Optimization

Computer Vision, Lecture 13
Michael Felsberg
Computer Vision Laboratory
Department of Electrical Engineering

LINKÖPINGS UNIVERSITET

Why Optimization?

- Computer vision algorithms are usually very complex
  - Many parameters (dependent)
  - Data dependencies (non-linear)
  - Outliers and occlusions (noise)
- Classical approach
  - Trial and error (hackers' approach)
  - Encyclopedic knowledge (recipes)
  - Black-boxes + glue (hide problems)

Why Optimization?

- Establishing CV as scientific discipline
  - Derive algorithms from first principles (*optimal solution*)
  - Automatic choice of parameters (*parameter free*)
  - Systematic evaluation (*benchmarks on standard datasets*)

Optimization: howto

1. Choose a *scalar* measure (objective function) of success
   - From the benchmark
   - Such that optimization becomes *feasible*
   - Project functionality onto *one dimension*
2. Approximate the world with a model
   - Definition: allows to make *predictions*
   - Purpose: makes optimization *feasible*
   - Enables: *proper* choice of dataset

Similar to economics (money rules)

Optimization: howto

3. Apply suitable framework for model fitting
   - This lecture
   - Systematic part (1 & 2 are ad hoc)
   - Current focus of research
4. Analyze resulting algorithm
   - Find *appropriate* dataset
   - Ignore runtime behavior (*highly non-optimized Matlab code*) ;-)

Examples

- Relative pose (F-matrix) estimation:
  - Algebraic error (quadratic form)
  - Linear solution by SVD
  - Robustness by random sampling (RANSAC)
  - Result: F and inlier set
- Bundle adjustment
  - Geometric (reprojection) error (quadratic error)
  - Iterative solution using LM
  - Result: camera pose and 3D points

Taxonomy

- Objective function
  - Domain/manifold (algebraic error, geometric error, data dependent)
  - Robustness (explicitly in error norm, implicitly by Monte-Carlo approach)
- Model / simplification
  - Linearity (limited order), Markov property, regularization
- Algorithm
  - Approximate / analytic solutions (minimal problem)
  - Minimal solutions (over-determined)

Taxonomy example: KLT

- Objective function
  - Domain/manifold: grey values / RGB / …
  - Robustness: no (quadratic error, no regularization)

$$\varepsilon(\mathbf{d}) = \sum_{\mathbf{x}\in\mathcal{N}} w(\mathbf{x})|f(\mathbf{x}-\mathbf{d}) - g(\mathbf{x})|^2$$

- Model: Brightness constancy, image shift

$$f(\mathbf{x}-\mathbf{d}) = g(\mathbf{x}) \quad \forall \mathbf{x}\in\mathcal{N}$$

  - local linearization (Taylor expansion)

$$f(\mathbf{x}-\mathbf{d}) \approx f(\mathbf{x}) - \mathbf{d}^T\nabla f(\mathbf{x})$$

$$\nabla f = \left[\frac{\partial f}{\partial x}\ \frac{\partial f}{\partial y}\right]^T$$

Taxonomy: KLT

- Algorithm
  - iterative solution of normal equations (Gauss-Newton)

$$\left(\sum_{\mathcal{R}} w(\mathbf{x})\nabla f(\mathbf{x})\nabla^T f(\mathbf{x})\right)\mathbf{d} = \sum_{\mathcal{R}} w(\mathbf{x})\nabla f(\mathbf{x})(f(\mathbf{x}) - g(\mathbf{x}))$$

$$\mathbf{Td} = \mathbf{r}$$

  - $\mathbf{T}$: structure tensor (orientation tensor from outer product of gradients)

$$\nabla f \nabla^T f = \begin{bmatrix} \left(\frac{\partial f}{\partial x}\right)^2 & \frac{\partial f}{\partial x}\frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial x}\frac{\partial f}{\partial y} & \left(\frac{\partial f}{\partial y}\right)^2 \end{bmatrix}$$

- Block matching: same cost & model, but discretized shifts

Regularization and MAP

- In maximum a-posteriori (MAP), the objective (or loss) ε consists of a data term (fidelity) and a prior

$$\min_{\mathbf{d}} \varepsilon_{\text{data}}(f(\mathbf{d}), g) + \varepsilon_{\text{prior}}(\mathbf{d})$$

$$\Leftrightarrow \max_{\mathbf{d}} \exp(-\varepsilon_{\text{data}}(f(\mathbf{d}), g))\exp(-\varepsilon_{\text{prior}}(\mathbf{d}))$$

$$\Leftrightarrow \max_{\mathbf{d}} P(g|\mathbf{d})P(\mathbf{d})$$

$$\Leftrightarrow \max_{\mathbf{d}} P(\mathbf{d}|g)$$

- A common prior is a smoothness term (regularizer)

MAP Example: KLT

- Assume a prior probability for the displacement: $P(\mathbf{d})$ (e.g. Gaussian distribution from a motion model)
- In logarithmic domain, we now have two terms in the cost function:

$$\varepsilon(\mathbf{d}) = \sum_{\mathbf{x} \in \mathcal{N}} w(\mathbf{x}) |f(\mathbf{x} - \mathbf{d}) - g(\mathbf{x})|^2 + \lambda \|\mathbf{d} - \mathbf{d}_{\text{pred}}\|^2$$

  – The standard KLT term

  – A term that *drags* the solution towards the predicted displacement (cf. Kalman filtering)

Demo: KLT

- KLTdemo.m

Image Reconstruction

- Assume that $\mathbf{f}$ is an unknown image that is observed through the linear operator $\mathbf{G}$: $\mathbf{f}_0 = \mathbf{Gf} +$ noise
- Example: blurring, linear projection
- Goal is to minimize the error $\mathbf{f}_0 - \mathbf{Gf}$
- Example: squared error
- Assume that we have a prior probability for the image: $P(\mathbf{f})$
- Example: we assume that the image should be smooth (small gradients)

Image Reconstruction

- Minimizing

$$\varepsilon(\mathbf{f}) = \frac{1}{2}(|\mathbf{G}\mathbf{f} - \mathbf{f}_0|^2 + \lambda(|\mathbf{D}_x\mathbf{f}|^2 + |\mathbf{D}_y\mathbf{f}|^2))$$

- Gives the normal equations

$$\mathbf{G}^T\mathbf{G}\mathbf{f} - \mathbf{G}^T\mathbf{f}_0 + \lambda(\mathbf{D}_x^T\mathbf{D}_x\mathbf{f} + \mathbf{D}_y^T\mathbf{D}_y\mathbf{f}) = 0$$

- Such that

$$\mathbf{f} = (\mathbf{G}^T\mathbf{G} + \lambda(\mathbf{D}_x^T\mathbf{D}_x + \mathbf{D}_y^T\mathbf{D}_y))^{-1}\mathbf{G}^T\mathbf{f}_0$$

Gradient Operators

- Taylor expansion of image gives

$$f(x+h,y) = f(x,y) + hf_x(x,y) + \mathcal{O}(h^2)$$
$$f(x-h,y) = f(x,y) - hf_x(x,y) + \mathcal{O}(h^2)$$

- Finite left/right differences give

$$\partial_x^+ f = \frac{f(x+h,y) - f(x,y)}{h} + \mathcal{O}(h^2)$$
$$\partial_x^- f = \frac{f(x,y) - f(x-h,y)}{h} + \mathcal{O}(h^2)$$

- Often needed: products of derivative operators

Gradient Operators

- Squaring left (right) difference $(\partial_x^+)^2 f$ gives linear error in *h*
- Squaring central difference $\dfrac{f(x+h,y)-f(x-h,y)}{2h}$ gives quadratic error in *h,* but leaves out every second sample
- Multiplying left and right difference

$$\partial_x^+\partial_x^- f = \frac{f(x+h,y)-2f(x,y)+f(x-h,y)}{h^2} = \Delta_x f$$

gives quadratic error in *h* (usual discrete Laplace operator)

Demo: Image Reconstruction

- IRdemo.m

Another View: Variational Methods

- Minimize the local integral of a Lagrange function $L(f, f_x, f_y, x, y)$

$$\varepsilon(f) = \int_\Omega L(f, \nabla f, \mathbf{x})\, d\mathbf{x}$$

- gives the Euler-Lagrange equation on Ω

$$L_f - \operatorname{div} L_{\nabla f} = L_f - \partial_x L_{f_x} - \partial_y L_{f_y} = 0 \qquad \forall x, y$$

- if we require $\langle \nabla f | \mathbf{n} \rangle = 0 \quad \text{on } \partial\Omega$

Variational View: Linear Restoration

- Assume **G**=Id. Minimizing

$$\varepsilon(f) = \frac{1}{2}\int_\Omega \underbrace{(f-f_0)^2 + \lambda(f_x^2 + f_y^2)}_{L(f,f_x,f_y,x,y)}\,dx\,dy$$

- Gives the Euler-Lagrange equation (note: $L_f = f - f_0,\ L_{f_x} = \lambda f_x,\ L_{f_y} = \lambda f_y$)

$$\underbrace{f - f_0}_{L_f} - \underbrace{\lambda \Delta f}_{\mathrm{div}(L_{f_x}, L_{f_y})} = 0 \qquad (\partial_x f_x + \partial_y f_y) = \Delta f$$

cmp. $\mathbf{G}^T\mathbf{G}\mathbf{f} - \mathbf{G}^T\mathbf{f}_0 + \lambda(\mathbf{D}_x^T\mathbf{D}_x\mathbf{f} + \mathbf{D}_y^T\mathbf{D}_y\mathbf{f}) = 0$

Deblurring

- Minimizing

$$\varepsilon(f) = \frac{1}{2}\int_{\Omega} (g * f - f_0)^2 + \lambda(f_x^2 + f_y^2)\,dx\,dy$$

- Gives the Euler-Lagrange equation

$$g(-\cdot) * (g * f - f_0) - \lambda \Delta f = 0$$

- *g*: point spread function (PSF)
- *g*(-*x*): correlation operator / adjoint operator
- even symmetry PSF: self adjoint
- definition of adjoint operator $\langle x | Ay \rangle = \langle A^* x | y \rangle$

Demo: Deblurring

- DBdemo.m

Robust error norms

- Alternative to RANSAC (Monte-Carlo)
- Assume quadratic error: *influence* of change f to $f+\partial f$ to the estimate is linear (why?)
- Result on set of measurements: mean
- Assume absolute error: influence of change is constant (why?)
- Result on set of measurements: median
- In general: sub-linear influence leads to robust estimates, but *non-linear*

Smoothness

- Quadratic smoothness term: influence linear with height of edge
- Total variation smoothness (absolute value of gradient): influence constant
- With quadratic measurement error: Rudin-Osher-Fatemi (ROF) model (Physica D, 1992)

$$\min_f \frac{\|f - f_0\|^2}{2\lambda} + \sum_{i,j} |(\nabla f)_{i,j}|$$

Non-Linear Restoration

- Minimizing

$$\varepsilon(f) = \int_\Omega \frac{1}{2}(f - f_0)^2 + \lambda\Psi(|\nabla f|)\, dx\, dy$$

- Gives the Euler-Lagrange equation

$$f - f_0 - \lambda \operatorname{div}\left(\frac{\Psi'(|\nabla f|)}{|\nabla f|}\nabla f\right) = 0$$

where we exploited

$$\partial_x \frac{\Psi'(|\nabla f|)}{|\nabla f|} f_x + \partial_y \frac{\Psi'(|\nabla f|)}{|\nabla f|} f_y = \operatorname{div}\left(\frac{\Psi'(|\nabla f|)}{|\nabla f|}\nabla f\right)$$

Total Variation (TV) / ROF

- Minimizing $$\min_f \frac{\|f - f_0\|^2}{2\lambda} + \sum_{i,j} |(\nabla f)_{i,j}|$$

  means $$\Psi() = \mathrm{Id}() \Rightarrow \Psi'() = 1$$

- Stationary point

$$f - f_0 - \lambda \,\mathrm{div}\left(\frac{1}{|\nabla f|} \nabla f \right) = 0$$

  and after some calculations

$$f - f_0 - \lambda \frac{f_{xx} f_y^2 - 2 f_{xy} f_x f_y + f_{yy} f_x^2}{|\nabla f|^3} = 0$$

Efficient TV Algorithms

- In 1D: Chambolle's algorithm (JMIV, 2004)
- In 2D:
  - Alternating direction method of multipliers (ADMM, variant of augmented Lagrangian): Split Bregman by Goldstein & Osher (SIAM 2009)
  - Based on threshold Landweber: Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) by Beck & Teboulle (SIAM 2009)
  - Based on Lagrange multipliers: Primal Dual Algorithm by Chambolle & Pock (JMIV 2011)

Demo: TV Image Denoising

- Paramters: α =0.0005, λ=0.5, noise(0,0.001), TVdemo_script.m

TV Image Inpainting / Convex Optimization

- Note that many problems (including quadratic and TV) are convex optimization problems
- A good first approach: map these problems to a standard solver, e.g. CVXPY (S. Diamond & S. Boyd)
- Example: minimize the total variation of an image

$$\sum_{i,j} |(\nabla f)_{i,j}|$$ under the constraint of a subset of known image values f

```
prob=Problem(Minimize(tv(X)),[X[known] == MG[known]])

opt_val = prob.solve()
```

Demo: TV Inpainting

- inpaint.py

Algorithmic Taxonomy

- Minimal problems (e.g. 5 point algorithm)
  - Fully determined solution(s)
  - Analytic solvers (polynomials, Gröbner bases)
  - Numerical methods (Dogleg, Newton-Raphson)
- Overdetermined problems (e.g. OF,BA)
  - Minimization problem
  - Numerical solvers only
  - Levenberg-Marquardt (interpolation Gauss-Newton and gradient descent / trust region)

Non-linear LS, Dog Leg

- For comparison: LM $\mathbf{r}(\mathbf{x}+\boldsymbol{\delta}) \approx \mathbf{r}(\mathbf{x}) + \mathbf{J}\boldsymbol{\delta}$

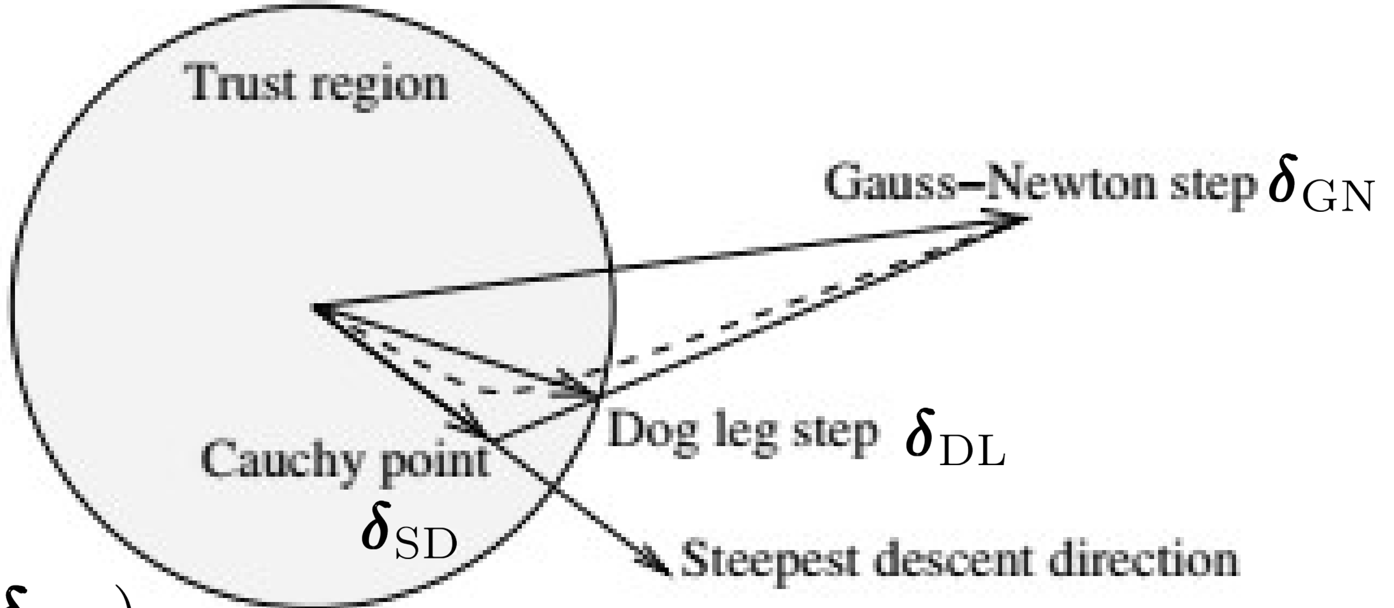
$$(\mathbf{J}^T\mathbf{J} + \lambda\,\mathbf{diag}(\mathbf{J}^T\mathbf{J}))\,\boldsymbol{\delta} = \mathbf{J}^T\mathbf{r}(\mathbf{x})$$

$$x_j \mapsto x_j + \delta_j \qquad J_{ij} = \frac{\partial r_i}{\partial x_j}$$

- More efficient: replace damping factor λ with trust region radius Δ

method	abbr.	properties
steepest descent	SD	$\boldsymbol{\delta} = \mathbf{J}^T\mathbf{r}$
Gauss-Newton	GN	$\mathbf{J}^T\mathbf{J}\boldsymbol{\delta} = \mathbf{J}^T\mathbf{r}$
Levenberg-Marquardt	LM	combines SD and GN by damping factor
Dog Leg	DL	combines SD and GN by trust region radius Δ

Dog Leg (basic idea)



1. initialize radius Δ =1
2. compute gain factor

3. if gain factor >0

$$\mathbf{x}_{\mathrm{new}} = \mathbf{x} + \underbrace{\boldsymbol{\delta}_{\mathrm{SD}} + \alpha(\boldsymbol{\delta}_{\mathrm{GN}} - \boldsymbol{\delta}_{\mathrm{SD}})}_{\boldsymbol{\delta}_{\mathrm{DL}}}$$

$$\|\boldsymbol{\delta}_{\mathrm{SD}}\| \leq \Delta\,, \quad 0 \leq \alpha \leq 1\,, \quad \|\boldsymbol{\delta}_{\mathrm{DL}}\| = \Delta$$

4. grow/shrink Δ and update gain factor
5. if update and residual nonzero goto 3

Optical Flow

- Minimizing (lecture 4) $\varepsilon(\mathbf{v}_h) = \sum_{\mathcal{R}} w |[\nabla^T f \ f_t]\mathbf{v}_h|^2$
- Under the constraint $|\mathbf{v}_h|^2 = 1$
- Using Lagrangian multiplier leads to the minimization problem

$$\varepsilon_T(\mathbf{v}_h, \lambda) = \varepsilon(\mathbf{v}_h) + \lambda(1 - |\mathbf{v}_h|^2)$$

- This is the *total least squares* formulation to determine the flow

Optical Flow

- Solution is given by the eigenvalue problem

$$\left(\sum_{\mathcal{R}} w \begin{bmatrix} \nabla f \\ f_t \end{bmatrix} \begin{bmatrix} \nabla^T f & f_t \end{bmatrix} \right) \mathbf{v}_h = \lambda \mathbf{v}_h$$

$$\mathbf{T}\mathbf{v}_h = \lambda \mathbf{v}_h$$

- The matrix term **T** is the spatio-temporal structure tensor
- The eigenvector with the smallest eigenvalue is the solution (up to normalization of homogeneous element)

Optical Flow

- *Local* flow estimation
  - Design question: w and *R*
  - Aperture problem: motion at linear structures can only be estimated in normal direction (underdetermined)
  - Infilling limited
- *Global* flow instead

Optical Flow

- Minimizing BCCE over the whole image with additional smoothness term

$$\varepsilon(\mathbf{v}) = \frac{1}{2}\int_\Omega \overbrace{(\langle \mathbf{v}|\nabla f\rangle + f_t)^2}^{\text{BCCE}} + \lambda(|\nabla v_1|^2 + |\nabla v_2|^2)\,dx\,dy$$

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

- Gives the Euler-Lagrange equation

$$(\langle \mathbf{v}|\nabla f\rangle + f_t)\nabla f - \lambda\Delta\mathbf{v} = 0$$

- Laplacian is approximately

$$\Delta\mathbf{v} \approx \bar{\mathbf{v}} - \mathbf{v}$$

1	1	1	-	3	·	0	1	0	=	1	-2	1

Optical Flow

- Plugging into the EL-equation gives

$$(\lambda + \nabla f \nabla f^T)\mathbf{v} = \lambda \bar{\mathbf{v}} - f_t \nabla f$$

- Explicitly solving for **v** results in

$$\begin{aligned}
(\lambda + \nabla f \nabla f^T)\mathbf{v} &= (\lambda + \nabla f \nabla f^T)\bar{\mathbf{v}} - (\nabla f \nabla f^T \bar{\mathbf{v}} + \nabla f f_t) \\
&= (\lambda + \nabla f \nabla f^T)\bar{\mathbf{v}} - \nabla f (\nabla f^T \bar{\mathbf{v}} + f_t) \\
&= (\lambda + \nabla f \nabla f^T)\bar{\mathbf{v}} - \frac{\lambda + \nabla f^T \nabla f}{\lambda + \nabla f^T \nabla f} \nabla f (\nabla f^T \bar{\mathbf{v}} + f_t) \\
&= (\lambda + \nabla f \nabla f^T)\bar{\mathbf{v}} - \frac{\lambda + \nabla f \nabla f^T}{\lambda + \nabla f^T \nabla f} \nabla f (\nabla f^T \bar{\mathbf{v}} + f_t) \\
\mathbf{v} &= \bar{\mathbf{v}} - \frac{1}{\lambda + \nabla f^T \nabla f} \nabla f (\nabla f^T \bar{\mathbf{v}} + f_t)
\end{aligned}$$

Optical Flow

- Iterating the solution

$$\mathbf{v} = \bar{\mathbf{v}} - \frac{1}{\lambda + \nabla f^T \nabla f} \nabla f (\nabla f^T \bar{\mathbf{v}} + f_t)$$

results in the Horn & Schunck iteration

$$\mathbf{v}^{(s+1)} = \bar{\mathbf{v}}^{(s)} - \frac{1}{\lambda + |\nabla f|^2} (\langle \bar{\mathbf{v}}^{(s)} | \nabla f \rangle + f_t) \nabla f$$

- Significant improvement: use median instead of $\bar{\mathbf{v}}$!

Demo: Horn & Schunck

- HSdemo.m

Graph Algorithms

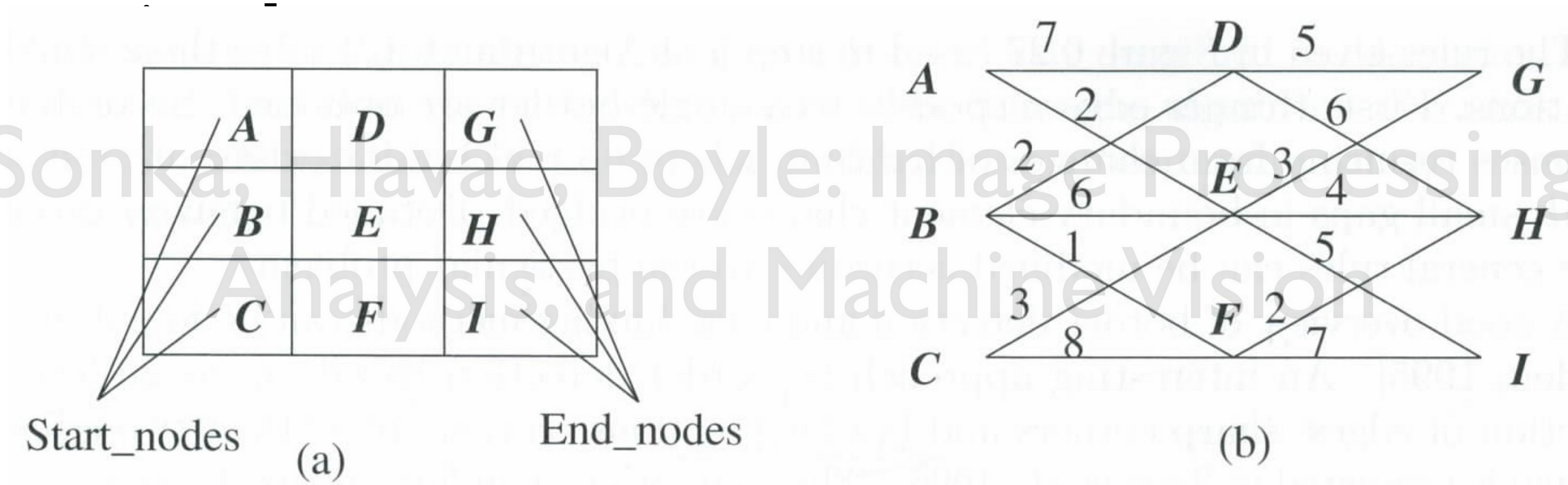
- All examples so far: vectors as solutions, i.e. finite set of (pseudo) continuous values
- Now: discrete (and binary) values
- Directly related to (labeled) graph-based optimization
- In probabilistic modeling (on regular grid): Markov random fields

Graphs

- Graph: algebraic structure $G=(V, E)$
- Nodes $V=\{v_1,v_2,\ldots,v_n\}$
- Arcs $E=\{e_1,e_2,\ldots,e_m\}$, where e_k is incident to
  - an unordered pair of nodes $\{v_i,v_j\}$
  - an ordered pair of nodes (v_i,v_j) (directed graph)
  - degree of node: number of incident arcs
- Weighted graph: costs assigned to nodes or arcs

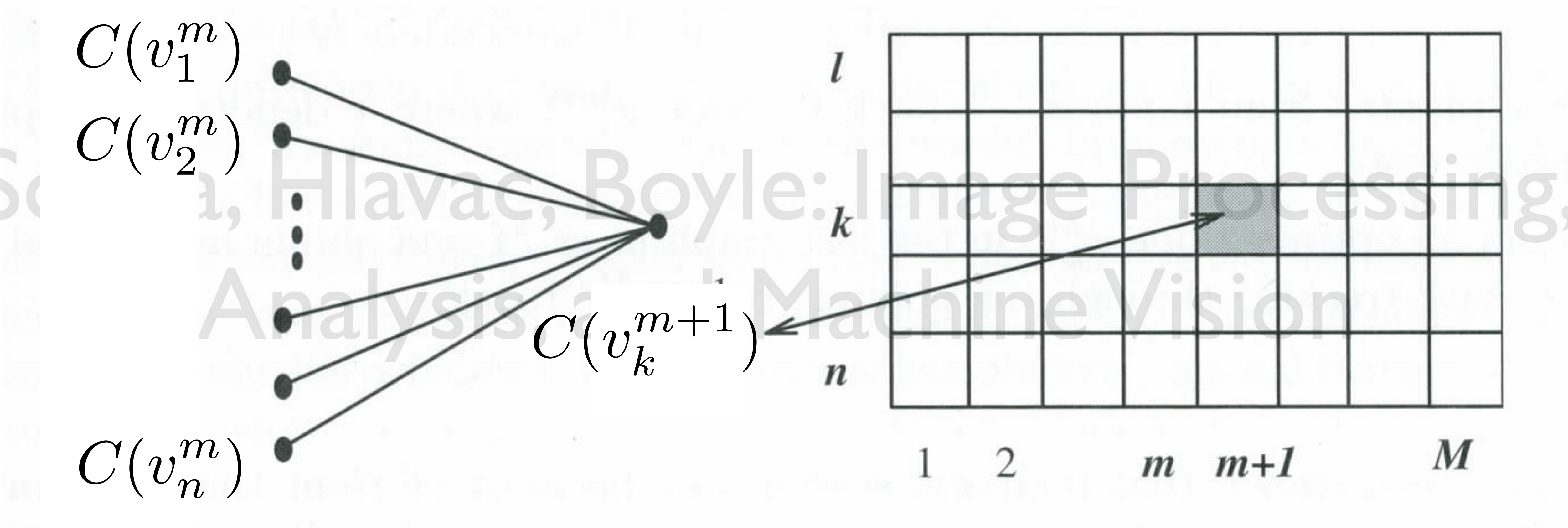
1D: Dynamic Programming

- Problem: find optimal path from source node *s* to sink note *t*
- Principle of Optimality: If the optimal path *s*-*t* goes through *r,* then both *s*-*r* and *r*-*t,* are also



1D: Dynamic Programming

- $C(v_k^{m+1})$ is the new cost assigned to node v_k
- $g^m(i,k)$ is the partial path cost between nodes v_i and v_k



1D: Dynamic Programming

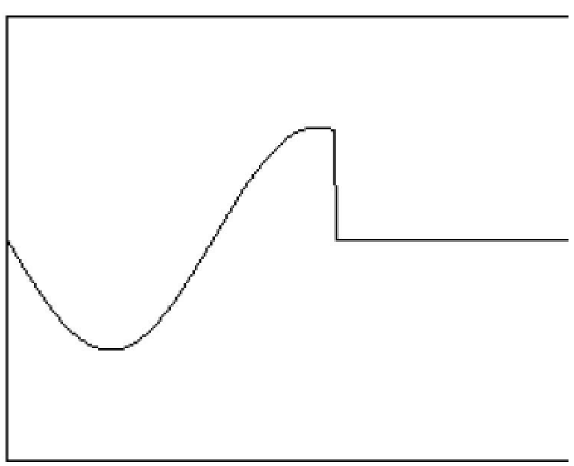
- $C(v_k^{m+1})$ is the new cost assigned to node v_k
- $g^m(i,k)$ is the partial path cost between nodes v_i and v_k

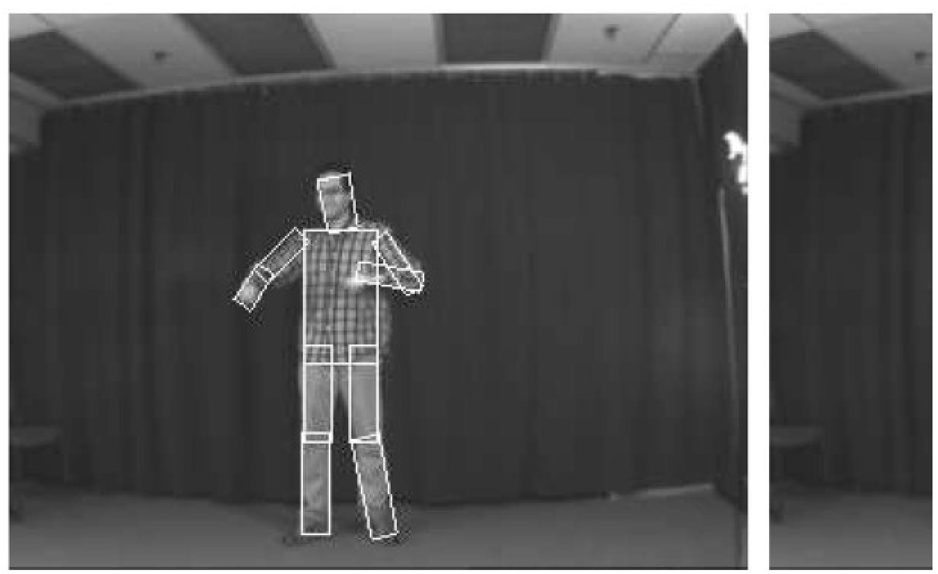
$$C(v_k^{m+1}) = \min_i \left(C(v_i^m) + g^m(i,k)\right)$$

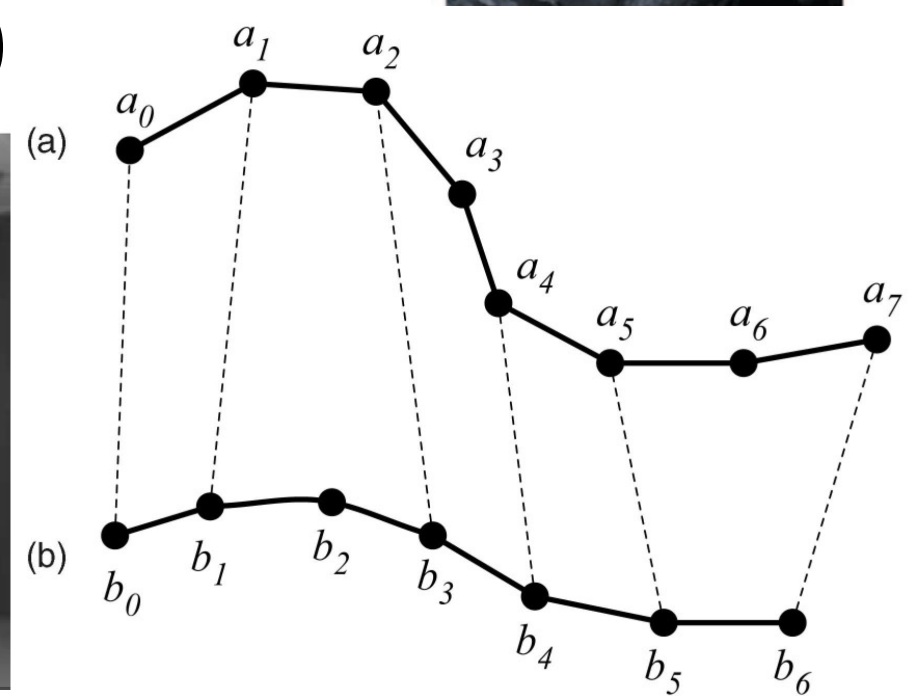
$$\min\left(C(v^1, v^2, \ldots, v^M)\right) = \min_{k=1,\ldots,n}\left(C(v_k^M)\right)$$

Examples

- Shortest path computation (contours / intelligent scissors)
- 1D signal restoration (denoising)
- Tree labeling (pictorial structures)
- Matching of sequences (curves)





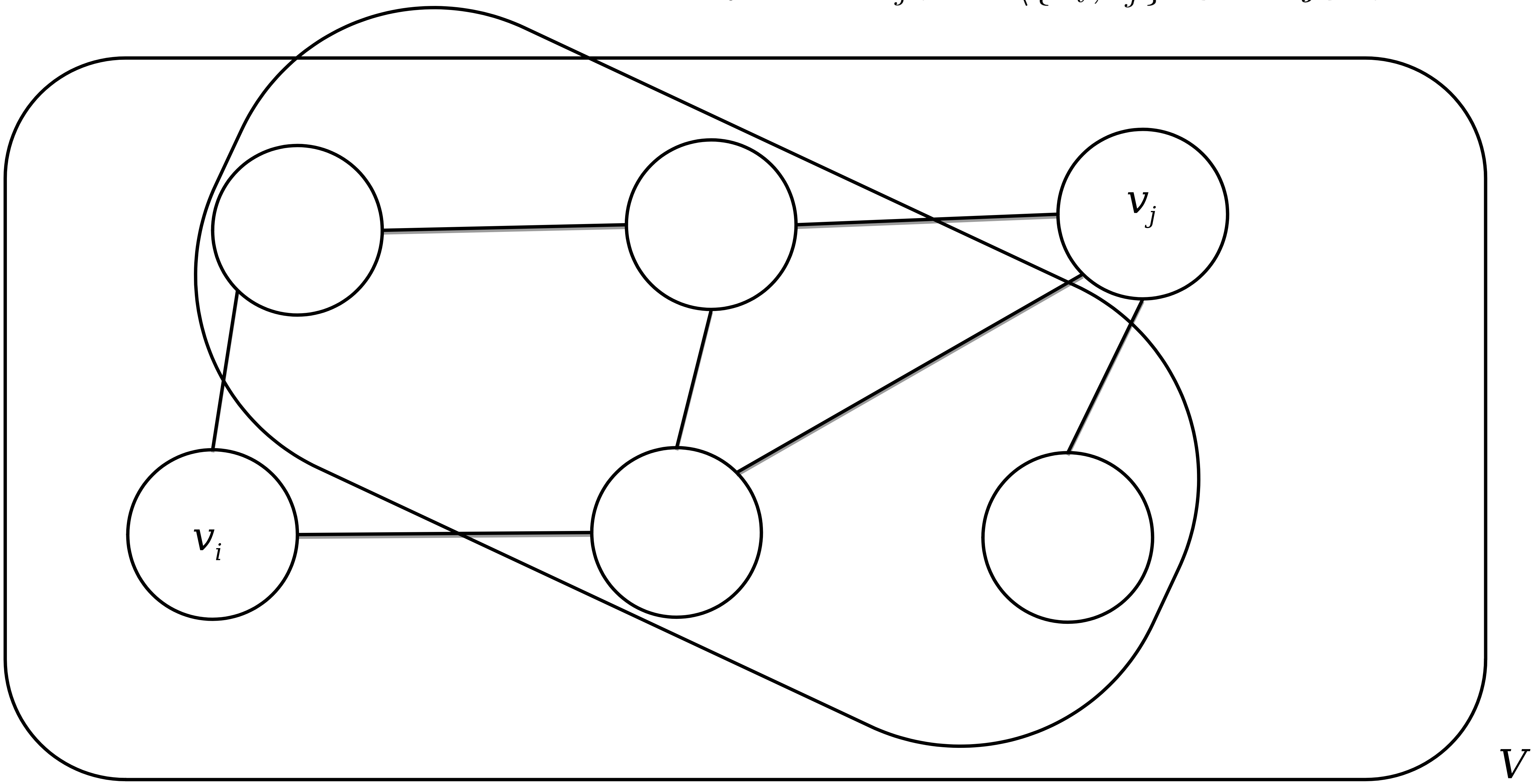


Markov property

- Markov chain: memoryless process with r.v. X
- Markov random field (undirected graphical model): random variables (e.g. labels) over nodes with Markov property (conditional independence)
  - Pairwise $X_{v_i} \perp\!\!\!\perp X_{v_j} | X_{V \setminus \{v_i, v_j\}}$ $\{v_i, v_j\} \notin E$
  - Local $X_v \perp\!\!\!\perp X_{V \setminus (\{v\} \cup N(v))} | X_{N(v)}$
  - Global $X_A \perp\!\!\!\perp X_B | X_S$ where every path from a node in A to node in B passes through S

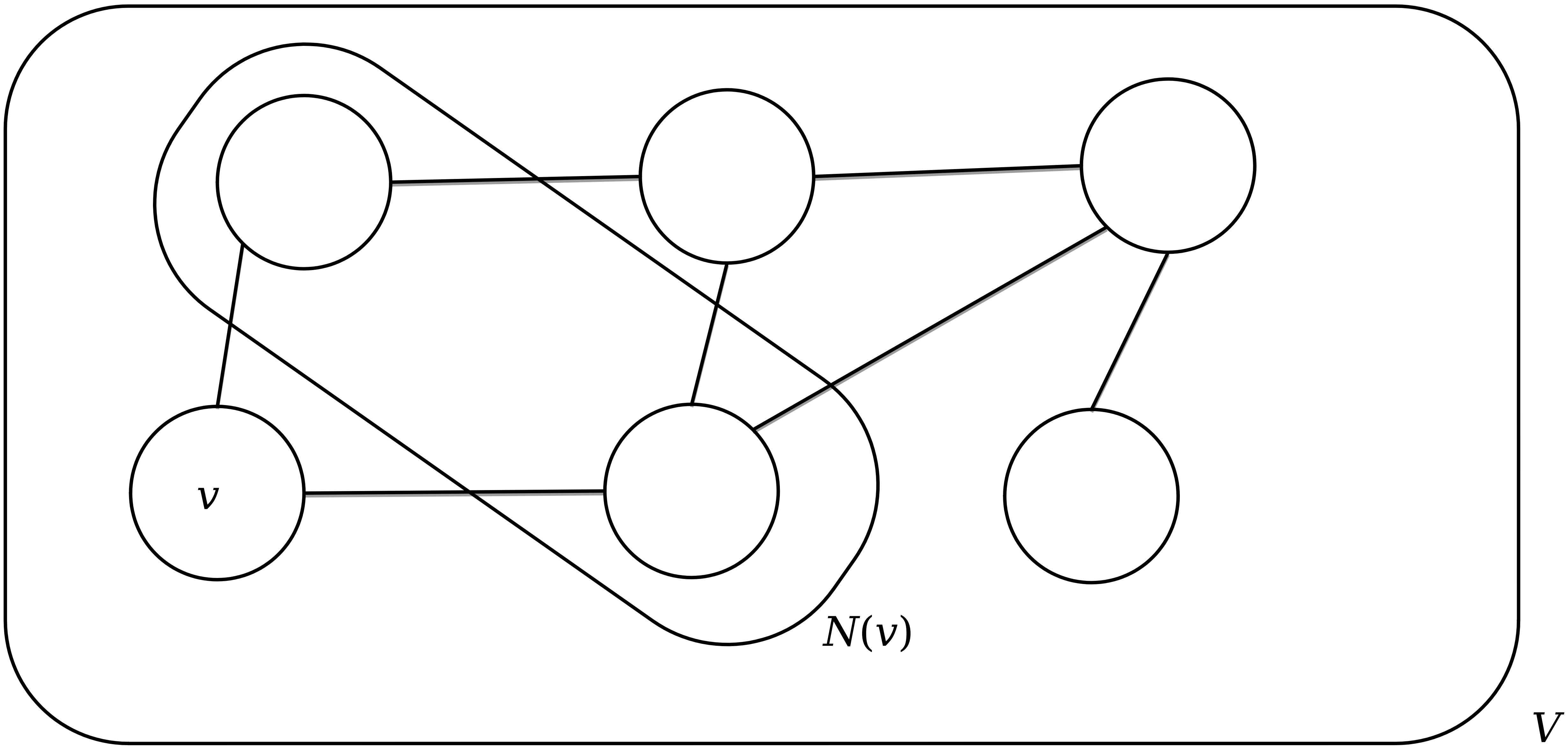
Conditional Independence

$$X_{v_i} \perp\!\!\!\perp X_{v_j} | X_{V \setbackslash \{v_i, v_j\}} \quad \{v_i, v_j\} \notin E$$



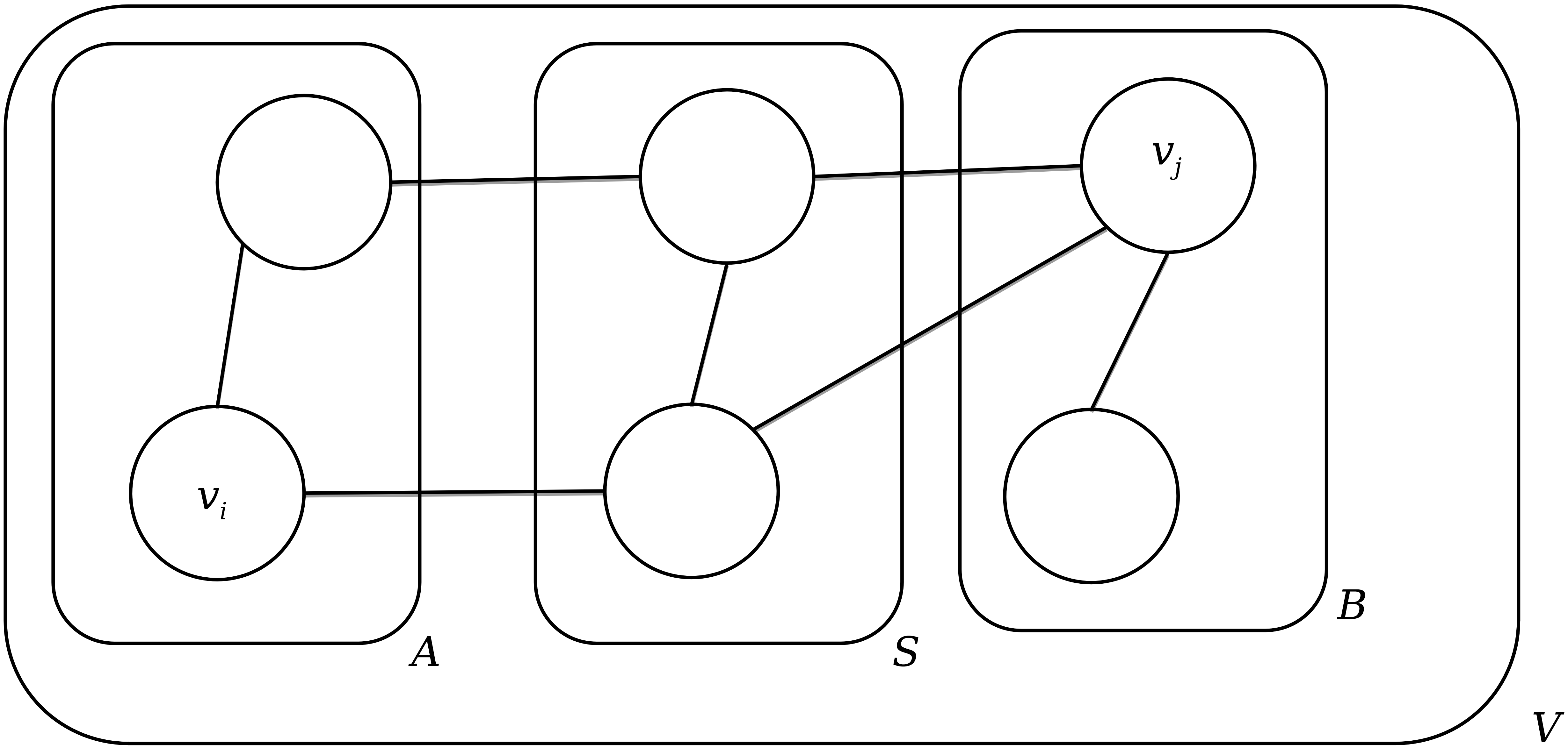
Conditional Independence

$$X_v \perp\!\!\!\perp X_{V \setminus (\{v\} \cup N(v))} | X_{N(v)}$$



Conditional Independence

$$X_A \perp\!\!\!\perp X_B | X_S$$



Terminology

- If joint density strictly positive: Gibbs RF
- Ising model (interacting magnetic spins), energy given as Hamiltonian function

$$\varepsilon(X_V) = -\sum_{e_k=\{v_i,v_j\}\in E} J_{e_k} X_{v_i} X_{v_j} - \sum_{v_j} h_{v_j} X_{v_j}$$

- General form

$$\varepsilon(X_V) = \lambda \sum_{e_k=\{v_i,v_j\}\in E} V(X_{v_i}, X_{v_j}) + \sum_{v_j} D(X_{v_j})$$

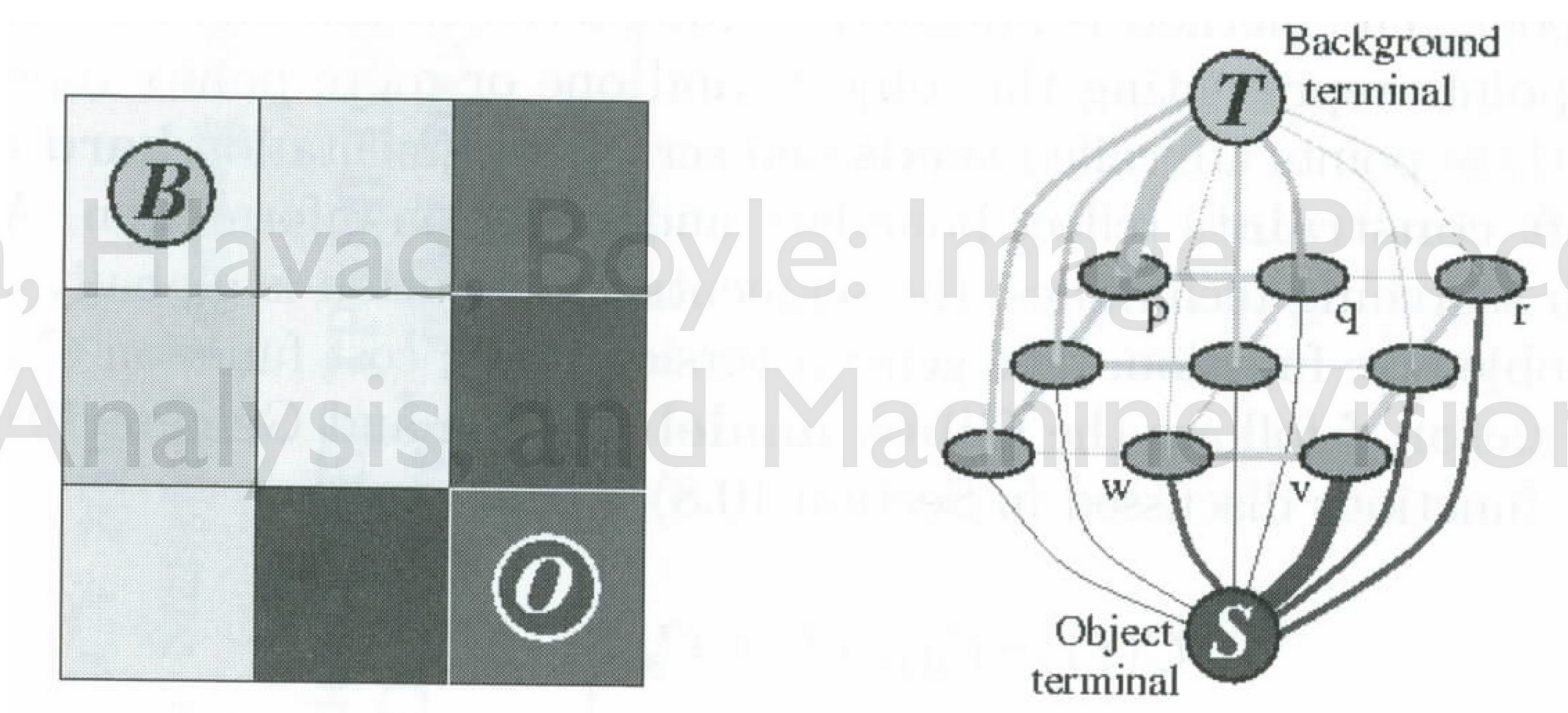
- Configuration probability $P(X_V) \propto \exp(-\varepsilon(X_V))$

Gibbs Model / Markov Random Field

- Attempts to generalize dynamic programming to higher dimensions unsuccessful
- Minimize $C(f) = C_{\mathrm{data}}(f) + C_{\mathrm{smooth}}(f)$ using arc-weighted graphs $G_{\mathrm{st}} = (V \cup \{s, t\}, E)$
- Two special terminal nodes, source *s* (e.g. object) and sink *t* (e.g. background) hard-linked with seed points

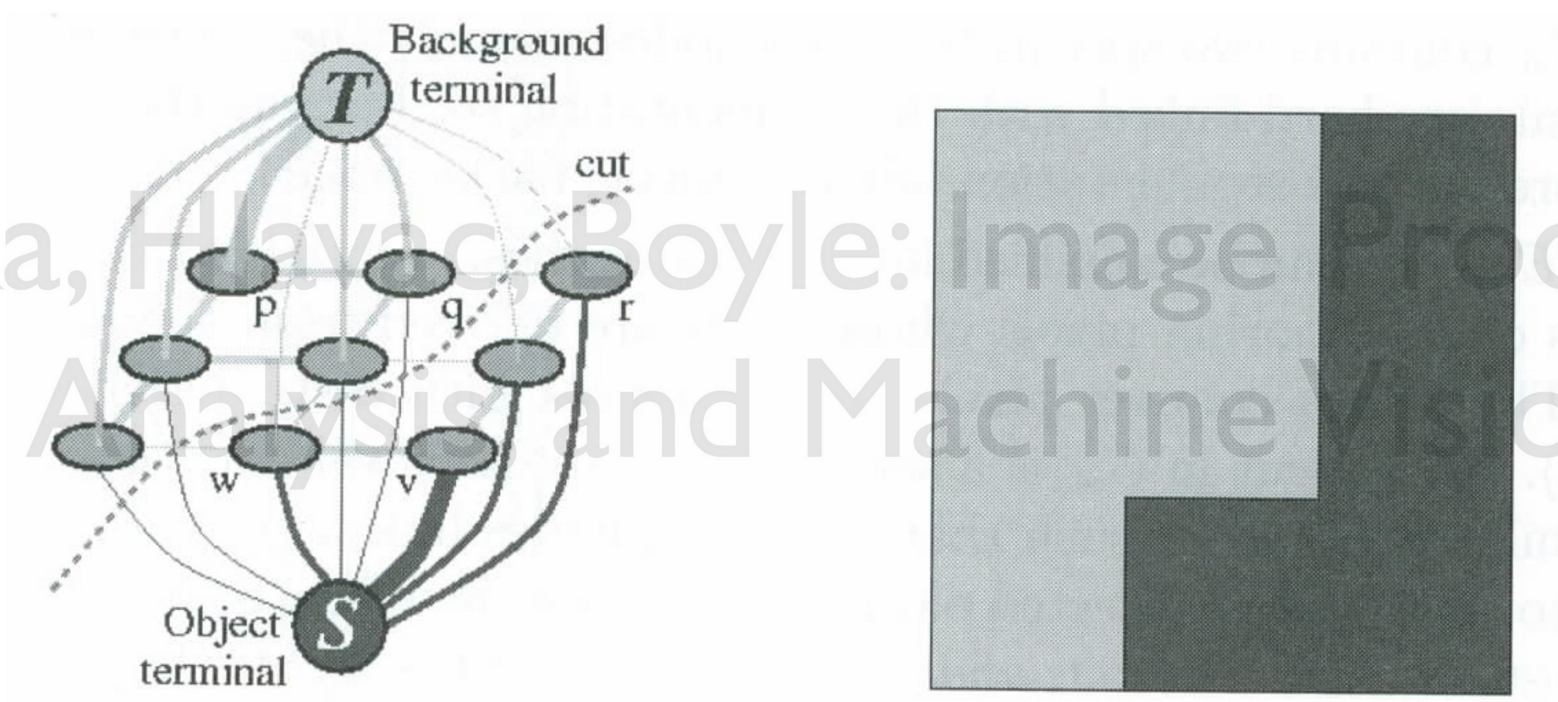
Graph Cut: Two types of arcs

- n-links: connecting neighboring pixels, cost given by the smoothness term *V*
- t-links: connecting pixels and terminals, cost given by the data term *D*



Graph Cut

- *s*-*t* cut: set of arcs, such that the nodes and the remaining arcs form two disjoint graphs with points sets *S* and *T*
- cost of cut: sum of arc cost
- minimum *s*-*t* cut problem (dual: maximum flow problem)



Graph Cut

- n-link costs: large if two nodes belong to same segment, e.g. inverse gradient magnitude, Gaussian function, Potts model
- t-link costs:
  - K for hard-linked seed points (K > maximum sum of data terms)
  - 0 for the opposite seed point
- Submodularity $V(\alpha, \alpha) + V(\beta, \beta) \leq V(\alpha, \beta) + V(\beta, \alpha)$

Demo: Graph cut

- gc_example.m

Examples / Discussion

- Binary problems solvable in polynomial time (albeit slow)
  - Binary image restoration
  - Bipartite matching (perfect assignment of graphs)
- N-ary problems (more than two terminals) are NP-hard and can only be approximated (e.g. α-expansion move)

Michael Felsberg
michael.felsberg@liu.se

LINKÖPINGS UNIVERSITET