SSBA Summerschool 2015: Image Processing using Graphs

Uppsala August 18-20 2015

Filip Malmberg

About the course

- Course webpage: `http://www.cb.uu.se/~filip/ImageProcessingUsingGraphs/`
- Format:
  - Lectures.
  - Review questions/exercises
  - Computer exercises
  - Examination in the form of an individual project.

Course goals

After completing the course, you should:

- be familiar with basic graph theory and how it applies to image processing application.
- have a good understanding of combinatorial optimization.
- have a good understanding of state-of-the-art methods for solving combinatorial optimization problems arising in image processing.
- have some experience implementing and using such methods, and applying them in your own research.

Teachers

- Filip Malmberg
- Johan Nysjö (computer exercises)

Examination: Individual project

To recieve credits for the course, you should complete an individual project:

- Each participant should also select a topic for his/her individual project. The project can be applied or theoretical.
- When you have decided on a topic, discuss this with me (Filip) to ensure that the scope is appropriate.
- Your work should be presented as a written report ($\sim$ 4 pages).
- Submit your report to me (Filip).
- The final reports will be published on the course webpage.

Background

- In recent years, graphs have emerged as a unified representation for image analysis and processing. In this course, we will give an overview of recent developments in this field.
- How and why do we represent images as graphs?
- Graph-based methods for:
    - Image segmentation
    - Image restoration/filtering
    - Image registration / stereo matching
    - Scattered data interpolation

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."

Rosenfeld, *Picture Processing by Computer*, ACM Computing Surveys, 1969.

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:

- 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!

Euler and the seven bridges of Konigsberg

Wikipedia: "The city of Königsberg in Prussia (now Kaliningrad, Russia) was set on both sides of the Pregel River, and included two large islands which were connected to each other and the mainland by seven bridges. The problem was to find a walk through the city that would cross each bridge once and only once."

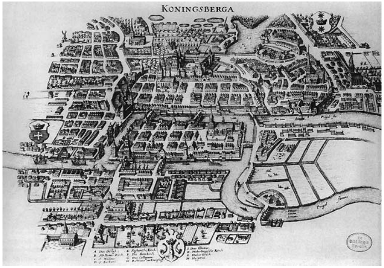

Figure 1: Königsberg in 1652.

Euler and the seven bridges of Konigsberg

In 1735, Euler published the paper “Solutio problematis ad geometriam situs pertinentis“ (“The solution of a problem relating to the geometry of position”), showing that the problem had no solution. This is regarded as the first paper in graph theory.

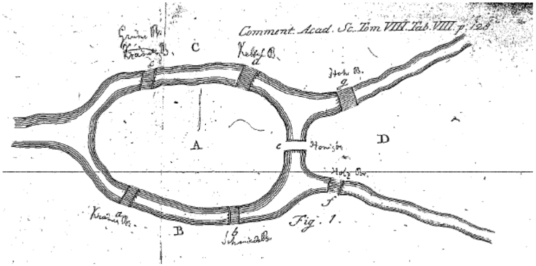

Figure 2: The seven bridges of Konigsberg, as drawn by Leonhard Euler.

Graphs, basic definition

- 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 $\mathcal{N}(v)$.

Example

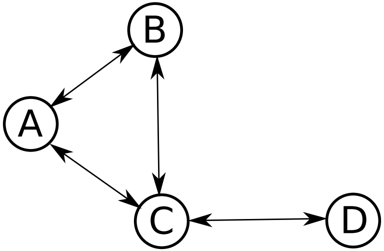

Figure 3: A drawing of an undirected graph with four vertices $\{A, B, C, D\}$ and four edges $\{e_{A,B},\ e_{A,C},\ e_{B,C},\ e_{C,D}\}$.

Example

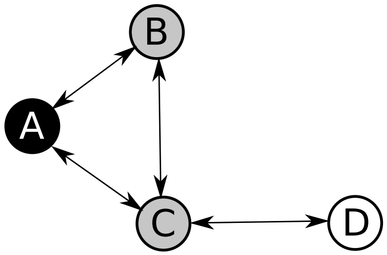

Figure 4: The set $\mathcal{N}(A) = \{B, C\}$ of vertices adjacent to A.

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

$$d(v, w) \leq \rho \ . \tag{1}$$

- This is called the *Euclidean adjacency relation*.

Pixel adjacency graphs, 2D

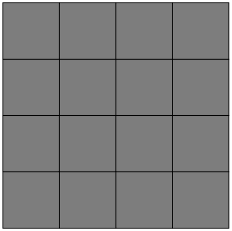
Figure 5: A 2D image with 4 $\times$ 4 pixels.

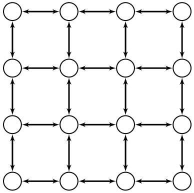
Figure 6: A 4-connected pixel adjacency graph.

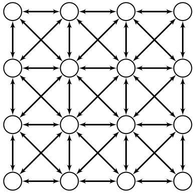
Figure 7: A 8-connected pixel adjacency graph.

Pixel adjacency graphs, 3D

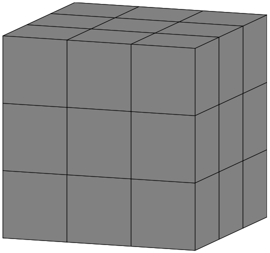
Figure 8: A volume image with $3 \times 3 \times 3$ voxels.

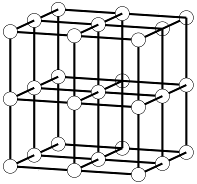
Figure 9: A 6-connected voxel adjacency graph.

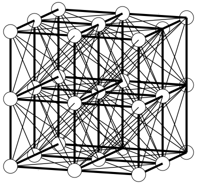
Figure 10: A 26-connected voxel adjacency graph.

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." [1]

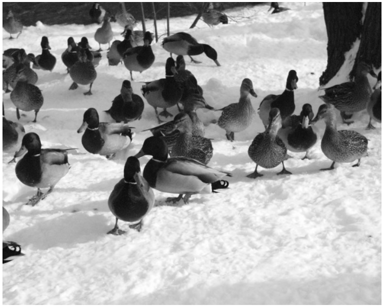

Figure 11: Some ducks. (Image from Grady 2004)

Foveal sampling

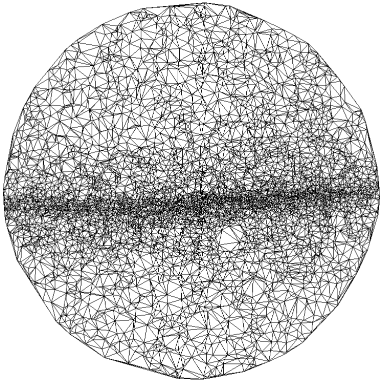

Figure 12: Left: Retinal topography of a Kangaroo. Right: Re-sampled image. (Images from Grady 2004)

Region adjacency graphs

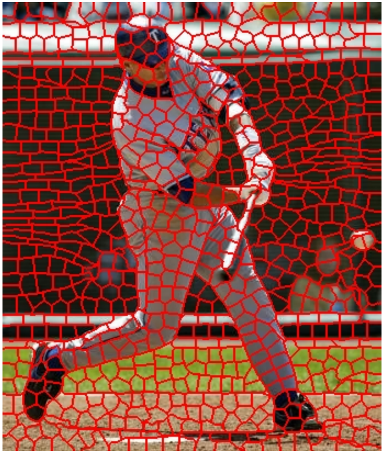
Figure 13: An image divided into superpixels

Multi-scale image representation

Resolution pyramids can be used to perform image analysis on multiple scales. Rather than treating the layers of this pyramid independently, we can represent the entire pyramid as a graph.

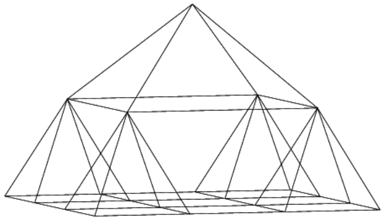

Figure 14: A pyramid graph (Grady 2004).

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 course, we will mainly consider undirected graphs.

Paths

- A *path* is an ordered sequence of vertices where each vertex is adjacent to the previous one.
- A path is *simple* if it has no repeated vertices.
- A *cycle* is a path where the start vertex is the same as the end vertex.
- A cycle is *simple* if it has no repeated vertices other than the endpoints.

Commonly, simplicity of paths and cycles is implied, i.e., the word "simple" is ommited.

Example, Path

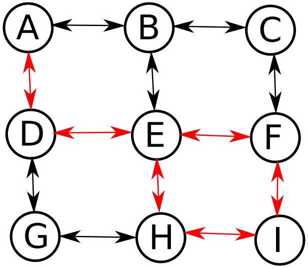

Figure 15: A path $\pi = \langle A, D, E, H, I, F, E \rangle$.

Example, Simple path

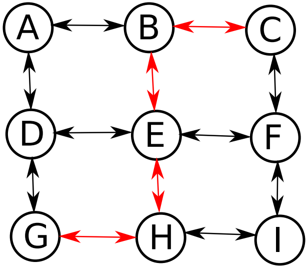

Figure 16: A simple path $\pi = \langle G, H, E, B, C \rangle$.

Example, Cycle

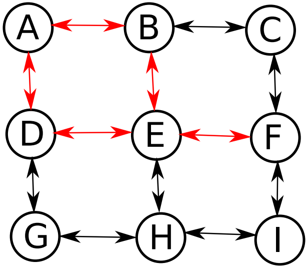

Figure 17: A cycle $\pi = \langle A, B, E, F, E, D, A \rangle$.

Example, Simple cycle

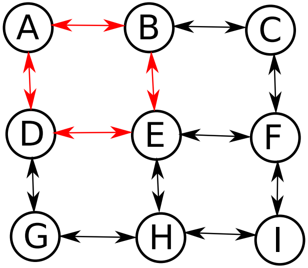

Figure 18: A simple cycle $\pi = \langle A, D, E, B, A \rangle$.

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 $v \underset{G}{\sim} w$. 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.
- A subgraph H of G is said to be *induced* if, for any pair of vertices $v, w \in H$ it holds that $e_{v,w} \in E(H)$ iff $e_{v,W} \in E(G)$.
- If H is a (induced) connected subgraph of G and $v \underset{G}{\not\sim} w$ for all vertices $v \in H$ and $w \notin H$, then H is a *connected component* of G.

Example, connected components

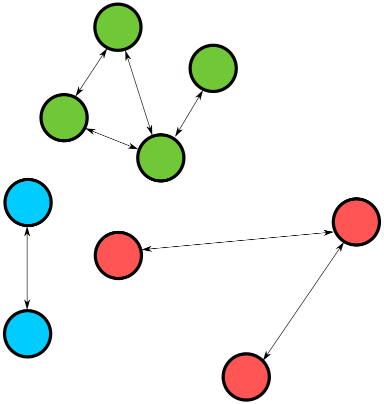

Figure 19: A graph with three connected components.

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., $L = \{object, background\}$.

Definition, vertex labeling

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

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 $v \underset{G'}{\not\sim} w$, then S is a *(graph) cut* on G.

Example, cuts

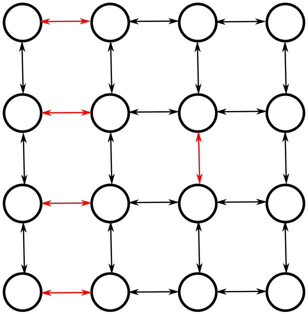

Figure 20: A set of edge (red) that do *not* form a cut.

Example, cuts

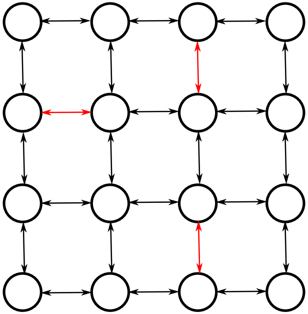

Figure 21: A set of edge (red) that do *not* form a cut.

Example, cuts

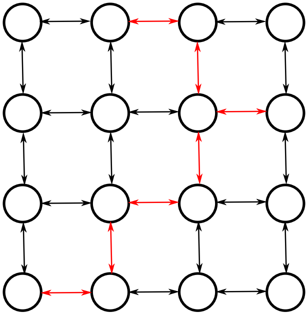

Figure 22: A set of edge (red) that form a cut.

Relation between labelings and cuts

Definition, labeling boundary

The boundary $\partial\mathcal{L}$, of a vertex labeling is the edge set $\partial\mathcal{L} = \{e_{v,w} \in E \mid \mathcal{L}(v) \neq \mathcal{L}(w)\}$.

Theorem

For any graph $G = (V, E)$ and set of edges $S \subseteq E$, the following statements are equivalent: [2]*

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

*) Provided that $|L|$ is "large enough".

Relation between labelings and cuts

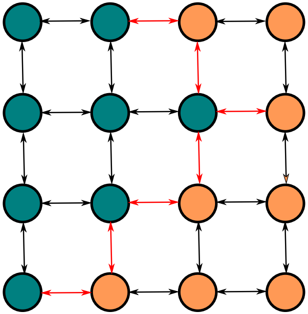

Figure 23: Duality betwen cuts and labelings.

Implementing graph-based algorithms

- Even if we have formulated an algorithm on a general graphs, we do not neccesarily have to allow arbitrary graphs in *implementations* of the algorithm.
- For standard pixel/voxel adjacency graphs, we can evaluate adjacency relations without having to store the graph explicitly.

Implementing graph-based algorithms

If we do want to store the graph explicitly, there are some available libraries:

- For C++, I recommend the *Boost Graph* libraries. (www.boost.org)
- For Matlab, check out the *Graph Analysis Toolbox* (http://cns.bu.edu/ lgrady/software.html).

Summary

- 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

Next lecture

- Intro to combinatorial optimization
- Applications in image processing

References

L. Grady.
Space-Variant Machine Vision — A Graph Theoretic Approach.
PhD thesis, Boston University, 2004.

F. Malmberg, J. Lindblad, N. Sladoje, and I. Nyström.
A graph-based framework for sub-pixel image segmentation.
Theoretical Computer Science, 2010.
doi: 10.1016/j.tcs.2010.11.030.