

Programming Assignment 3: Image Segmentation

Dr. Tsung-Wei Huang

Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, UT

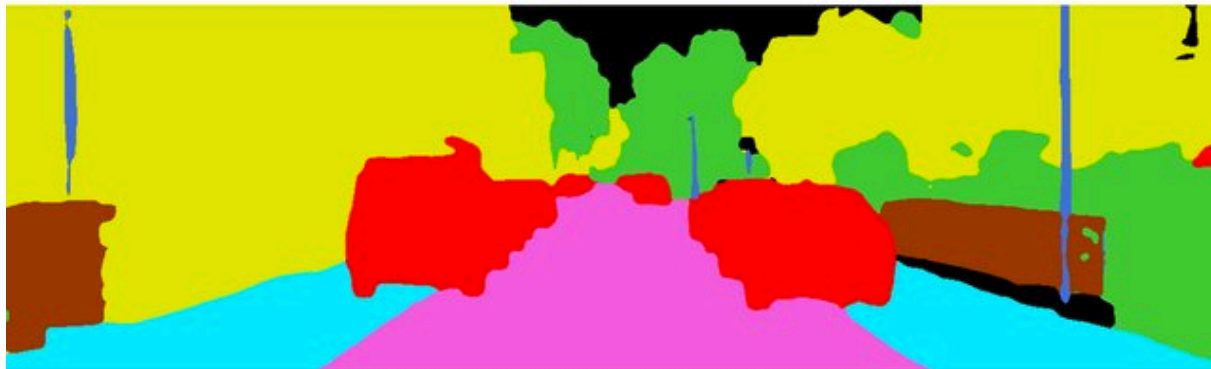










Goal

- ☐ Understand the image segmentation problem
- ☐ Relate segmentation problem of maxflow
- ☐ Write algorithms to solve the problem

Image Segmentation

- ❑ Separate the foreground from the background



 Road	 Sidewalk	 Building	 Fence
 Pole	 Vegetation	 Vehicle	 Unlabel

Setup

- ❑ Image is a grid of pixels
- ❑ Need to decide which pixels are in the foreground
 - ❑ Others are in the background
- ❑ Applications provide likelihood value of each pixel v
 - ❑ f_v : Likelihood to be in the foreground
 - ❑ b_v : Likelihood to be in the background

Simple Version of the Problem

- ❑ Input: values a_v and b_v
- ❑ Partition pixels into sets F and B so that

$$\sum_{v \in F} f_v + \sum_{v \in B} b_v \text{ is maximized}$$

Example

❑ What is the best value for the following three pixels?

Pixel v	1	2	3
f_v	3	5	6
b_v	4	3	5

Example

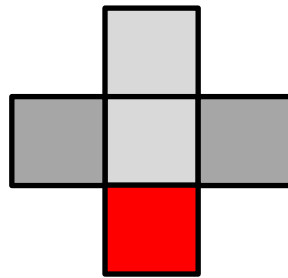
❑ Greedy assignment can solve this problem ...

❑ $4+5+6$ is the maximum value we can get

Pixel v	1	2	3
f_v	3	5	6
b_v	4	3	5

Nearby Pixels

- ❑ Expect nearby pixels will be on the same divide line



- ❑ Have penalty p_{vw} for each nearby pixel pair v and w :
 - ❑ v in foreground and w in background

Image Segmentation Problem

- Input: values a_v and b_v ; penalty p_{vw}
- Partition pixels into sets F and B so that

$$\sum_{v \in F} f_v + \sum_{v \in B} b_v - \sum_{v \in F, w \in B} p_{vw} \text{ is maximized}$$

Transformed Formula

- ❑ Input: values a_v and b_v ; penalty p_{vw}
- ❑ Partition pixels into sets F and B so that

$\sum_{v \in F} f_v + \sum_{v \in B} b_v - \sum_{v \in F, w \in B} p_{vw}$ is maximized
equivalent to

$-\sum_{v \in F} f_v - \sum_{v \in B} b_v + \sum_{v \in F, w \in B} p_{vw}$ is minimized

Transformed Formula

- ❑ Input: values a_v and b_v ; penalty p_{vw}
- ❑ Partition pixels into sets F and B so that

$\sum_{v \text{ in } F} f_v + \sum_{v \text{ in } B} b_v - \sum_{v \text{ in } F, w \text{ in } B} p_{vw}$ is maximized
equivalent to

$-\sum_{v \text{ in } F} f_v - \sum_{v \text{ in } B} b_v + \sum_{v \text{ in } F, w \text{ in } B} p_{vw}$ is minimized
equivalent to

$\sum_{v \text{ in } B} f_v + \sum_{v \text{ in } F} b_v + \sum_{v \text{ in } F, w \text{ in } B} p_{vw}$ is minimized

Pixel v	1	2	3
f_v	3	5	6
b_v	4	3	5

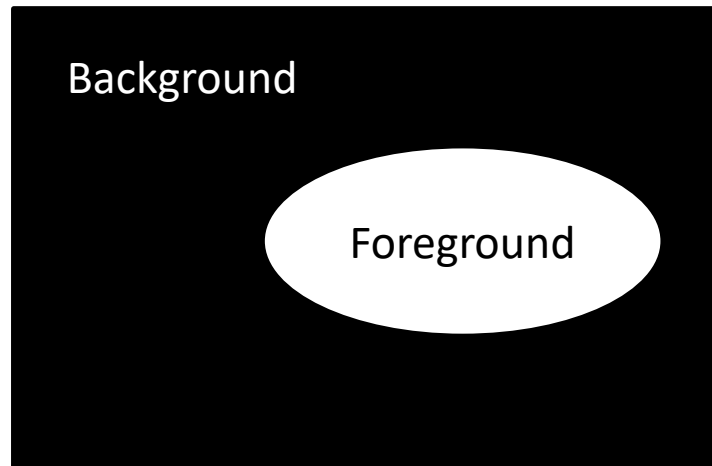
Why? Because $f_v + b_v$ is constant per pixel

Idea

☐ Observed

- ☐ Want to split pixels to two disjoint sets
- ☐ Pay cost based on boundary penalty

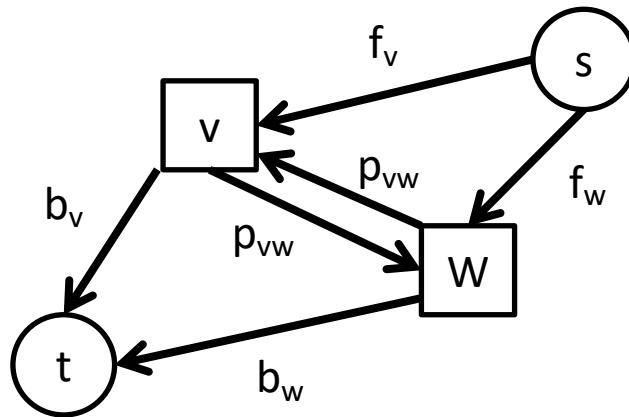
$\sum_{v \in B} f_v + \sum_{v \in F} b_v + \sum_{v \in F, w \in B} p_{vw}$ is minimized



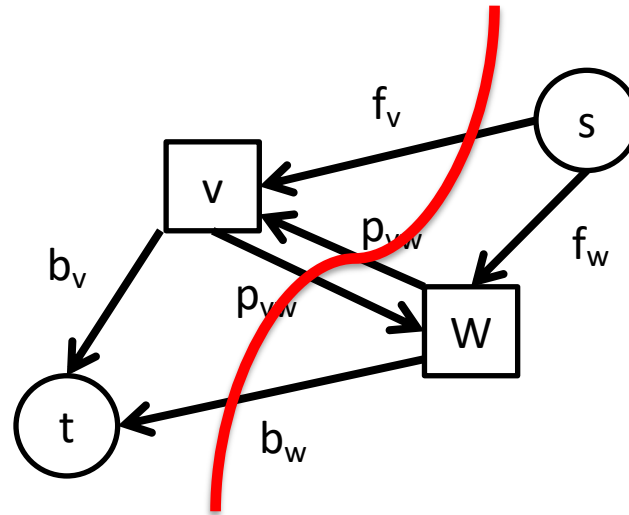
☐ Looks like a min-cut

Flow Network

- ❑ New vertices s and t
- ❑ Edge s to v with capacity f_v
- ❑ Edge v to t with capacity b_v
- ❑ Edge v to w with capacity p_{vw}
- ❑ Edge w to v with capacity p_{vw}



s-t Cut is the Best Segmentation



$$\sum_{v \in B} f_v + \sum_{v \in F} b_v + \sum_{v \in F, w \in B} p_{vw}$$

TODO Tasks

- ☐ Visit <https://vision.cs.uwaterloo.ca/data/maxflow>
- ☐ Download BVZ and KZ2 benchmarks (totally 6)
 - ☐ Each benchmark describes a flow network
 - ☐ Each benchmark provides a golden solution
- ☐ Implement the following maxflow problem
 - ☐ Ford-Fulkerson with DFS in the loop
 - ☐ Ford-Fulkerson with BFS in the loop
 - ☐ Push-Relabel algorithm
 - ☐ Boykov-Kolmogorov algorithm
 - “An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision”, IEEE Transaction on PAMI
 - <http://www.csd.uwo.ca/~yuri/Papers/pami04.pdf>

Fill the Table with Runtime Values

	Ford-Fulkerson DFS	Ford-Fulkerson BFS	Push-Relabel	Boykov- Kolmogorov
BVZ-tsukuba				
BVZ-sawtooth				
BVZ-venus				
KZ2-tsukuba				
KZ2-sawtooth				
KZ2-venus				

Benchmark Format

Comment Lines: Comment lines give human-readable information about the file and are ignored by programs. Comment lines can appear anywhere in the file. Each comment line begins with a lower-case character `c`.

```
c This is an example of a comment line.
```

Problem Line: There is one problem line per input file. The problem line must appear before any node or arc descriptor lines. For maximum flow network instances the problem line has the following format:

```
p max NODES ARCS
```

The lower-case character `p` signifies that this is a problem line. The three-character problem designator `max` identifies the file as containing specification information for a maximum flow problem. The `NODES` field contains an integer value specifying n , the number of nodes in the network. The `ARCS` field contains an integer value specifying m , the number of arcs in the network.

Benchmark Format Cont'd

Node Descriptors: All node descriptor lines must appear before all arc descriptor lines. Node descriptors are of the form:

n ID WHICH

where ID is the node id and WHICH is **s** for the source and **t** for the sink. Two node descriptors, one for the source and one for the sink, must appear between the problem line and the arc descriptor lines.

Arc Descriptors: There is one arc descriptor line for each arc in the network. For a maximum flow instance, arc descriptor lines are of the following form.

a SRC DST CAP

The lower-case character **a** signifies that this is an arc descriptor line. For a directed arc (v, w) the SRC field gives the identification number for the source vertex v , and the DST field gives the destination vertex w . Identification numbers are integers between 1 and n . The CAP field gives the arc capacity.

Submission Instruction

- ☐ Email the following to tsung-wei.huang@utah.edu
 - ☐ Compress your entire project to a tar ball or zip
 - ☐ A text or excel table summarizing runtime
 - 4 methods x 6 benchmarks
- ☐ **We expect you to be able to start from scratch**
 - ☐ Read and parse the input format (it's actually very simple)
 - ☐ Build your own flow network data structure
 - ☐ Study the very famous computer vision paper (Boykov-Kolmogorov maxflow algorithm)
- ☐ **Due 4/29 (final exam week)**