Graph-based methods and segmentation

Filip Malmberg
This lecture

- Image processing using *graphs*.
- Image segmentation using *minimal graph cuts*
Part 1: Image Processing Using Graphs
What is an image?

“We will sometimes regard a *picture* as being a real-valued, non-negative function of two real variables; the value of this function at a point will be called the *gray-level* of the picture at the point.”

What is a digital image?

Storing the (continuous) image in a computer requires digitization, e.g.

- Sampling (recording image values at a finite set of *sampling points*).
- Quantization (discretizing the continuous function values).

Typically, sampling points are located on a Cartesian grid.
Generalized images

This basic model can be generalized in several ways: (cf. Lecture 1)

- Generalized image modalities (e.g., multispectral images)
- Generalized image domains (e.g., video, volume images)
- Generalized sampling point distributions (e.g., non-Cartesian grids)

The methods we develop in image analysis should (ideally) be able to handle this.
Why graph-based?

- Discrete and mathematically simple representation that lends itself well to the development of efficient and provably correct methods.
- A minimalistic image representation – flexibility in representing different types of images.
- A lot of work has been done on graph theory in other applications, We can re-use existing algorithms and theorems developed for other fields in image analysis!
A graph is a pair $G = (V, E)$, where
- $V$ is a set.
- $E$ consists of pairs of elements in $V$.

The elements of $V$ are called the *vertices* of $G$.
The elements of $E$ are called the *edges* of $G$. 
Graphs basic definition

- An edge spanning two vertices $v$ and $w$ is denoted $e_{v,w}$.
- If $e_{v,w} \in E$, we say that $v$ and $w$ are adjacent.
- The set of vertices adjacent to $v$ is denoted $\mathcal{N}(v)$. 
Figure 1: A drawing of an undirected graph with four vertices \{A, B, C, D\} and four edges \{e_{A,B}, e_{A,C}, e_{B,C}, e_{C,D}\}.
Example

Figure 2: The set $\mathcal{N}(A) = \{B, C\}$ of vertices adjacent to $A$. 
Images as graphs

- Graph based image processing methods typically operate on *pixel adjacency graphs*, i.e., graphs whose vertex set is the set of image elements, and whose edge set is given by an adjacency relation on the image elements.

- Commonly, the edge set is defined as all vertices \( v, w \) such that

  \[
  d(v, w) \leq \rho .
  \]  

- This is called the *Euclidean adjacency relation*. 
Pixel adjacency graphs, 2D

Figure 3: A 2D image with $4 \times 4$ pixels.

Figure 4: A 4-connected pixel adjacency graph.

Figure 5: A 8-connected pixel adjacency graph.
Pixel adjacency graphs, 3D

Figure 6: A volume image with $3 \times 3 \times 3$ voxels.

Figure 7: A 6-connected voxel adjacency graph.

Figure 8: A 26-connected voxel adjacency graph.
Foveal sampling

“Space-variant sampling of visual input is ubiquitous in the higher vertebrate brain, because a large input space may be processed with high peak precision without requiring an unacceptably large brain mass.” [6]

Figure 9: Ducks. (Image from Grady 2004)
Foveal sampling

Figure 10: Left: Retinal topography of a Kangaroo. Right: Re-sampled duck image. (Images from Grady 2004)
Region adjacency graphs

Figure 11: An image divided into superpixels
Directed and undirected graphs

- The pairs of vertices in $E$ may be ordered or unordered.
  - In the former case, we say that $G$ is directed.
  - In the latter case, we say that $G$ is undirected.
- In this lecture, we will mainly consider undirected graphs.
Paths

A path is an ordered sequence of vertices where each vertex is adjacent to the previous one.

Figure 12: A path $\pi = \langle A, D, E, H, I, F, E \rangle$.
Example, Simple path

A path is *simple* if it has no repeated vertices. Often, simplicity of paths is implied, i.e., the word “simple” is omitted.

![Diagram of a simple path](image)

**Figure 13:** A simple path $\pi = \langle G, H, E, B, C \rangle$. 
Paths and connectedness

- Two vertices $v$ and $w$ are *linked* if there exists a path that starts at $v$ and ends at $w$. We use the notation $v \sim_w G$. We can also say that $w$ is *reachable* from $v$.

- If all vertices in a graph are linked, then the graph is *connected*. 
Subgraphs and connected components

- If $G$ and $H$ are graphs such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$, then $H$ is a subgraph of $G$.
- If $H$ is a connected subgraph of $G$ and
  - $v \not\sim_G w$ for all vertices $v \in H$ and $w \notin H$,
  - (for any pair of vertices $v, w \in H$ it holds that $e_{v,w} \in E(H)$ iff $e_{v,w} \in E(G)$),
then $H$ is a connected component of $G$. 
Example, connected components

Figure 14: A graph with three connected components.
Graph segmentation

- To segment an image represented as a graph, we want to partition the graph into a number of separate connected components.
- The partitioning can be described either as a *vertex labeling* or as a *graph cut*.
Vertex labeling

We associate each vertex with an element in some set $L$ of labels, e.g., $L = \{\text{object}, \text{background}\}$.

**Definition, vertex labeling**

A (vertex) labeling $\mathcal{L}$ of $G$ is a map $\mathcal{L} : V \rightarrow L$. 
Graph cuts

Informally, a (graph) cut is a set of edges that, if they are removed from the graph, separate the graph into two or more connected components.

Definition, Graph cuts

Let $S \subseteq E$, and $G' = (V, E \setminus S)$. If, for all $e_{v,w} \in S$, it holds that $v \not\sim_{G'} w$, then $S$ is a (graph) cut on $G$. 
Example, cuts

Figure 15: A set of edge (red) that do not form a cut.
Example, cuts

Figure 16: A set of edge (red) that do not form a cut.
Example, cuts

Figure 17: A set of edge (red) that form a cut.
Relation between labelings and cuts

**Definition, labeling boundary**

The boundary $\partial L$, of a vertex labeling is the edge set

$$\partial L = \{ e_{v,w} \in E \mid L(v) \neq L(w) \}.$$ 

**Theorem**

For any graph $G = (V, E)$ and set of edges $S \subseteq E$, the following statements are equivalent*: [7]

1. There exists a vertex labeling $L$ of $G$ such that $S = \partial L$.
2. $S$ is a cut on $G$.

*) Provided that $|L|$ is “large enough”.

Centre for Image Analysis
Swedish University of Agricultural Sciences
Uppsala University
Relation between labelings and cuts

Figure 18: Duality between cuts and labelings.
Summary, Part 1

- Basic graph theory
  - Directed and undirected graphs
  - Paths and connectedness
  - Subgraphs and connected components
- Images as graphs
  - Pixel adjacency graphs in 2D and 3D
  - Alternative graph constructions
- Graph partitioning
  - Vertex labeling and graph cuts
Part 2: Image Segmentation Using Minimal Graph Cuts
Image Segmentation with Minimal graph cuts

- In the previous segmentation Lecture, we talked about *active contours* (snakes and level sets) for image segmentation. These methods find boundaries of object in an image by
  - Starting with an initial guess.
  - Iteratively modifying the initial guess until the contour is *locally optimal* according to some energy functional.
- In this lecture, we will introduce *minimal graph cuts*. This method finds discrete surfaces (cuts) that are *globally optimal* according to a specific energy functional.
Global vs local optimization

- We can phrase image segmentation as an optimization problem by defining an objective function that measures the goodness of every possible segmentation.
- A segmentation is locally optimal if it is better than or equal to all nearby solutions.
- A segmentation is globally optimal if it is better than or equal to all other solutions.
- For an arbitrary objective function, finding a globally optimal segmentation requires checking all possible solutions. For restricted classes of objective functions however, it is sometimes possible to design efficient algorithms that are guaranteed to find global optima. Minimal graph cuts is one such method.
Edge weighted graphs

- We associate each edge $e \in E$ with a real valued, non-negative weight, $w(e)$.
- The weight of an edge represents the similarity (or, alternatively, dissimilarity) between the vertices connected by the edge.
- For example, we may define the edge weights as

$$w(e_{ij}) = 1 - |I(v) - I(j)|,$$

where $I(v) \in [0, 1]$ is the intensity of the image element corresponding to $v$. 

Minimal graph cuts

- Consider a graph $G = (V, E)$, where we have selected two vertices $s, t \in V$.
- A cut on $G$ is an $s - t$ cut if it separates $s$ from $t$.
- In the minimal graph cut problem, we want to find an $s - t$ cut $C$ that globally minimizes
  \[ \sum_{e \in C} w(e) . \]  

- In 1956, it was shown by Ford and Fulkerson, that the problem of finding a minimal $s$-$t$ cut is equivalent to the problem of finding a maximum $s$-$t$ flow.
Minimal $s - t$ cuts

Figure 19: $s - t$ graph cut
Flow network, intuitive notion

Figure 20: Flow network
Networks

**Definition, network**

A *network* is a directed graph $G = (V, E)$ where

1. Two vertices $s, t \in V$ are defined as the *source* and *sink* of $G$, respectively. The source has only outgoing edges and the sink has only incoming edges.

2. A *capacity* function, $c : E \to \mathbb{R}^+$ maps each edge to how much “traffic” it can carry. (In our case, capacity $=$ weight)
Flow

Definition, flow

Given a network $G = (V, E)$, a *(s-t)* flow is a mapping $f : E \rightarrow \mathbb{R}^+$, such that:

1. $f(p, q) \leq c(p, q)$ for all $e_{p,q} \in E$. (Capacity constraint)
2. $\sum_{q \in N(p)} f(p, q) = \sum_{q \in N(p)} f(q, p)$ for all $p \in V \setminus \{s, t\}$. (Flow conservation)
Maximum flow

- The value, $|f|$, of a flow is the total amount of flow being sent from the sink to the source, i.e.,

$$|f| = \sum_{p \in \mathcal{N}(s)} f(s, p). \quad (4)$$

- The maximum flow problem is to maximize $|f|$, i.e., to route as much flow as possible from $s$ to $t$. 
Ford-Fulkerson theorem

Theorem

The maximum value of a $s - t$ flow on $G$ is equal to the minimum capacity of an $s - t$ cut.

Moreover, a maximum flow on $G$ will saturate a set of edges that gives us the minimum cut.
Computing minimum cuts/maximal flows

- According to the Ford-Fulkerson theorem, we can compute minimum s-t cuts by computing maximum flow.
- We will now look at one algorithm for computing maximum flows: the Ford-Fulkerson algorithm [5].
Ford-Fulkerson algorithm

**Definition, augmenting path**

Let $G = (V, E)$ be a network and let $f$ be a flow on $G$. A path $\pi$ in $G$ is called an *augmenting path* if

1. The origin of $\pi$ is $s$ and the destination of $\pi$ is $t$.
2. $f(p, q) < c(p, q)$ for all edges $e_{p,q}$ along the path.
Ford-Fulkerson algorithm

while There exists an augmenting path in $G$ do
  Send flow along that path
end
Ford-Fulkerson algorithm, example

Figure 21: A network with zero flow.
Ford-Fulkerson algorithm, example

Figure 22: Find an augmenting path.
Ford-Fulkerson algorithm, example

Figure 23: Send flow along the path.
Ford-Fulkerson algorithm, example

Figure 24: Find an augmenting path.
Ford-Fulkerson algorithm, example

Figure 25: Send flow along the path.
Ford-Fulkerson algorithm, example

Figure 26: Find an augmenting path.
Figure 27: Send flow along the path.
Figure 28: No more augmenting paths can be found. Label all vertices that can be reached via non-saturated edges as “belonging to the source”.

Ford-Fulkerson algorithm, example
Ford-Fulkerson algorithm, example

Figure 29: Label all remaining vertices as “belonging to the sink”.
Ford-Fulkerson algorithm, example

Figure 30: The edges on the boundary of this labeling form a minimum s-t cut.
At each step of the Ford-Fulkerson algorithm, we select an augmenting path.

The running time of the algorithm depends on the order in which we select the augmenting paths, and on the search strategy we use to find a path.
Edmonds-Karp algorithm

- The Edmonds-Karp algorithm is a specialization of the Ford-Fulkerson algorithm. [3]
- At each step, the algorithm selects a shortest augmenting path (where the length of a path is the number of vertices in the path).
- The algorithm can be shown to run in \( O(|V||E|^2) \).
Boykov-Kolmogorov algorithm

- Another specialization of the Ford-Fulkerson algorithm, tuned for the types of graphs commonly occurring in image processing [1].
- The basic idea of the algorithm is to maintain two search trees, one from the source and one from the sink. These trees are updated during the execution of the algorithm, so we do not need to perform the search for an augmenting path from scratch.
- The theoretical running time is worse than for the Edmonds-Karp algorithm, but it has been shown to be faster in many practical scenarios.
- An implementation in C is available: http://pub.ist.ac.at/~vnk/software.html
Minimal graph cuts and image labeling: Hard constraints

Minimal graph cuts can be used to implement the following segmentation procedure:

- Assume that we know the correct segmentation labels (object or background) for a small subset of the pixels, called \textit{seedpoints}, in an image. The seedpoints may be provided manually by a user, or found by an automatic algorithm.
- Complete the labeling for all pixels by computing the minimum cut that separates the object seeds from the background seeds.
Interactive seeded segmentation

Figure 31: Segmentation of 2D images using minimal graph cuts with hard constraints.

(a) A woman from a village  
(b) A church in Mozhaisk (near Moscow)
Interactive seeded segmentation

Figure 32: Segmentation of bones in a CT volume using minimal graph cuts with hard constraints.
Minimal graph cuts and image labeling: Hard constraints

Figure 33: A pixel adjacency graph with “extra” terminal vertices $s$ and $t$, corresponding to object and background respectively. Each seedpoint is connected to one of the terminal vertices with an edge of infinite capacity.
Minimal graph cuts and image labeling: Soft constraints

As an alternative to using hard constraints, we can specify a real valued “preference” for each pixel to belong to the foreground and background, respectively. Specifically, we can compute a binary labeling $L$ of a graph (image) that minimizes cost functions of the form

$$E(L) = R(L) + B(L)$$  \hspace{1cm} (5)

where

$$R(L) = \sum_{v \in V} R_v(L_v)$$  \hspace{1cm} (6)

is called the \textit{regional term} (or \textit{data term}) and

$$B(L) = \sum_{v,w \in \partial L} B_{v,w}$$  \hspace{1cm} (7)

is called the \textit{boundary term} (or \textit{regularization term}).
Figure 34: A noisy image. It seems possible to segment this image by saying that "object pixels are dark and background pixels are bright".
Data term

We define the data term $R(L)$ as

$$R = \sum_{v \in V} \Phi(v),$$

(8)

where

$$\Phi(v) = \begin{cases} 
  \text{abs}(\max(t - I(v), 0)) & \text{if } L(v) = \text{foreground} \\
  \text{abs}(\max(I(v) - t, 0)) & \text{otherwise}
\end{cases}.$$

(9)
Figure 35: Data term in the example. The red curve is the cost of assigning the label “background” to a vertex with a certain intensity, and the green curve is the cost of assigning the “foreground” label.
Figure 36: Segmentation result with data term only.
Adding a regularization term

- We can add a boundary term $B(L)$, that penalizes long boundaries:

$$B(L) = \alpha |\partial L|,$$

(10)

where $\alpha$ is a real number that controls the degree of "smoothing".
Adding a regularization term

Figure 37: Segmentation result after adding regularization term.
Minimal graph cuts and image labeling: Soft constraints

Figure 38: Minimum graph cut segmentation of a 3x3 image. Image from [1].
How to construct the graph for the data term?

- For every pixel $v$, add two edges: one that connects $v$ to the sink $t$ and one that connects $v$ to the source $s$.
- If the pixel is labeled as \textit{object}, then the edge from $v$ to $t$ will be cut. If the pixel is labeled as \textit{background}, then the edge from $v$ to $s$ will be cut.
- Therefore, the weight of the edge from $v$ to $t$ should equal the cost $R_v(\text{object})$ of assigning the pixel to the object, and the weight of the edge from $v$ to $s$ should equal the cost $R_v(\text{background})$ of assigning the pixel to the background.
Metrification errors

- We can think of the cost of a cut as the area of a surface separating the two regions.
- On a regular 2D or 3D grid, we will see metrification errors—signs of the discrete nature of the graph representation.
Comparison, discrete distance transforms

Figure 39: Distances in discrete grids [2]. The weight of each edge is equal to its Euclidean length.
Reducing metrication errors

- Just like in the distance transform example, we can reduce metrication errors by using a “larger” neighborhood system.
- In fact, it is possible to construct a graph such that the weight of the cut is arbitrarily close to the length (area) of the corresponding contours (surfaces) for any Riemannian metric [2].
Reducing metrication errors

(a) Original data
(b) 4 n-system
(c) 8 n-system
image restoration experiments on 2D data

(d) Original data
(e) 6 n-system
(f) 26 n-system
object extraction experiments on 3D data
Shrinking bias

- Since we are minimizing the sum of the edge weights in the cut, we implicitly favour “small” cuts. This may or may not be what we want.
- To avoid this issue, some authors [8, 4] have considered “normalized cuts” that minimize

\[ \sum_{e \in C} \frac{w(e)}{|C|}. \]  (11)
Comparison: Minimal graph cuts vs. Active Contour models

Active contours (level sets, snakes)
- Continuous representation.
- Local minimization.

Minimal graph cuts
- Discrete representation.
- Global optimization.
The ability to optimize cost functions of the form discussed here has applications to many image processing tasks other than segmentation. Examples:

- Filtering (Labels are intensities).
- Stereo matching (Labels are disparities or depths).
Image restoration

Figure 41: “Restoration” of noisy image.
Stereo disparity

Figure 42: Left: One image in a stereo pair. Right: Depth estimated with graph cuts.
More than two terminals?

- The examples on the previous slides have more than two types of labels.
- So far, we have only considered minimal graph cuts on graphs with two terminals. Thus, we can only compute binary labelings.
- Unfortunately, computing globally minimal graph cuts for more than two terminals is NP-hard.
- There are techniques for computing multi-label graph cuts that are \textit{locally minimal} in a strong sense. This is beyond the scope of this lecture.
Comparison, minimal graph cuts and active contours

Active contours

- **Advantages**
  - Flexible, can be used with a wide range of energy functionals.
  - Multiple objects can be segmented simultaneously.
  - No metrication artifacts.

- **Drawbacks**
  - May get trapped in poor local minima.
  - Requires proper initialization.
Comparison, minimal graph cuts and active contours

Minimal graph cuts

- **Advantages**
  - Guaranteed to find globally optimal solutions.
  - No initialization required.
  - Easy to incorporate both soft and hard constraints.
  - Can approximate continuous cut metrics with arbitrary precision.

- **Drawbacks**
  - Restricted to binary segmentation.
  - Restricted to a special class of energy functionals.
  - Metrication artifacts on standard grids.
References

Y. Boykov and V. Kolmogorov.
An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision.

Yuri Boykov.
Computing geodesics and minimal surfaces via graph cuts.

J. Edmonds and R. Karp.
Theoretical improvements in algorithmic efficiency for network flow problems.

A.P. Eriksson, C. Olsson, and F. Kahl.
Normalized cuts revisited: A reformulation for segmentation with linear grouping constraints.

L. Ford and D. Fulkerson.
Flows in networks.

L. Grady.
*Space-Variant Machine Vision — A Graph Theoretic Approach.*

F. Malmberg, J. Lindblad, N. Sladoje, and I. Nyström.
A graph-based framework for sub-pixel image segmentation.

Jianbo Shi and J. Malik.
Normalized cuts and image segmentation.