "W" = 0, N.A., "NW" = 0, N.A.

Step 7. pop from stack, "999" Empty, Move ON (Scanning)

		1	1	1
	2			

Flood Fill For Color Images

Today's Topics:

1° Floodfill Implementation.

2° Feature Extraction Based on Moments Analysis.

If "yes" (Seed Colour and "Neighbour" colour Are Equal)
Then, Assign New ID (Index, e.g. "255")

(2) Python Code 2.1 Python 2 & 3 Compatible.

2.2 Use "sys" to Input Image

2.3 Dimension of the Input $B(x,y)$

Example: OpenCV. Python Code. `$python floodfill.py my.png`

(1) For Colour Images. (Connection to Binary

Images - Example in Last Lecture)

Binary, Object Colour ("1" or "255")
"Seed" (Background Colour "0")

if "Seed" colour = "Neighbours" Colour → Label
this neighbour as valid Neighbour, New
ID (Index)

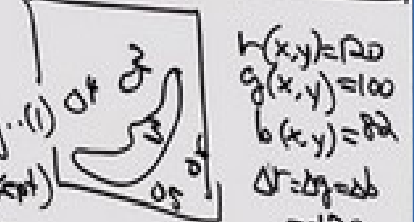
Colour(1,1) "Seed" Selection → mouse clicks (x,y) , select
Image $I(x,y) = (r(x,y), g(x,y), b(x,y))$

(1.2) Flood fill, By checking "Neighbours" colour \neq "Seed" colour

Image: `h, w = img.shape[:2]`

Table 1

	Image $I(x,y)$	Filtering By Size
F.F. (Binary)	$B(x,y)$ Binary Only	Threshold Low (T_L) Threshold upper (T_u) $Size[B(x,y)] \in [T_L, T_u] \dots (1)$ then $B(x,y) = 255$ (kept)
F.F. (Color)	$I(x,y)$ colour.	Color Segmentation. (1) Every colour keep or eliminate (2) "Seed" colour $(r(x,y), g(x,y), b(x,y))$ (x,y) $(r(x,y) + 0.5g(x,y) + 0.5b(x,y) + 0.5)$



$h(x,y) = 120$
 $g(x,y) = 100$
 $b(x,y) = 80$
 $r = 0.5g + 0.5b$
 $= 90$

Flood Fill For Color Images

def myfunction():
 global seed-pt
 :
 update()
 Variable.

Note: Use float track to modify the dynamic Range.

① mouse click @ (x, y)
 $ONI(x, y)$
 "Seed" colour $(r(x, y), g(x, y), b(x, y))$

② User defined dynamic Range.
 $\Delta r = 120, \Delta g = 15, \Delta b = 15$

③ Change flood fill (color F.F.) in such a way

$B(x, y) \begin{cases} = 255 & \text{iff } (r(x, y) + \Delta r, g(x, y) + \Delta g, b(x, y) + \Delta b) \leq \Delta \\ = 0 & \text{iff } (r(x, y) - \Delta r, g(x, y) - \Delta g, b(x, y) - \Delta b) \geq \Delta \end{cases} \dots (1)$

$(r(x, y), g(x, y), b(x, y))$

$(r(x, y) - \Delta r, g(x, y) - \Delta g, b(x, y) - \Delta b)$

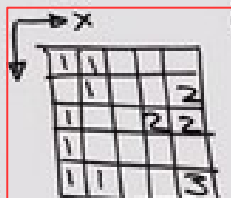
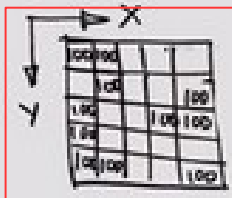
$(r(x, y) + \Delta r, g(x, y) + \Delta g, b(x, y) + \Delta b)$

cv2.floodFill() Color Images

March 1st, 2018. Harry Li

Homework: (2pts)

1. Generate 2 Test Pattern Images.
- (1.1) $I_1(x,y)$ Binary Image To Test "8-Connected" Binary Floodfill Algorithm.



OpenCV Floodfill Program.

- (1.2) $I_2(x,y)$ Binary Image.

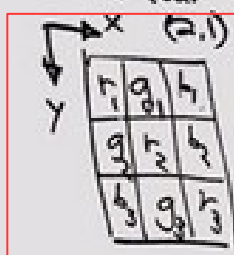
Floodfill Algorithm.

`cv2.floodFill(flooded, mask, seed, (255,255,255), (0,0,0), (100,100,100), flags)`

For Resulted White colour Lower v.g. b.

$K \times K$ (Test Result)

2. Blur Image $I_3(x,y)$



$h=100, r_2=200, r_3=255$
 $g, b \in [100, 200, 255]$

(2.2) Dynamic Range testing
 $[100, 200]$ $[200, 255]$
 For Red.

