# Graph Theory Application in Computer Vision
# "Image Segmentation using Graph Cut"

**Presenter: Woohyun Eum**
Department of Electrical and Computer Engineering, University of Florida

UF Herbert Wertheim College of Engineering
*Department of Electrical & Computer Engineering*
UNIVERSITY *of* FLORIDA

# Agenda

- **Image as a Graph**
  - Graph-based representation
- **Graph Cuts**
  - Min-cut/Max-flow Algorithm → Nonlinear optimization problem
- **Ford-Fulkerson Algorithm**
  - User Interactive example

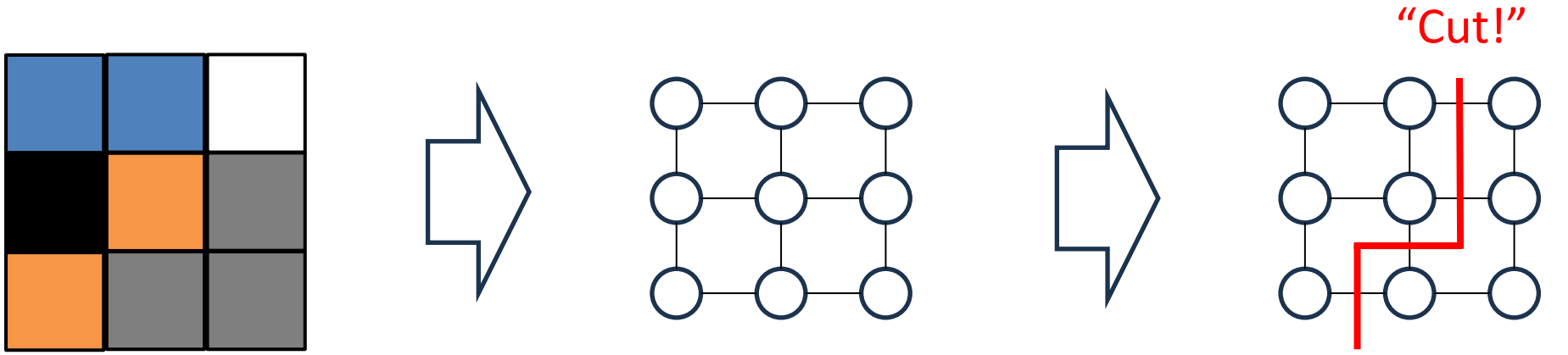
# Image as a Graph

- **Graph-based representation**
  - Two main components: Node & Edge
  - Formation: $G =< V, E > (V: node, E: Edge)$


Interactive Graph Cuts for Optimal Boundary

# Image as a Graph

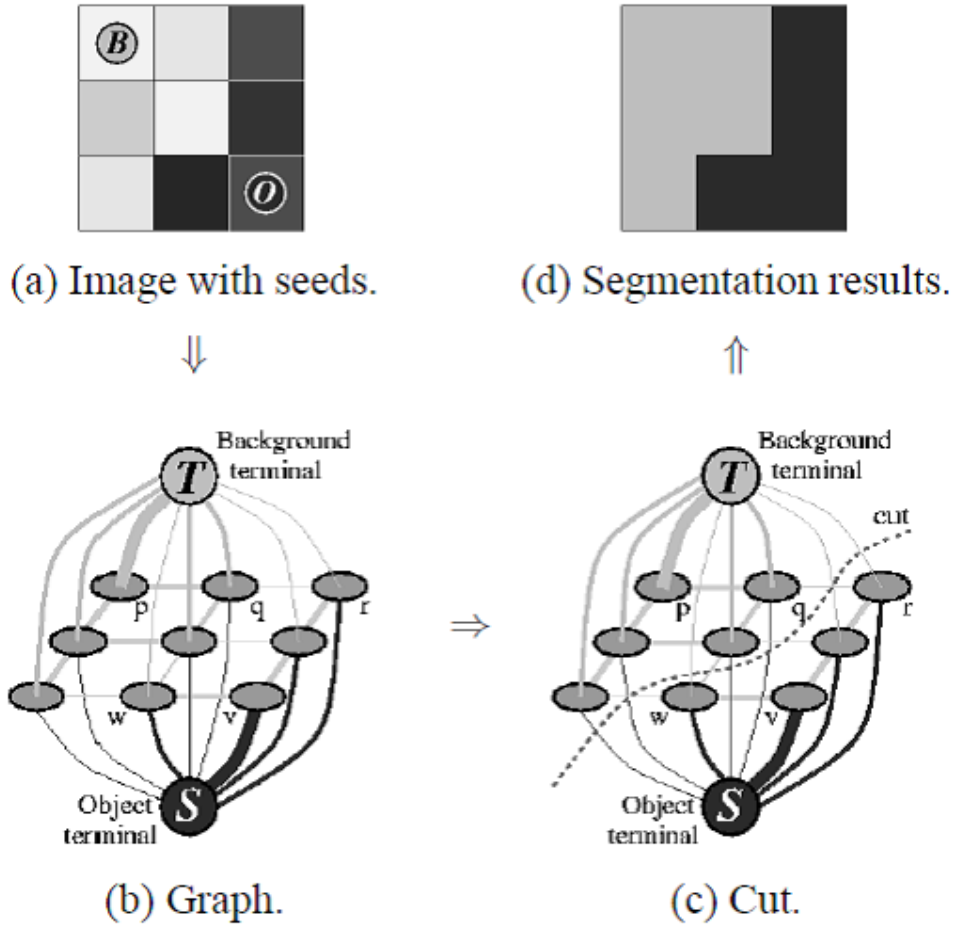
- **Graph-based representation**
  - Two main components: Node & Edge
  - Formation: $G =< V, E > (V: node, E: Edge)$




→ **Segmentation can be regarded as a "cut" in a graph**

Interactive Graph Cuts for Optimal Boundary

# Graph Cuts Algorithm




- **Min-cut/Max-flow Algorithm**
  - Seed is given by user by drawing
  - Drawn image provides terminal nodes
    - Source and Sink
    - → Cut the Graphs with a minimum cost

[Interactive Graph Cuts for Optimal Boundary]

# Graph Cuts Algorithm

- **Energy function on a graph**

$$\boldsymbol{E(L) = \sum_{p \in P} D_P(L_P) + \sum_{(p,q) \in N} V_{p,q}\,(L_P, L_q)}$$

$L$ : Label of each pixel $L = \{L_p | p \in P\}$

D: Data Penalty Function (How the intensity of pixel po is close to the model)

V: Smoothness function (Spatial distance, color similarity)

N: Set of neighbor pixels

- **There is no closed form solution → Optimization techniques are required: Min-cut/ Max-flow**

→ **Energy will be maximized at the ground truth**

Interactive Graph Cuts for Optimal Boundary

# Graph Cuts Algorithm

- **Non-linear optimization of energy function**
  - 1) Nodes can be formalized as $V = P \cup \{S, T\}$
    $P$: Pixel Set, $S$: Source Node (Object), $T$: Sink node (Background)
  - 2) Edges can be formalized as $\mathrm{E} = N \cup \{\{p, S\}, \{p, T\}\}$
  - 3) Data penalty function can be formalized as
    $\mathrm{D_p}(\text{"object"}) = -\ln Pr(I_P|O)$ and $\mathrm{D_p}(\text{"background"}) = -\ln Pr(I_P|B)$
  - Smoothness function can be formalized as

$$V_{p,q}(L_p, L_q) = \frac{\exp\left(-\frac{(I_p - I_q)^2}{2\sigma^2}\right)}{\|p - q\|}$$

$\mathrm{I_p} - \mathrm{I_q}$ : Difference in Pixel Intensity, $\|p - q\|$: Spatial distance

$\frac{1}{\|p-q\|}$: Considering closer pixels as the same set and further pixels as different set

Ford-Fulkerson Algorithm

# Graph Cuts Algorithm

- **Construct a graph and compute the corresponding weights**

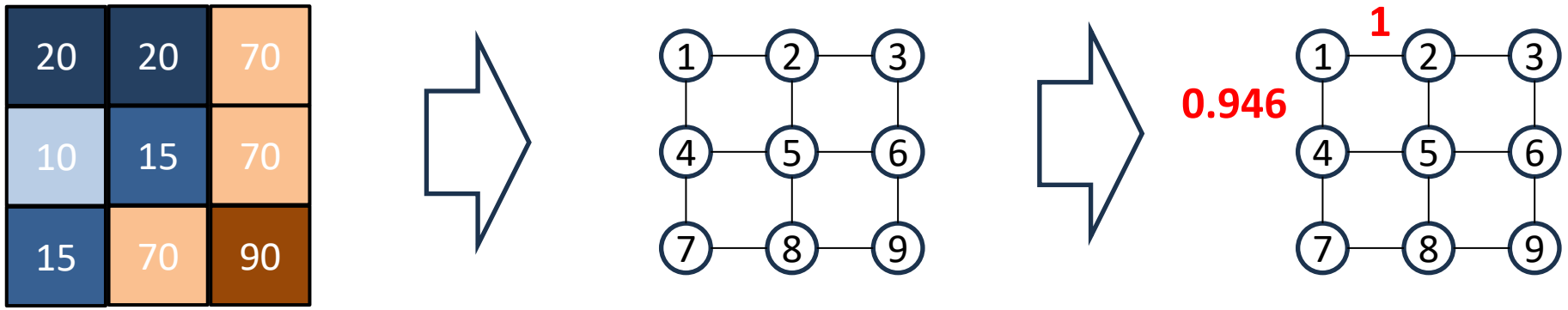



Image Graph Weighted Graph

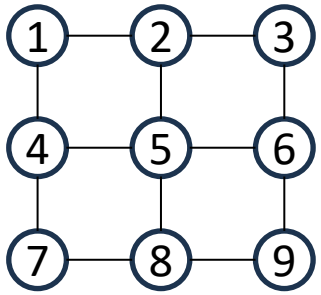
Let's set $\sigma$ =30, $\|p-q\|$=1

$$\frac{\exp\left(-\frac{(20-20)^2}{2*30^2}\right)}{1}=1$$

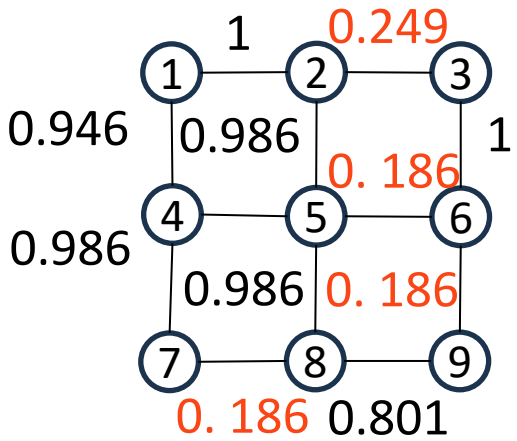
$$\frac{\exp\left(-\frac{(20-10)^2}{2*30^2}\right)}{1}=0.946$$

# Graph Cuts Algorithm

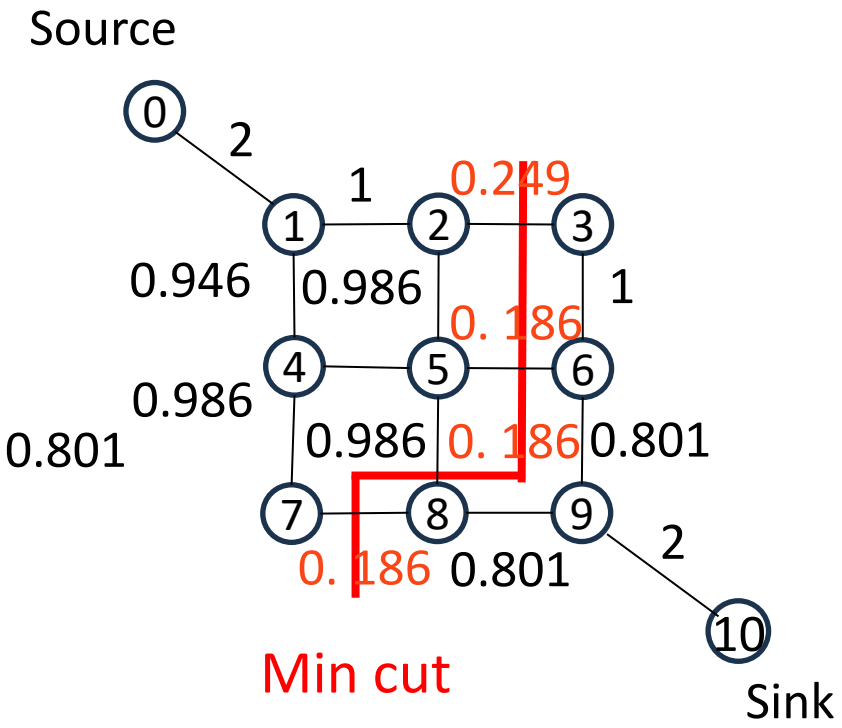
- **Construct a graph and compute the corresponding weights**




Graph




Weighted Graph




Min cut

# Ford-Fulkerson Algorithm

- How can we find the min-cut path automatically?

→ Ford-Fulkerson Algorithm

- Apply the concept of "augmenting path" based on the residual graph
- Code Application

Ford-Fulkerson Algorithm