



**MATEMATICKO-FYZIKÁLNÍ
FAKULTA**
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Klára Ficová

Rekonštrukcia obrazu pomocou grafických modelov

Katedra algebry

Vedoucí bakalářské práce: RNDr. Alexandr Kazda, Ph.D.

Studijní program: Matematika

Studijní obor: Matematika pro informační technologie

Praha 2020

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V dne

Podpis autora

Týmto by som sa chcela poďakovať vedúcemu práce RNDr. Alexandrovi Kazdovi za jeho pomoc, trpezlivosť a veľmi ochotný prístup.

Název práce: Rekonštrukcia obrazu pomocou grafických modelov

Autor: Klára Ficová

Katedra: Katedra algebry

Vedoucí bakalářské práce: RNDr. Alexandr Kazda, Ph.D., Katedra algebry

Abstrakt: Grafické modely slúžia na reprezentáciu pravdepodobnostných vzťahov medzi náhodnými veličinami pomocou grafu. Ponúkajú dobrý spôsob na vyjadrenie reálnych situácií a preto sa často využívajú v strojovom učení a štatistickom uvažovaní. Cieľom práce je popísať a implementovať spôsoby, ako grafickým modelom odstrániť z obrazu šum. Za grafický model zvolíme faktorgraf, v ktorom reprezentujeme ako vrcholy pixely v obraze a interakcie medzi nimi. Pomocou algoritmov popísaných na grafe budeme hľadať najpravdepodobnejší pôvodný obraz. Bližšie sa budeme venovať algoritmu belief propagation, ktorý je založený na vzájomnom posielaní správ medzi susednými vrcholmi. Teoretické metódy aplikujeme na obrazy so šumom a porovnáme výsledky.

Klíčová slova: rekonštrukcia obrazu, grafické modely, faktorgrafy, belief propagation

Title: Image reconstruction using graphical models

Author: Klára Ficová

Department: Department of Algebra

Supervisor: RNDr. Alexandr Kazda, Ph.D., Department of Algebra

Abstract: Graphical models represent probability relations among random variables using a graph. They offer an effective way of modelling real life situations and are frequently used in machine learning and statistical thinking. The main goal of this thesis is to describe and implement techniques of denoising an image using graphical models. The graphical model we choose is factorgraph, representing pixels in image and interactions between them as nodes. Using algorithms on the graph, we determine the most probable true image. We also apply those theoretical methods on noisy images and compare the results.

Keywords: image reconstruction, graphical models, factorgraphs, belief propagation

Obsah

Úvod	2
1 Základné pojmy	3
1.1 Pravdepodobnosť	3
1.2 Konvexné a konkávne funkcie	3
1.3 Grafy	4
1.4 Grafické modely	4
1.4.1 Faktorgrafy a belief propagation algoritmus	5
1.5 Diskrétna konvolúcia a Fourierova transformácia	12
2 Aplikácia na obrazy	14
2.1 Obraz	14
2.2 Aplikácia grafických modelov	14
2.2.1 Reprezentácia faktorgrafom	14
2.2.2 Voľba faktorizujúcich funkcií	16
2.2.3 Belief propagation	18
3 Implementácia	20
3.1 Popis postupov	20
3.1.1 Konvexná optimalizácia	20
3.1.2 Najmenšie štvorce	20
3.1.3 Belief propagation	20
3.2 Výsledky na obrazoch	22
3.3 Porovnanie výsledkov	22
Záver	26
Zoznam použitej literatúry	27
Zoznam obrázkov	28

Úvod

Pri zachytávaní obrazov často vzniká nepriaznivý vedľajší efekt – šum. Jeden z najzákladnejších spôsobov, ako ho odstrániť, je použiť filter, ktorý sa snaží zbaviť sa šum priemerovaním pixelov. Takto avšak stratíme iné vlastnosti obrazu.

V tejto práci pomocou grafických modelov navrhujeme a implementujeme zložitejšie metódy. Grafické modely znázorňujú vzťahy medzi náhodnými veličinami a často sa opierajú o predpoklad, že vieme faktorizovať združenú pravdepodobnosť.

Model, ktorý nám pomôže rekonštruovať obraz, navrhujeme podľa spôsobu prezentovaného v [1]. Budeme vychádzať z Bayesovej vety a hľadať budeme najpravdepodobnejší pôvodný obraz za podmienky zašumeného obrazu.

Pravdepodobnosť budeme maximalizovať troma spôsobmi. Prvý sa opiera o konvexnú optimalizáciu a využíva to, že konvexné funkcie umožňujú jednoduchšie hľadanie minima. Druhý spôsob je veľmi jednoduchý, jedná sa o najmenšie štvorce ako špeciálny prípad konvexnej optimalizácie. O niečo viac sa budeme venovať tretej metóde. Použijeme algoritmus *belief propagation* ako v článku [2], ktorý je založený na posielaní správ medzi vrcholmi grafu. Takisto pomocou článku [3] navrhujeme jeho zrýchlenie diskretnou Fourierovou transformáciou.

V závere práce ukážeme výsledky rekonštrukcie rôznymi spôsobmi. Použité metódy zhodnotíme z viacerých hľadísk. Do úvahy vezmeme rýchlosť algoritmu, veľkosť obrazov, ktoré dokázal spracovať, množstvo odstráneného šumu a podobnosť s pôvodným obrazom.

Vlastný prínos sa v prvom rade skladá z implementácie algoritmov. Ďalej sme spracovali podrobnejšie popis *belief propagation* a ostatných metód. Takisto dôkazy všetkých tvrdení v tejto práci sú pôvodné.

1. Základné pojmy

V tejto kapitole predstavíme pojmy, ktoré nám poskytnú teoretický základ pre navrhnutie modelu pre rekonštrukciu obrazu a algoritmi, ktoré použijeme.

1.1 Pravdepodobnosť

V tejto práci budeme obrazy rekonštruovať hľadaním najpravdepodobnejšieho obrazu. Na pixely v obraze sa budeme dívať ako na náhodné veličiny a náhodný jav bude intenzita pixelu.

Budeme uvažovať pravdepodobnostný priestor (Ω, F, P) , kde Ω je konečná, neprázdna množina, F je systém všetkých podmnožín Ω a P je pravdepodobnosť funkcia $P : F \rightarrow [0,1]$. *Náhodnou veličinou* nazveme zobrazenie $X : \Omega \rightarrow \mathbb{R}$ také, že $\{\omega : X(\omega) \leq a\} \in F$ pre všetky $a \in \mathbb{R}$.

Definícia. Pre náhodné veličiny X_1, X_2, \dots, X_n nazveme

$$p(x) = P(X_1 = x_1, X_2 = x_2, \dots, X_n = x_n)$$

združenou pravdepodobnosťou. Potom definujeme $p_i(x_i) = P(X_i = x_i)$ ako marginálnu pravdepodobnosť.

Ďalej si uvedieme Bayesovu vetu, pomocou ktorej navrhne model pre rekonštrukciu obrazu:

Veta 1 (Bayesova). *Nech A, B sú náhodné javy, také, že $P(A) > 0$ a $P(B) > 0$. Potom*

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

1.2 Konvexné a konkávne funkcie

Na hľadanie najpravdepodobnejšieho obrazu použijeme aj konvexné funkcie. Konvexné funkcie totiž umožňujú dobrú minimalizáciu, s nimi súvisia aj konkávne funkcie, ktorá zas umožňujú maximalizovať. Použijeme definície a tvrdenia z [4].

Definícia. *Množina C sa nazýva konvexná, pokiaľ pre každé $x_1, x_2 \in C$ a každé $\lambda \in \mathbb{R}, 0 \leq \lambda \leq 1$ platí:*

$$\lambda x_1 + (1 - \lambda)x_2 \in C.$$

Definícia. *Funkcia f sa nazýva konvexná, pokiaľ $\text{dom} f$ je konvexná a pre každé $x_1, x_2 \in \text{dom} f$ a každé $\lambda \in \mathbb{R}, 0 \leq \lambda \leq 1$ platí:*

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2).$$

Povieme, že funkcia f je konkávna, ak je $-f$ konvexná.

Definícia. *Funkcia f sa nazýva log-konkávna, pokiaľ $f(x) > 0$ pre každé $x \in \text{dom} f$ a funkcia $\log f$ je konkávna. Funkcia f sa nazýva log-konvexná, pokiaľ je $\log f$ konvexná.*

Využijeme následovné vlastnosti:

- Každá afínna funkcia je konvexná a zároveň konkávna.
- Súčet konvexných (konkávnych) funkcií je opäť konvexná (konkávna) funkcia.
- Každé lokálne minimum konvexnej funkcie je zároveň globálne minimum. Podobne, každé lokálne maximum konkávnej funkcie je zároveň globálne maximum.

1.3 Grafy

Ďalej uvedieme niekoľko základných pojmov týkajúcich sa grafov, ktoré využijeme. Pojmy definujeme ako v [5].

Definícia. Grafom rozumieme usporiadanú dvojicu (V, E) , kde V je nejaká neprázdna množina a E je množina dvojbodových podmnožín množiny V . Prvky množiny V sa nazývajú vrcholy a prvky množiny E hrany grafu.

Definícia. Graf sa nazýva bipartitný, pokiaľ môžeme množinu vrcholov rozložiť na dve disjunktné podmnožiny V_1 a V_2 také, že každá hrana spája vrchol z V_1 s vrcholom z V_2 .

Definícia. Cestou z vrcholu v_0 do vrcholu v_t dĺžky t rozumieme postupnosť

$$(v_0, e_1, v_1, \dots, e_t, v_t),$$

kde v_0, v_1, \dots, v_t sú navzájom rôzne vrcholy grafu a pre každé $i = 1, \dots, t$ je hrana $e_i = \{v_{i-1}, v_i\}$.

Definícia. Strom je graf taký, že pre každú dvojicu vrcholov v_1, v_2 existuje cesta z v_1 do v_2 a zároveň neexistuje cesta dĺžky aspoň 1 taká, že $v_0 = v_t$.

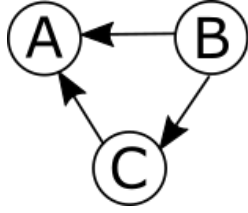
Definícia. Nech v je vrchol v strome. Povieme, že v je list, pokiaľ má najviac jeden susedný vrchol.

1.4 Grafické modely

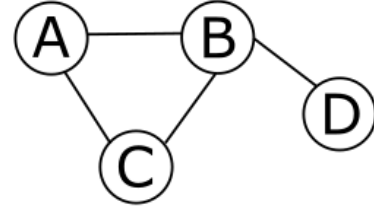
Grafické modely slúžia na vyjadrenie podmienok medzi náhodnými veličinami pomocou grafu. Často sa využívajú v strojovom učení, keďže ponúkajú dobrý spôsob modelovania skutočnej situácie.

Jedným z najpoužívanejších, ktorý si uvedieme ako príklad, je *Bayesova sieť* (Bayesian network). Tento model je orientovaný acyklický graf, v ktorom hrany určujú závislosti medzi náhodnými veličinami. Teda veličina A je závislá na veličine B , ak v je v grafe hrana z vrcholu B do vrcholu A .

Ďalším často používaným modelom je *Markovovo náhodné pole* (Markov random field). Tu je rozdiel od Bayesovej siete graf neorientovaný a môže byť aj cyklický. V tomto grafe predstavujú vrcholy náhodné veličiny a hrany opäť závislosti, t.j. veličina je závislá na všetkých svojich susedoch.



Obrázok 1.1: Príklad Bayesovej siete. Veličina A je závislá na veličinách B a C , C je závislá na veličinách B .



Obrázok 1.2: Príklad Markovovho poľa. Veličina A je závislá na veličinách B a C , B je závislá na veličinách A, C a D , C je závislá na veličinách A a B a D je závislá na B .

Keďže tieto modely nebudeme používať na rekonštrukciu obrazov, nebudeme sa na nich bližšie pozerat'. Bayesove siete totiž dobre nevyjadrujú závislosti pixelov v obraze, susedné pixely sú závislé navzájom. Markovove polia sa síce využívajú na rekonštrukciu obrazu, ale pre naše potreby si vystačíme s faktorgrafmi, ktoré predstavíme v ďalšej časti. Faktorgrafy nám totiž už poskytujú faktorizáciu pravdepodobnosti, ktoré potrebujeme pre maximalizáciu pravdepodobnosti obrazu.

1.4.1 Faktorgrafy a belief propagation algoritmus

Grafickým modelom, ktorý aplikujeme, je faktorgraf. *Faktorgraf* je definovaný ako bipartitný graf, ktorý reprezentuje faktorizáciu funkcie. V našom prípade je faktorizovaná funkcia združená pravdepodobnosť.

Uvážme diskkrétne náhodné veličiny X_1, X_2, \dots, X_n s hodnotami v množine H . Budeme predpokladať, že sa združená pravdepodobnosť $p(X_1 = x_1, X_2 = x_2, \dots, X_n = x_n)$, kde $x_i \in H$ (skrátene $p(x_1, x_2, \dots, x_n)$) faktorizuje nasledujúcim spôsobom:

$$p(x_1, x_2, \dots, x_n) = \frac{1}{Z} \prod_{a=1}^M f_a(\arg_a), \quad (1.1)$$

kde f_1, f_2, \dots, f_M sú nezáporné funkcie, pre každú f_a je $\arg_a \subseteq \{x_1, \dots, x_n\}$ množina argumentov funkcie f_a , a Z je normalizačná konštanta.

Vo faktorgrafe budeme mať dva druhy vrcholov – vrcholy premenných (resp. náhodných veličín) pre každú premennú X_1, X_2, \dots, X_n , ktoré budeme indexovať číslami $i \in \{1, 2, \dots, n\}$, a vrcholy faktorov pre každú f_1, f_2, \dots, f_M , tie budeme takisto indexovať číslami $a \in \{1, 2, \dots, M\}$. Hrana medzi vrcholmi v_1 a v_2 bude práve vtedy, ak v_1 bude vrchol premennej a v_2 vrchol faktoru také, že premenná reprezentovaná vrcholom v_1 je argumentom funkcie reprezentovanej vrcholom v_2 .

Faktorgraf využijeme na vypočítanie marginálnej pravdepodobnosti, teda pre vypočítanie $p(X_i = h)$, pre ktorú platí

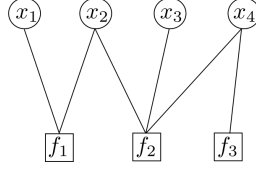
$$p(X_i = h) = \sum_{\substack{x \in H^n \\ x_i = h}} p(x) \quad (1.2)$$

$$= \frac{1}{Z} \sum_{\substack{x \in H^n \\ x_i = h}} \prod_{a=1}^M f_a(x), \quad (1.3)$$

Príklad. Faktorgraf pre združenú pravdepodobnosť

$$p(x_1, x_2, x_3, x_4) = \frac{1}{Z} f_1(x_1, x_2) f_2(x_2, x_3, x_4) f_3(x_4)$$

vyzerá nasledovne:



Zavedieme algoritmus *belief propagation* (šírenie viery), značíme BP. Prevezmeme ho z článku [2], s tým rozdielom, že my namiesto Markovovho poľa použijeme faktorgraf. Tento algoritmus je iteračný algoritmus založený na posielaní správ medzi vrcholmi premenných a vrcholmi faktorov a jeho cieľom je dostať sa do stabilizovaného stavu, v ktorom dostane každý vrchol premennej ako správu aproximáciu svojej marginálnej pravdepodobnosti.

Správy si posielajú iba susedné vrcholy a môžu byť dvoch druhov:

- správa od faktoru f_a veličine X_i : správa je vektor

$$m_{a \rightarrow i} = \begin{pmatrix} m_{a \rightarrow i}(h_1) \\ m_{a \rightarrow i}(h_2) \\ \vdots \\ m_{a \rightarrow i}(h_K) \end{pmatrix},$$

teda každá zložka odpovedá jedenj z možných hodnôt X_i

- správa od veličiny X_i faktoru f : opäť sa pošle vektor, ktoré tentokrát značíme

$$n_{i \rightarrow a} = \begin{pmatrix} n_{i \rightarrow a}(h_1) \\ n_{i \rightarrow a}(h_2) \\ \vdots \\ n_{i \rightarrow a}(h_K) \end{pmatrix}$$

Jedným krokom algoritmu rozumieme výmenu správ medzi všetkými susednými vrcholmi. Označíme $m_{a \rightarrow i}^{(t)}(h)$ a $n_{i \rightarrow a}^{(t)}(h)$ správy v kroku t . Na začiatku sa každá správa inicializuje nasledovne: $m_{a \rightarrow i}^{(0)}(h) = \sum_{\substack{x \in \sigma_a \\ x_i = h}} f_a(x)$ a $n_{i \rightarrow a}^{(0)}(h) = 1$.

Správy v kroku $t + 1$ potom vypočítame podľa nasledujúcich pravidiel:

$$n_{i \rightarrow a}^{(t+1)}(h) = \prod_{c \in N(i) \setminus a} m_{c \rightarrow i}^{(t)}(h)$$

a

$$m_{a \rightarrow i}^{(t+1)}(h) = \sum_{\substack{x \in \sigma_a \\ x_i = h}} f_a(x) \prod_{j \in N(a) \setminus i} n_{j \rightarrow a}^{(t+1)}(x_j) \quad (1.4)$$

kde $N(i)$ je množina susedov vrcholu i a výrazom σ_a rozumieme množinu všetkých možných hodnôt, ktoré môže nadobúdať vektor argumentov funkcie f_a .

Podľa týchto rovníc môžeme teda správu od premennej faktoru interpretovať ako prehlásenie o tom, čo jej o jej hodnote hovoria ostatné faktory a správu od faktoru premennej ako prehlásenie o relatívnych pravdepodobnostiach, že veličina je v jej rôznych stavoch, založených na faktore.

Povieme, správa $n_{i \rightarrow a}^{(t)}(h)$ je v kroku t *stabilná*, ak $n_{i \rightarrow a}^{(s)}(h) = n_{i \rightarrow a}^{(t)}(h)$ pre každé $s > t$. Rovnako pre $m_{a \rightarrow i}^{(t)}(h)$. Algoritmus sa *stabilizuje* v kroku t , ak sú v kroku t stabilné všetky správy.

Algoritmus v ideálnom prípade prebieha dovtedy, kým sa nestabilizuje, teda skončí vtedy, keď si vrcholy posielajú už iba rovnaké správy.

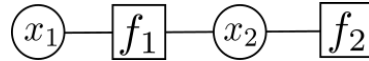
Zavedieme ešte pojem *belief* (viera), značíme $b_i(h)$, pre vrchol veličiny X_i . Re-rezentuje hodnotu, ktorou BP algoritmus aproximuje presnú marginálnu pravdepodobnosť. Počíta sa po ukončení algoritmu, teda správy uvažujeme v poslednom kroku. Vyrátame ju z nasledujúcej rovnice:

$$b_i(h) \propto \prod_{a \in N(i)} m_{a \rightarrow i}(h), \quad (1.5)$$

kde \propto je symbol značiaci, že všetky belief sa musia normalizovať, keďže chceme, aby aproximovali pravdepodobnosť.

Ukážeme si priebeh algoritmu na nasledujúcom príklade:

Príklad. Majme faktorizáciu $p(x_1, x_2) = f_1(x_1, x_2)f_2(x_2)$, náhodné veličiny nadobúdajú hodnoty v $\{0, 1\}$.



Obrázok 1.3: Graf príkladu

Na tomto grafe bude prebiehať algoritmus následovne:

0. Inicializácia:

$$\begin{aligned} n_{1 \rightarrow 1}^{(0)} &= \begin{pmatrix} 1 \\ 1 \end{pmatrix}, n_{2 \rightarrow 1}^{(0)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, n_{2 \rightarrow 2}^{(0)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, m_{1 \rightarrow 1}^{(0)} = \begin{pmatrix} f_1(0,0) + f_1(0,1) \\ f_1(1,0) + f_1(1,1) \end{pmatrix}, \\ m_{1 \rightarrow 2}^{(0)} &= \begin{pmatrix} f_1(0,0) + f_1(1,0) \\ f_1(0,1) + f_1(1,1) \end{pmatrix}, m_{2 \rightarrow 2}^{(0)} = \begin{pmatrix} f_2(0) \\ f_2(1) \end{pmatrix} \end{aligned}$$

Tu si všimneme, že sa správy vo vektore $n_{1 \rightarrow 1}$ a správy vo vektore $m_{2 \rightarrow 2}$ v tomto kroku stabilizujú – ani vrchol x_1 , ani vrchol f_2 totiž nemajú žiadne susedné vrcholy.

1.

$$\begin{aligned} n_{2 \rightarrow 1}^{(1)} &= m_{2 \rightarrow 2}^{(0)} = \begin{pmatrix} f_2(0) \\ f_2(1) \end{pmatrix}, n_{2 \rightarrow 2}^{(1)} = m_{1 \rightarrow 2}^{(0)} = \begin{pmatrix} f_1(0,0) + f_1(1,0) \\ f_1(0,1) + f_1(1,1) \end{pmatrix}, \\ m_{1 \rightarrow 1}^{(1)} &= \begin{pmatrix} f_1(0,0)n_{2 \rightarrow 1}^{(1)}(0) + f_1(0,1)n_{2 \rightarrow 1}^{(1)}(1) \\ f_1(1,0)n_{2 \rightarrow 1}^{(1)}(0) + f_1(1,1)n_{2 \rightarrow 1}^{(1)}(1) \end{pmatrix} \\ &= \begin{pmatrix} f_1(0,0)f_2(0) + f_1(0,1)f_2(1) \\ f_1(1,0)f_2(0) + f_1(1,1)f_2(1) \end{pmatrix}, \\ m_{1 \rightarrow 2}^{(1)} &= \begin{pmatrix} f_1(0,0)n_{1 \rightarrow 1}^{(1)}(0) + f_1(1,0)n_{1 \rightarrow 1}^{(1)}(1) \\ f_1(0,1)n_{1 \rightarrow 1}^{(1)}(0) + f_1(1,1)n_{1 \rightarrow 1}^{(1)}(1) \end{pmatrix} = \begin{pmatrix} f_1(0,0) + f_1(1,0) \\ f_1(0,1) + f_1(1,1) \end{pmatrix} \end{aligned}$$

Keďže vektor správ $m_{2 \rightarrow 2}$ je už stabilný, v tomto kroku sa stabilizuje aj $n_{2 \rightarrow 1}$, čím sa stabilizuje $m_{1 \rightarrow 1}$. Ďalej vieme už zo stabilizácie $n_{1 \rightarrow 1}$, že sú stabilizované aj správy v $m_{1 \rightarrow 2}$.

2.

$$n_{2 \rightarrow 2}^{(2)} = m_{1 \rightarrow 2}^{(1)} = \begin{pmatrix} f_1(0,0) + f_1(1,0) \\ f_1(0,1) + f_1(1,1) \end{pmatrix}$$

Zo stabilizácie $m_{1 \rightarrow 2}$ máme stabilizáciu $n_{2 \rightarrow 2}$. Teda v algoritmus sa stabilizoval už v prvom kroku.

Ešte vypočítame belief:

$$b_1 \propto \begin{pmatrix} m_{1 \rightarrow 1}(0) \\ m_{1 \rightarrow 1}(1) \end{pmatrix} = \begin{pmatrix} f_1(0,0)f_2(0) + f_1(0,1)f_2(1) \\ f_1(1,0)f_2(0) + f_1(1,1)f_2(1) \end{pmatrix} = \begin{pmatrix} p(X_1 = 0) \\ p(X_1 = 1) \end{pmatrix},$$

$$b_1 \propto \begin{pmatrix} m_{1 \rightarrow 2}(0)m_{2 \rightarrow 2}(0) \\ m_{1 \rightarrow 2}(1)m_{2 \rightarrow 2}(1) \end{pmatrix} = \begin{pmatrix} (f_1(0,0) + f_1(1,0))f_2(0) \\ (f_1(0,1) + f_1(1,1))f_2(1) \end{pmatrix} = \begin{pmatrix} p(X_2 = 0) \\ p(X_2 = 1) \end{pmatrix}$$

Vo všeobecnosti nemusíme dostať presnú marginálnu pravdepodobnosť, ani nemusíme dostať stabilizáciu, ale ak je graf strom, tak sa to podarí vždy, čo dokážeme v nasledujúcej vete.

Veta 2. *Nech X_1, X_2, \dots, X_n sú diskkrétne náhodné veličiny, ktoré nadobúdajú hodnoty v množine H a $p(x)$ ich združená pravdepodobnosť taká, že jej faktorgraf je strom. Označme d maximálnu vzdialenosť vrcholov v grafe. Potom sa algoritmus belief propagation stabilizuje v kroku $d - 1$ a $b_i(h) \propto p(X_i = h)$ pre každé $i = 1, 2, \dots, n$ a každé $h \in H$.*

Dôkaz. Označme G faktorgraf pravdepodobnosti $p(x)$. Najprv ukážeme stabilizáciu. Všimnime si, že správa $m_{a \rightarrow i}(h)$ závisí len na správach posielaných v podstrome $M_{a,i}$ vzniknutom z pôvodného grafu odobratím všetkých vrcholov, z ktorých vedie cesta do a cez i , okrem vrcholov a a i . To je tým, že graf G je strom a vrchol a dostáva správy len od vrcholov z podstromu $M_{a,i}$. Podobne, $n_{i \rightarrow a}(h)$ závisí len na správach posielaných v podstrome $N_{i,a}$, ktorý vznikne z pôvodného grafu odobratím všetkých vrcholov, z ktorých vedie cesta do i cez a , okrem vrcholov a a i . Ukážeme, že každá správa $m_{a \rightarrow i}(h)$, resp. $n_{i \rightarrow a}(h)$ sa stabilizuje po $l - 1$ krokoch, kde l je maximálna dĺžka cesty v $M_{a,i}$ končiaca vrcholom i , v $N_{i,a}$ končiaca vrcholom a . Túto cestu označíme C . Vezmime vrcholy a a i a ukážeme indukciou podľa l , že správy $m_{a \rightarrow i}^{(l-1)}(h)$ a $n_{i \rightarrow a}^{(l-1)}(h)$ sú stabilné. Najprv ukážeme stabilizáciu pre správu $m_{a \rightarrow i}^{(l-1)}(h)$:

- $l = 1$: To znamená, že a už nemá žiadnych ďalších susedov, teda $m_{a \rightarrow i}(h) = f_a(h)$ je konštantná.
- Teraz predpokladajme, že tvrdenie pre všetky správy také, že $M_{j,c}$, resp. $N_{c,j}$, má maximálnu dĺžku cesty menšiu ako l . Z definície:

$$m_{a \rightarrow i}^{(l-1)}(h) = \sum_{\substack{x \in \sigma_a \\ x_i = h}} f_a(x) \prod_{j \in N(a) \setminus i} n_{j \rightarrow a}^{(l-1)}(x_j).$$

Strom $N_{j,a}, j \in N(a) \setminus i$ je podstromom v $M_{a,i}$, takže v ňom musí byť dĺžka najdlhšej cesty C' končiaca vrcholom a menšia ako l . Ak by nebola, tak z toho, že C vždy končí vrcholmi a a i , by sme pridaním vrcholu i k ceste C' dostali cestu dĺžky väčšej ako l v $M_{a,i}$. Takže môžeme použiť indukčný predpoklad na $n_{j \rightarrow a}^{(l-1)}(x_j)$, čiže táto správa sa stabilizovala v kroku menšom ako $l - 1$. Teda $m_{a \rightarrow i}^{(l-1)}(h)$ má v súčine len stabilné správy a je stabilná.

Podobne ukážeme pre správu $n_{i \rightarrow a}^{(l-1)}(h)$:

- $l = 1$: To znamená, že i už nemá žiadnych ďalších susedov, teda v každom kroku je $n_{i \rightarrow a}^{(l)}(h) = 1$.
- Rovnako predpokladajme, že pre všetky správy také, že $M_{j,c}$, resp. $N_{c,j}$, má maximálnu dĺžku cesty končiacu c , resp. j , menšiu ako l , platí tvrdenie. Z definície:

$$n_{i \rightarrow a}^{(l-1)}(h) = \prod_{c \in N(i) \setminus a} m_{c \rightarrow i}^{(l-2)}(h).$$

Z rovnakých dôvodov ako vyššie platí, že v $M_{c,i}$ je maximálna dĺžka cesty končiaca vrcholom i menšia ako l . Podľa indukčného predpokladu sa $m_{c \rightarrow i}(h)$ stabilizovala v kroku menšom $l - 1$, čiže $m_{c \rightarrow i}^{(l-2)}(h)$ je stabilná. Teda opäť má $n_{i \rightarrow a}^{(l-1)}(h)$ v súčine len stabilné správy a je stabilná.

Špeciálne to platí pre vrchol na konci najdlhšej cesty v G . Cesta C musí mať dĺžku d , teda správa z tohto vrcholu sa stabilizuje po $d - 1$ krokoch, ostatné správy sa stabilizovali buď po menej ako $d - 1$ alebo po $d - 1$ krokoch, teda celkovo sa algoritmus stabilizoval po $d - 1$ krokoch.

Teraz ukážeme druhú časť tvrdenia; všetky správy budeme uvažovať v kroku $d - 1$. Urobíme dôkaz indukciou podľa počtu vrcholov stromu, ktorý označíme k . Keďže v tomto prípade faktorgraf určuje faktorizáciu pravdepodobnosti, graf vždy bude mať aspoň dva vrcholy.

- $k = 2$: Ak má strom dva vrcholy, potom máme jeden vrchol premennej X_1 a jeden vrchol faktoru f_1 . Vypočítame belief: $b_1(h) \propto m_{1 \rightarrow 1}(h) = f_1(h) \propto p(X_1 = h)$.
- *indukčný krok*: Vezmeme graf G s počtom vrchlov $k > 2$ a budeme predpokladať, že tvrdenie vety platí pre grafy s počtom vrcholom menším ako k . Najprv si prepíšeme rovnicu (1.3) združenej pravdepodobnosti tak, že zoskupíme faktory, ktoré majú spoločné argumenty. Označme $F_{a,i}$ množinu f_c faktorov takých, že vo faktorgrafe existuje cesta z vrcholu faktoru f_a do vrcholu faktoru f_c , ktorá nevedie cez vrchol premennej X_i .

Zafixujeme i . Pre $a, c \in N(i), a \neq c$ sú množiny $F_{a,i}$ a $F_{c,i}$ disjunktné. Ak by neboli, tak pre $d \in F_{a,i}$ máme cesty (i, a, \dots, d) a (i, c, \dots, d) , ich spojením dostaneme cestu $(i, a, \dots, d, \dots, c, i)$, čo je spor s tým, že faktorgraf je strom. To nám umožňuje prepísať $p(X_i = h)$ ako

$$p(X_i = h) \propto \sum_{\substack{x \in H^n \\ x_i = h}} \prod_{a \in N(i)} \prod_{f_c \in F_{a,i}} f_c(x).$$

Takisto pre každé $j \neq i$ platí, že všetky jeho susedné faktory (teda faktory, ktoré majú x_j za argument) sú v práve jednej z množín $F_{a,i}$, kde $a \in N(i)$, čo sa ukáže podobne. Preto môžeme ďalej upraviť:

$$p(X_i = h) \propto \prod_{a \in N(i)} \sum_{\substack{x \in \sigma_{F_{a,i}} \\ x_i = h}} \prod_{f_c \in F_{a,i}} f_c(x).$$

Tu rozširujeme značenie σ_a na σ_F pre nejakú množinu faktorov F , kde budeme počítat cez možné hodnoty všetkých vektorov zložených z argumentov všetkých faktorov v F .

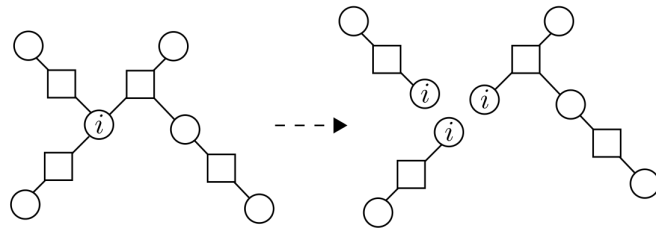
Rozlíšime dva prípady:

1. *Vrchol i nie je list*: Z faktorgrafu odoberieme vrchol i , čím sa G rozpadne na stromy. Ku každému z týchto novo vzniknutých stromov opäť pridáme kópiu vrcholu i k faktoru, s ktorým predtým susedil, čím dostaneme stromy s počtom vrcholom menším ako k (viď obrázok), na ktoré môžeme použiť indukčný predpoklad. Vezmime jeden z týchto stromov, označme ho G' , a jediný susedný vrchol vrcholu i v ňom a $b'_i(h)$ belief vypočítaný z tohto stromu. Všimnime si, že $G' = M_{a,i}$ z pozorovania na začiatku dôkazu, teda $m_{a \rightarrow i}(h)$ je rovnaká v G' rovnaká ako v G . V tomto strome určuje belief pravdepodobnosť $p'(X_i = h)$, ktorú vypočítame len z faktorov v $M_{a,i}$:

$$b'_i(h) \propto \prod_{a \in N(i)} m_{a \rightarrow i}(h) = m_{a \rightarrow i}(h) \propto p'(X_i = h) \propto \sum_{\substack{x \in \sigma_{F_{a,i}} \\ x_i = h}} \prod_{f_c \in F_{a,i}} f_c(x).$$

Takže ak sa vrátíme k pôvodnému stromu s k vrcholmi, dostaneme:

$$b_i(h) \propto \prod_{a \in N(i)} m_{a \rightarrow i}(h) \propto \prod_{a \in N(i)} \sum_{\substack{x \in \sigma_{F_{a,i}} \\ x_i = h}} \prod_{f_c \in F_{a,i}} f_c(x) \propto p(X_i = h).$$



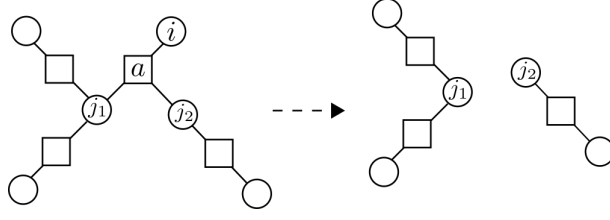
Obrázok 1.4: Príklad použitia IP na vrchol i , ktorý nie je list

2. *Vrchol i je list*: Potom, keďže má i len jeden susedný faktor f_a :

$$b_i(h) \propto m_{a \rightarrow i}(h) \tag{1.6}$$

$$\propto \sum_{\substack{x \in \sigma_a \\ x_i = h}} f_a(x) \prod_{j \in N(a) \setminus i} n_{j \rightarrow a}(x_j) \tag{1.7}$$

$$\propto \sum_{\substack{x \in \sigma_a \\ x_i = h}} f_a(x) \prod_{j \in N(a) \setminus i} \prod_{c \in N(j) \setminus a} m_{c \rightarrow j}(x_j) \tag{1.8}$$



Obrázok 1.5: Príklad použitia IP na vrchol i , ktorý je list

Teraz použijeme indukčný predpoklad na stromy G' , ktoré vzniknú odobratím vrcholov i a a . S použitím značenia $F_j = \bigcup_{c \in N(j) \setminus a} F_{c,j}$ prepíšeme rovnicu združenej pravdepodobnosti:

$$p(X_i = h) \propto \sum_{\substack{x \in H^n \\ x_i = h}} \prod_{a=1}^M f_a(x) \propto \sum_{\substack{x' \in \sigma_{F_j} \\ x'_j = x_j}} \prod_{f_c \in F_j} f_c(x').$$

V grafoch G' teda platí pre vrcholy j , ktoré boli v pôvodnom grafe susedmi a :

$$\prod_{c \in N(j) \setminus a} m_{c \rightarrow j}(x_j) \propto b'_j(x_j) \propto \sum_{\substack{x' \in \sigma_{F_j} \\ x'_j = x_j}} \prod_{f_c \in F_j} f_c(x')$$

Dosadíme indukčný predpoklad do rovnice (1.8):

$$\sum_{\substack{x \in \sigma_a \\ x_i = h}} f_a(x) \prod_{j \in N(a) \setminus i} \prod_{c \in N(j) \setminus a} m_{c \rightarrow j}(x_j) \propto \sum_{\substack{x \in \sigma_a \\ x_i = h}} f_a(x) \prod_{j \in N(a) \setminus i} \sum_{\substack{x' \in \sigma_{F_j} \\ x'_j = x_j}} \prod_{f_c \in F_j} f_c(x')$$

Označíme $F = \bigcup_{j \in N(a) \setminus i} F_j$, roznásobíme $\prod_{j \in N(a) \setminus i} \sum_{\substack{x' \in \sigma_{F_j} \\ x'_j = x_j}} \prod_{f_c \in F_j} f_c(x')$ a dostame:

neme:

$$\sum_{\substack{x \in \sigma_a \\ x_i = h}} f_a(x) \prod_{j \in N(a) \setminus i} \sum_{\substack{x' \in \sigma_{F_j} \\ x'_j = x_j}} \prod_{f_c \in F_j} f_c(x') \propto \sum_{\substack{x \in \sigma_a \\ x_i = h}} f_a(x) \sum_{\substack{\tilde{x} \in \sigma_F \\ \tilde{x}_j = x_j}} \prod_{f_c \in F} f_c(\tilde{x}) \quad (1.9)$$

$$\propto \sum_{\substack{x \in \sigma_a \\ x_i = h}} \sum_{\substack{\tilde{x} \in \sigma_F \\ \tilde{x}_j = x_j}} \prod_{f_c \in F} f_c(\tilde{x}_j) f_a(x) \quad (1.10)$$

$$\propto \sum_{\substack{y \in \sigma_{F \cup f_a} \\ y_i = h}} \prod_{f_d \in F \cup f_a} f_d(y) \quad (1.11)$$

$$\propto p(X_i = h), \quad (1.12)$$

čo sme chceli dokázať.

□

1.5 Diskrétna konvolúcia a Fourierova transformácia

V tejto časti predstavíme pojmy, ktoré nám umožnia zrýchliť algoritmus belief propagation. Tento spôsob zrýchlenia prevezmeme z článku [3]. Keďže obrazy budeme brať diskkrétne po pixeloch, uvedieme iba diskkrétne varianty.

Definícia (Diskrétna Fourierova transformácia). *Nech f je funkcia definovaná na množine $\{-N, \dots, N\}$. Diskrétnou Fourierovou transformáciou funkcie f rozumíme transformáciu postupnosti $\{f(k)\}_{k=-N}^N$ na postupnosť $\{X_k\}_{k=-N}^N$, kde $X_k = \sum_{n=-N}^N f(n)e^{-\frac{2\pi i}{2N+1}kn}$. Výrazom $DFT(f)$ potom myslíme vektor (X_{-N}, \dots, X_N) .*

Z $DFT(f)$ dostaneme späť f inverznou Fourierovou transformáciou. Z [6] vieme, že $f(k) = \frac{1}{n} \sum_{n=-N}^N X_n \cdot e^{\frac{2\pi i}{2N+1}kn}$.

Definícia (Diskrétna konvolúcia). *Pre reálnu funkciu f definovanú na množine $\{-N, \dots, N\}$ a reálnu funkciu g definovanú na množine $\{0, \dots, N\}$ definujeme diskkrétne konvolúciu f a g nasledujúcim spôsobom:*

$$(f * g)(n) = \sum_{m=0}^N f(n-m)g(m)$$

pre $n \in \{-N, \dots, N\}$.

Ďalej budeme potrebovať diskrétne verziu konvolučného teorému. Uvedieme si ho v nasledujúcej podobe s dôkazom:

Veta 3 (Konvolučný teorém, diskrétne verzia). *Nech f je funkcia definovaná na množine $\{-N, \dots, N\}$ a g funkcia definovaná na $\{0, \dots, N\}$. Označme g' funkciu takú, že $g'(n) = g(n)$ pre $n \in \{0, \dots, N\}$ a $g'(n) = 0$ inak. Potom*

$$DFT(f * g) = DFT(f) \cdot DFT(g'),$$

kde na pravej strane násobíme vektory po zložkách.

Dôkaz. Ukážeme, že k -tá zložka na ľavej strane sa rovná k -tej zložke na pravej strane pre $k \in \{-N, \dots, N\}$. Teda chceme:

$$\sum_{n=-N}^N \sum_{m=0}^N f(n-m)g(m)e^{-\frac{2\pi i}{2N+1}kn} = \sum_{j=-N}^N f(j)e^{-\frac{2\pi i}{2N+1}kj} \sum_{m=-N}^N g'(m)e^{-\frac{2\pi i}{2N+1}km}.$$

Budeme ďalej upravovať pravú stranu:

$$\begin{aligned} & \sum_{j=-N}^N f(j)e^{-\frac{2\pi i}{2N+1}kj} \sum_{m=-N}^N g'(m)e^{-\frac{2\pi i}{2N+1}km} \\ &= \sum_{j=-N}^N f(j)e^{-\frac{2\pi i}{2N+1}kj} \sum_{m=0}^N g'(m)e^{-\frac{2\pi i}{2N+1}km} \\ &= \sum_{j=-N}^N f(j)e^{-\frac{2\pi i}{2N+1}kj} \sum_{m=0}^N g(m)e^{-\frac{2\pi i}{2N+1}km} \\ &= \sum_{j=-N}^N \sum_{m=0}^N f(j)g(m)e^{-\frac{2\pi i}{2N+1}k(m+j)}. \end{aligned}$$

Keďže $e^{-\frac{2\pi i}{2N+1}k}$ je primitívna $(2N+1)$ -tá odmocnina z jednej, platí $e^{-\frac{2\pi i}{2N+1}k(m+j)} = e^{-\frac{2\pi i}{2N+1}k(m+j \bmod (2N+1))}$, kde čísla modulo $2N+1$ reprezentujeme ako $-N, \dots, N$. Urobíme substitúciu $n = m + j \bmod (2N+1)$, takže $j = n - m \bmod (2N+1)$. Keďže pre pevné m je $j = n - m \bmod (2N+1)$ bijekcia, touto substitúciou preusporiadame poslednú sumu:

$$\begin{aligned} & \sum_{j=-N}^N \sum_{m=0}^N f(j)g(m)e^{-\frac{2\pi i}{2N+1}k(m+j)} \\ &= \sum_{n=-N}^N \sum_{m=0}^N f(n-m)g(m)e^{-\frac{2\pi i}{2N+1}kn} \\ &= (DFT(f * g))_k, \end{aligned}$$

čo sme chceli dokázať. □

Tento teorém nám umožní zrýchliť algoritmus belief propagation v špeciálnom prípade. V ďalšej kapitole ukážeme, že práve tento prípad odpovedá aplikácii algoritmu na obraz. Majme faktor f_a taký, že má len dve susedné premenné X_i, X_j a faktor f_a je funkciou rozdielu $x_i - x_j$. Veličiny majú hodnoty v $H = \{0, \dots, N\}$. Ukážeme, že je možné zapísať $m_{a \rightarrow i}(h_1)$ ako diskretnú konvolúciu. Označíme g funkciou $n_{j \rightarrow a}$, dosadíme:

$$m_{a \rightarrow i}(h_1) = \sum_{h_2 \in H} f(h_1 - h_2)g(h_2) = (f_a * g)(h_1). \quad (1.13)$$

Platí teda, že vektor správ $m_{a \rightarrow i} = (m_{a \rightarrow i}(0), \dots, m_{a \rightarrow i}(N))$ je rovný vektoru $((f_a * n)(0), \dots, (f_a * n)(N))$. Vďaka konvolučnému teorému môžeme namiesto konvolúcie počítať Fourierovu transformáciu funkcií f_a a g . Urobíme to v nasledujúcich krokoch:

1. Urobíme diskretnú Fourierovu transformáciu funkcie f_a . Funkcia f_a dostáva hodnoty v množine $\{-N, \dots, N\}$, takže dostaneme vektor $DFT(f)$ s $2N+1$ prvkami.
2. Zdefinujeme g' ako v znení konvolučného teorému a urobíme diskretnú Fourierovu transformáciu funkcie g' . Dostaneme vektor $DFT(g')$.
3. Vynásobíme bodovo $DFT(f_a) \cdot DFT(g')$. Konvolučný teorém nám hovorí, že dostaneme $DFT(f_a * g)$.
4. Inverznou Fourierovou transformáciou získame

$$(f_a * g) = ((f_a * g)(-N), \dots, (f_a * g)(0), \dots, (f_a * g)(N)).$$

Ak vezmeme posledných $N+1$ prvkov $(f_a * g)$, dostaneme $m_{a \rightarrow i}$.

2. Aplikácia na obrázky

V tejto kapitole predstavíme pojem obraz a vysvetlíme, ako použiť na odstránenie šumu grafické modely.

2.1 Obraz

Matematicky sa dá na obraz pozeráť viacerými spôsobmi, napríklad v spracovaní obrazu sa obraz vníma ako funkcia. Pre naše potreby sa budeme na obraz rozmerov $m \times n$ pozeráť ako na pole rozmerov $m \times n$ s hodnotami v množine $\{0 \dots 255\}$. Množina $\{0 \dots 255\}$ reprezentuje množinu intenzít farby, kde 0 je čierna farba a 255 je biela farba. Teda pre zjednodušenie sa budeme zaoberať iba čiernobielymi obrazmi, pri viacfarebnom obraze sa ako obor hodnôt pridajú odtiene RGB.

Nás zaujímajú obrázky so šumom. Šum môžeme popísať ako nežiadajúci vedľajší efekt, ktorý vznikol pri zachytávaní obrazu a zakrýva pôvodné informácie.

2.2 Aplikácia grafických modelov

Teraz navrhujeme podľa [1] spôsob, ako odstrániť šum v obraze pomocou grafických modelov. Pôvodný obraz, ktorý hľadáme, označíme \mathbf{T} (*true image*) a zašumený, čiže ten, ktorý sme dostali, \mathbf{N} (*noisy image*). Obe reprezentujú funkciu ako v definícii vyššie. Našou úlohou je nájsť \mathbf{T} taký, že bude maximalizovať $P(\mathbf{T}|\mathbf{N})$. Tento vzťah prepíšeme pomocou Bayesovej vety nasledovne:

$$P(\mathbf{T}|\mathbf{N}) = \frac{P(\mathbf{N}|\mathbf{T})P(\mathbf{T})}{P(\mathbf{N})}, \quad (2.1)$$

kde $P(\mathbf{N})$ je pravdepodobnosť toho, že dostaneme zašumený obraz. Táto pravdepodobnosť nezávisí na pôvodnom obraze, a preto sa ňou nemusíme pri maximalizovaní zaoberať. Teda sme náš problém previedli na nasledujúci: Nájsť obraz \mathbf{T} taký, že bude maximalizovať $P(\mathbf{N}|\mathbf{T})P(\mathbf{T})$. $P(\mathbf{N}|\mathbf{T})$ zas určuje podobnosť obrazu \mathbf{T} s obrazom N .

$P(\mathbf{T})$ určuje, ako veľmi je obraz podobný skutočnému obrazu - v ňom majú susedné pixely podobné intenzity, teda to vyžadujeme aj od hľadaného \mathbf{T} .

2.2.1 Reprezentácia faktorgrafom

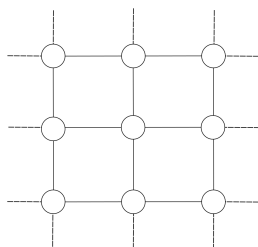
Vzťah $P(\mathbf{N}|\mathbf{T})P(\mathbf{T})$ budeme reprezentovať faktorgrafom. Na to potrebujeme faktorizovať $P(\mathbf{N}|\mathbf{T})P(\mathbf{T})$. V tomto prípade sú náhodné veličiny jednotlivé pixely obrazov \mathbf{T} a \mathbf{N} , teda veličina $T_{i,j}$ odpovedá pixelu na pozícii i,j . Množina hodnôt je množina intenzít, teda $\{0, \dots, 255\}$, podobne pre $N_{i,j}$.

Budeme predpokladať ako v [1], že sa $P(\mathbf{T})$ faktorizuje ako súčin funkcií susedných pixelov a $P(\mathbf{N}|\mathbf{T})$ ako súčin pixelov v obrazoch N a T na rovnakých pozíciách. Takže pracujeme s nasledujúcou faktorizáciou:

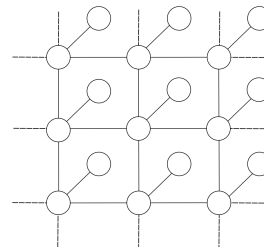
$$P(\mathbf{N}|\mathbf{T})P(\mathbf{T}) = \prod_{\substack{i=1,2,\dots,m-1 \\ j=1,2,\dots,n-1}} f(t_{i,j}, t_{i,j+1}) \cdot f(t_{i,j}, t_{i+1,j}) \prod_{\substack{i=1,2,\dots,m \\ j=1,2,\dots,n}} g(t_{i,j}, n_{i,j}). \quad (2.2)$$

Teraz už môžeme zostrojiť graf.

Začneme s tým, že každý pixel pôvodného obrazu zakreslíme ako vrchol v grafe a hrana bude vždy medzi susednými pixelmi. Potom k nim dokreslíme vrcholy reprezentujúce pixely v zašumenom obraze a tie spojíme s vrcholmi pixelov na rovnakej pozícii v hľadanom obraze.

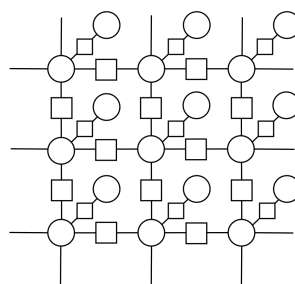


Obrázok 2.1: Vrcholy $t_{i,j}$



Obrázok 2.2: Vrcholy $t_{i,j}$ a $n_{i,j}$

Nakoniec doplníme vrcholy reprezentujúce faktory:



Budeme teda počítať s predpokladom, že vieme faktorizovať pravdepodobnosť ako súčin funkcií susedných pixelov. To ale nie je možné vždy, ako ukážeme v nasledujúcom príklade. Hoci tento príklad neodpovedá realite, ukazuje, že faktorizácia len modeluje skutočnosť a má svoje medze.

Príklad. Uvažujme „obraz“ o rozmeroch 2×2 pixely, pixely ktorého nadobúdajú len hodnoty 0 alebo 1. Vyzerá nasledovne:

x_{11}	x_{12}
x_{21}	x_{22}

Predstavme si, že pre tento obraz platí: ak je $x_{11} + x_{12} + x_{21} + x_{22} \equiv 0 \pmod{2}$, potom je $P(x_{11}, x_{12}, x_{21}, x_{22}) = \frac{1}{8}$, inak je rovná 0. Ukážeme, že potom nie je možné faktorizovať $P(x_{11}, x_{12}, x_{21}, x_{22})$ ako súčin

$$f_1(x_{11}, x_{12}) f_2(x_{11}, x_{21}) f_3(x_{21}, x_{22}) f_4(x_{22}, x_{12})$$

pre nejaké funkcie f_1, f_2, f_3, f_4

Dôkaz. Ukážeme sporom. Nech

$$P(x_{11}, x_{12}, x_{21}, x_{22}) = f_1(x_{11}, x_{12}) f_2(x_{11}, x_{21}) f_3(x_{21}, x_{22}) f_4(x_{22}, x_{12})$$

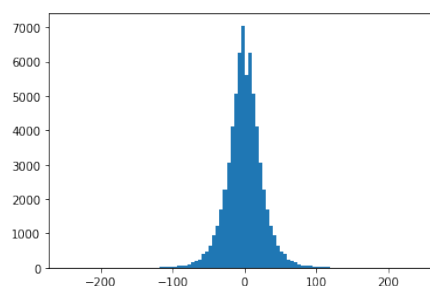
pre nejaké funkcie f_1, f_2, f_3, f_4 . Vezmime stav, kde $x_{11} = 0, x_{12} = 1, x_{21} = 1, x_{22} = 0$. Keďže súčet pixelov je 0, pravdepodobnosť tohto stavu je 0. Teda tiež súčin $f_1(x_{11}, x_{12})f_2(x_{11}, x_{21})f_3(x_{21}, x_{22}) = 0$. To znamená, že aspoň jeden z činiteľov je nulový. Pre každý z činiteľov ukážeme, že ak by bol nulový, tak potom by mal nulovú pravdepodobnosť stav, ktorý ju zo zadania mať nemá.

- $f_1(0,1) = 0$: stav $x_{11} = 0, x_{12} = 1, x_{21} = 1, x_{22} = 1$ má nulovú pravdepodobnosť.
- $f_2(0,1) = 0$: stav $x_{11} = 0, x_{12} = 1, x_{21} = 1, x_{22} = 1$ má nulovú pravdepodobnosť.
- $f_3(1,0) = 0$: stav $x_{11} = 1, x_{12} = 1, x_{21} = 1, x_{22} = 0$ má nulovú pravdepodobnosť.
- $f_4(1,0) = 0$: stav $x_{11} = 1, x_{12} = 1, x_{21} = 1, x_{22} = 0$ má nulovú pravdepodobnosť.

Vylúčili sme každú možnosť, teda pravdepodobnosť nie je možné v tomto prípade faktorizovať. □

2.2.2 Voľba faktorizujúcich funkcií

Funkcie f a g budeme voľiť tak, aby sa čo najviac priblížili reálnemu rozdeleniu susedných pixelov v obrazoch a zároveň sa s nimi dobre pracovalo. Zo [1] vieme, že susedné pixely majú väčšinou podobnú intenzitu, okrem miest, kde sa nachádza na obraze hrana. Teda ak zakreslíme hodnoty rozdielov susedných pixelov do histogramu, dostaneme graf, ktorý má v nule výrazné maximum a smerom k väčším rozdielom rýchlo klesá.



Obrázok 2.3: Príklad obrazu a jeho histogramu

Nasledujúce pozorovanie nám hovorí, že pri voľbe log-konkávnych funkcií môžeme s obrazom, ktorý dostaneme maximalizáciou, robiť operácie ako je normalizácia jasnosti alebo spriemerovať dve veľmi pravdepodobné obrazy a dostať opäť veľmi pravdepodobný obraz. Umožní nám to skutočnosť, že množina obrazov maximalizujúcich pravdepodobnosť je konvexná.

Pozorovanie 4. Ak sú funkcie $P(\mathbf{N}|\mathbf{T})$ a $P(\mathbf{T})$ log-konkávne, potom je log-konkávna aj funkcia $P(\mathbf{T}|\mathbf{N})$ a množina T_{max} prvkov maximalizujúcich $P(\mathbf{T}|\mathbf{N})$ je konvexná.

Dôkaz. Najprv ukážeme, že súčin $P(\mathbf{N}|\mathbf{T})P(\mathbf{T})$ je log-konkávna funkcia. Máme $\log P(\mathbf{N}|\mathbf{T})P(\mathbf{T}) = \log P(\mathbf{N}|\mathbf{T}) + \log P(\mathbf{T})$, súčet konkávnych funkcií je konkávna funkcia, teda $P(\mathbf{N}|\mathbf{T})P(\mathbf{T})$ je log-konkávna.

Máme $\log P(\mathbf{T}|\mathbf{N}) = \log \frac{P(\mathbf{N}|\mathbf{T})P(\mathbf{T})}{P(\mathbf{N})} = \log P(\mathbf{N}|\mathbf{T})P(\mathbf{T}) - \log P(\mathbf{N})$. Funkcia $\log P(\mathbf{N})$ je konštanta, teda z rovnakého dôvodu ako vyššie je aj $P(\mathbf{T}|\mathbf{N})$ log-konkávna.

Označme $P(\mathbf{T}|\mathbf{N})$ ako $f(T)$. Vezmeme $T_1, T_2 \in T_{max}$ a chceme ukázať, že pre $\lambda \in [0,1]$ aj $\lambda T_1 + (1 - \lambda)T_2 \in T_{max}$. Vezmeme $T_3 \notin T_{max}$. Keďže $f(T) \in [0,1]$, máme $f(\lambda T_1 + (1 - \lambda)T_2) \geq \log f(\lambda T_1 + (1 - \lambda)T_2)$. Z konkávnosti:

$$\log f(\lambda T_1 + (1 - \lambda)T_2) \geq \lambda \log f(T_1) + (1 - \lambda) \log f(T_2).$$

Z $T_3 \notin T_{max}$:

$$\lambda \log f(T_1) + (1 - \lambda) \log f(T_2) > \lambda \log f(T_3) + (1 - \lambda) \log f(T_3) = \log f(T_3).$$

Dohromady: $\log f(\lambda T_1 + (1 - \lambda)T_2) > \log f(T_3) \Rightarrow f(\lambda T_1 + (1 - \lambda)T_2) > f(T_3)$. Teda $\lambda T_1 + (1 - \lambda)T_2 \in T_{max}$ a T_{max} je konvexná. □

Uvažovali sme nasledujúce voľby funkcií:

1) $f = e^{-|x-y|}, g = e^{-(x-y)^2}$: Dosadením do 2.2 získame nasledujúci optimalizačný problém:

$$\text{maximalizuj } \prod_{i,j} e^{-|t_{i,j}-t_{i,j+1}|} \cdot e^{-|t_{i,j}-t_{i+1,j}|} \cdot e^{-(t_{i,j}-n_{i,j})^2}$$

za podmienky $0 \leq t_{i,j} \leq 255$.

Ak urobíme logaritmus účelovej funkcie, tak dostaneme

$$\log \prod_{i,j} e^{-|t_{i,j}-t_{i,j+1}|} \cdot e^{-|t_{i,j}-t_{i+1,j}|} \cdot e^{-(t_{i,j}-n_{i,j})^2} \quad (2.3)$$

$$= - \sum_{i,j} |t_{i,j} - t_{i,j+1}| + |t_{i,j} - t_{i+1,j}| + (t_{i,j} - n_{i,j})^2 \quad (2.4)$$

Funkcia $\sum_{i,j} |t_{i,j} - t_{i,j+1}| + |t_{i,j} - t_{i+1,j}| + (t_{i,j} - n_{i,j})^2$ je konvexná, keďže je súčtom konvexných funkcií, teda ju môžeme minimalizovať pomocou knižnice CVXPY. Do tejto rovnice pridáme ešte parameter $\lambda > 0$, ktorý bude udávať, akú váhu dávame podobnosti so zašumeným obrazom, čiže minimalizujeme

$$\sum_{i,j} |t_{i,j} - t_{i,j+1}| + |t_{i,j} - t_{i+1,j}| + \lambda \sum_{i,j} (t_{i,j} - n_{i,j})^2$$

pre $\lambda > 0$.

2) $f = g = e^{-(x-y)^2}$: V tomto prípade môžeme rovnicu (2.3) riešiť metódou najmenších štvorcov.

Označíme t vektor, ktorého prvky sú pixely pôvodného obrazu, zapísané po riadkoch, teda $t = (t_{1,1}, t_{1,2}, \dots, t_{1,m}, \dots, t_{n,1}, t_{n,2}, \dots, t_{n,m})$. Podobne označíme

n vektor obrazu so šumom. Potom máme pre vhodne zvolené matice F a G : $\|Ft\| = \sum_{i,j} (t_{i,j} - t_{i,j+1})^2 + (t_{i,j} - t_{i+1,j})^2$ a $\|Gt - n\| = \sum_{i,j} (t_{i,j} - n_{i,j})^2$. Teda chceme minimalizovať $\|Ft\| + \lambda\|Gt - n\|$ pre nejaký parameter $\lambda > 0$. Parameter λ rovnako ako v prvom prípade udáva váhu podobnosti so zašumeným obrazom. Tento výraz ešte rozpíšeme:

$$\|Ft\| + \lambda\|Gt - n\| = \left\| \begin{pmatrix} F \\ \sqrt{\lambda}G \end{pmatrix} t - \begin{pmatrix} 0 \\ \sqrt{\lambda}n \end{pmatrix} \right\|, \quad (2.5)$$

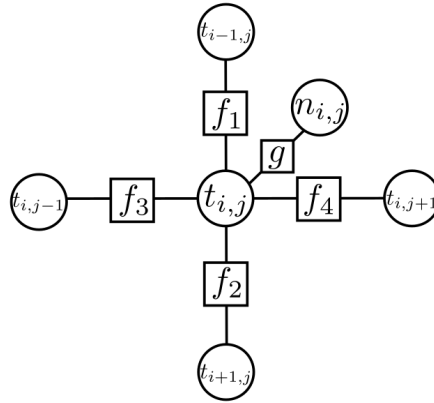
čo už môžeme riešiť ako problém najmenších štvorcov.

3) Studentovo t-rozdelenie: Toto rozdelenie sa často používa, pretože svojím tvarom najviac pripomína histogram rozdielov susedných pixelov. Táto funkcia avšak nie je log-konkávna, preto ju nebudeme používať ako v prvom prípade. Takže v tomto prípade budú naše funkcie:

$$f(x,y) = g(x,y) = \frac{\Gamma(\nu + 1)/2}{\sqrt{\pi\nu}\Gamma(\nu/2)} (1 + (x - y)^2/\nu)^{-(\nu+1)/2},$$

kde Γ je gamma-funkcia a $\nu > 0$ je parameter udávajúci stupeň voľnosti. Tento parameter ovlivňuje tvar grafu, teda ho chceme zvoliť tak, aby sa priblížil histogramu. My zvolíme napríklad $\nu = 10$, touto voľbou dostaneme graf podobný histogramu.

2.2.3 Belief propagation



Obrázok 2.4: Pixel a jemu prislúchajúce funkcie

Teraz na popísaný graf aplikujeme algoritmus belief propagation. Každému pixelu, ktorý nie je okrajový, prislúcha päť funkcií: f_1, f_2, f_3, f_4 pre susedné pixely a funkcia g pre pixel v zašumenom obraze. Funkcie f_i teda majú rovnaký predpis, líšia sa len vstupmi – vstupom je pixel v strede a príslušný jemu susedný pixel. Pre zjednodušenie budeme ďalej počítat len s vnútornými pixelmi, pri okrajových by sme uvažovali o jednu f_i menej. V našej situácii je množina hodnôt rovná množine $S = \{0, \dots, 255\}$, teda množine intenzít farby.

Pri správe z premennej do faktoru sa nedeje nič zaujímavé, jednoducho sa urobí súčin cez správy od všetkých faktorov, okrem toho prijímajúceho správu.

Správu z faktora do premennej si preberieme bližšie. Najprv si všimneme, že faktor g posiela premennej pre každé $s \in S$ stále rovnakú hodnotu $g(s, n_{i,j})$. To je spôsobené tým, že hodnota $n_{i,j}$ je fixná - pochádza zo zašumeného obrazu. Takže sa budeme zaoberať len správami od $f_i, i \in \{1,2,3,4\}$. Pre zjednodušenie označíme premennú, do ktorej ide správa, x a druhú premennú, ktorá je argumentom, y . Máme teda pre $s_1 \in S$:

$$m_{f_i \rightarrow x}(s_1) = \sum_{s_2 \in S} f_i(s_1, s_2) n_{y \rightarrow f_i}(s_2). \quad (2.6)$$

Ak zvolíme f_i ako funkciu rozdielu $s_1 - s_2$, môžeme použiť postup popísaný za konvolučným teorémom pre $f = f_i$ a $g = n_{y \rightarrow f_i}$, a počítat správu pomocou DFT.

Takisto poznamenajme, že v tomto prípade nie je graf strom, teda nemáme zaručenú presnú marginálnu pravdepodobnosť. Čiže sa ju budeme snažiť iba aproximovať, čo je zvyčajným postupom v strojovom učení. Náš postup je len teoretický, ale ako vidíme v článku [2], dáva dobré praktické výsledky.

3. Implementácia

V tejto kapitole popíšeme, ako sme implementovali uvedené algoritmy a ukážeme a porovnáme výsledky algoritmov na obrazoch. Všetky programy sú napísané v programovacom jazyku Python. Programy sme testovali primárne na 10 obrazoch z databázy pre spracovanie obrazov, s rozmermi strán od 100 do 162 pixelov. Obrazy sme implementovali ako dvojrozmerné polia a pracovali s nimi ako s maticami. Maticu pre zašumený obraz označme N a pre pôvodný obraz T .

3.1 Popis postupov

3.1.1 Konvexná optimalizácia

Na riešenie problému týmto spôsobom sme použili knižnicu CVXPY, ktorá umožňuje konvexnú optimalizáciu. Táto knižnica nám umožnila takmer priamočiaru implementáciu. Použili sme prvú možnosť voľby funkcií f a g z minulej kapitoly a popísaný postup.

Funkciu f sme implementovali súčet totálnu variáciu riadkov, resp. stĺpcov matice T , funkciu g sme implementovali ako Frobeniovu normu matíc $T - N$.

Avšak narazili sme na problémy s pamäťou – väčšie obrazy znamenajú viac premenných a tie už knižnica nezvládala.

Možným riešením by bolo rozložiť obraz na menšie časti, čím by sme ale spôsobili nezrovnalosti na okrajoch, pozdĺž ktorých sme rozdelili obraz. Takisto, ako ukážeme ďalej, výsledky záviseli od voľby parametra λ .

Táto metóda sa ukázala byť pomerne rýchla, priemerný čas spracovania obrazu bol približne 6.7 sekundy.

3.1.2 Najmenšie štvorce

V tom prípade sme použili postup popísaný pre druhú možnosť voľby funkcií. Keďže matice F a G sú riedke, využili sme knižnicu SciPy, ktorá má naimplementované operácie s riedkymi maticami.

Parameter λ sme volili v intervale $(0,2)$, teda tento parameter určoval pomer váh funkcií f a g . Najlepšie výsledky sme vypozerovali pre voľbu $\lambda \sim 1.4$.

Táto metóda bola najrýchlejšia, priemerný čas spracovania obrazu bol približne 0.2 sekundy.

3.1.3 Belief propagation

Pre porovnanie rýchlosti algoritmu sme implementovali oba verzie algoritmu – bez použitia DFT a s použitím DFT. V oboch prípadoch sme sa snažili znížiť počet operácií tým, že sme namiesto počítania správ prvok po prvku sme nahradili cyklus operáciami na maticiach.

Použili sme rovnaké funkcie ako pri konvexnej optimalizácii, opäť z toho dôvodu, že sme mohli namiesto exponenciály použiť logaritmus. Keďže tieto funkcie v našom prípade nadobúdajú len konečne mnoho hodnôt, pred počítaním správ

sme ich predpočítali. Správy od susedných funkcií f sme ukladali do štvorozmerného poľa M , kde súradnice určovali o ktorý zo susedných faktorov sa jedná, ktorej premennej prislúcha a hodnotu premennej. Ako sme spomenuli skôr, správy od funkcie g sú konštatné a nemusíme ich ďalej počítat.

Pri implementácii sme z praktických dôvodov použili o niečo odlišnú inicializáciu správ ako v popise algoritmu. Správy $n_{i \rightarrow a}^{(0)}$ zostali rovnaké, správy $m_{a \rightarrow i}^{(0)}$ sme inicializovali na hodnotu 1. Touto úpravou dostaneme rovnaké správy ako v popise, len o krok neskôr. Správy N od premennej faktoru sme v oboch prípadoch počítali takmer rovnako. Pripomeňme, že správa od faktoru f je súčin správ premennej od ostatných faktorov, resp. súčet, ak počítame v logaritmickej škále. Trik spočíval v správnom posunutí poľa M , tak, aby sme mohli vypočítat N ako súčet. V oboch prípadoch sme použili 20 krokov algoritmu, pozorovaním sme totiž zistili, že pridávaním krokov už nedochádzalo k viditeľným zmenám.

Z belief určíme hodnotu pixelu následovne: najprv normalizujeme každý vektor $b_{i,j} = (b_{i,j}(0), b_{i,j}(1), \dots, b_{i,j}(255))$, kde i, j sú indexy pixelu v obraze. Pixelu na pozícii i, j v rekonštruovanom obraze potom priradíme hodnotu skalárneho súčinu $b_{i,j} \cdot (0, 1, \dots, 255)$.

Verzia bez DFT

V tomto prípade sme M počítali ako bodový súčin N a F , kde F je matica možných hodnôt funkcie f . Tu sme sa museli vysporiadať takisto s tým, že hodnoty blízke nule boli často nahradzované nulou, ktorá sa rýchlo šírila. To sme vyriešili tak, že sme „resetovali“ vektor správy a nahradili ho správou s rovnomerným rozdelením. Takisto sme museli normalizovať správy, pretože rýchlo narastali a vznikali priveľké hodnoty.

V tomto prípade sme použili dve voľby funkcií, konkrétne voľby 1) a 3) popísané v sekcii 2.2.2. Keďže Studentovo rozdelenie nie je exponenciála, prevod na logaritmus bol pomalší. Táto metóda bola najpomalšia, priemerný čas spracovania obrazu použitím exponenciál bol približne 417.4 sekundy, študentovo rozdelenie zvýšilo čas na takmer polhodiny.

Verzia s DFT

Správy M sme počítali spôsobom popísaným na konci sekcie 1.4. Keďže Fourierova transformácia je v Pythone implementovaná na množine $\{0, \dots, 2N + 1\}$, museli zmeniť poradie prvkov vo vektore. Teda namiesto usporiadania ako v postupe na konci prvej kapitoly, t.j. $-255, \dots, 255$, sme použili usporiadanie $0, \dots, 255, -255, \dots, -1$. Je ekvivalentné tomu pôvodnému z cykličnosti primitívnej odmocniny z .

Konvolúcia nám avšak zvýšila pamäťovú zložitosť, preto sme museli znížiť rozmery testovaných obrazov. Aby sme porovnali časovú zložitosť, na rovnakú vzorku sme použili aj verziu bez DFT. Priemerný čas spracovania bez DFT bol 74.75 sekundy, s DFT sa znížil na 51.80 sekundy, celkovo teda došlo k približne 1.5-násobnému zrýchleniu.

3.2 Výsledky na obrazoch

V tejto sekcii ukážeme na príklade obrazov na konci kapitoly výsledky algoritmov.

- **konvexná optimalizácia:** V tomto prípade ukážeme aj to, ako parameter menil výsledok. Pri príliš nízka voľba λ sme prišli o hrany v obraze, príliš vysoká neodstránila šum, čo odpovedá očakávaniam – parameter totiž udáva váhu zašumenému obrazu. Experimentami sme zistili, že najlepšie výsledky dávala voľba $\lambda \approx 150$. Napriek tomu vidíme, že sa šum neodstránil úplne.
- **najmenšie štvorce:** Týmto spôsobom sme dostali najlepšie odstránenie šumu, na druhú stranu sme stratili iné vlastnosti obrazu, ako je kontrast a ostrosť.
- **BF bez DFT, exponenciály:** V tomto prípade sme dosiahli najlepšiu podobnosť pôvodnému obrazu. Vidíme, že sa šum neodstránil úplne, zachovali sa hlavne tmavšie pixely.
- **BF bez DFT, študentovo rozdelenie:** Ak výsledný obraz porovnáme s predchádzajúcou voľbou funkcií, pozorujeme, že je obraz menej ostrý. Menšia ostrosť avšak znamená lepšie odstránenie šumu.
- **BF s DFT:** Tu vidíme, že šum sa odstránil najmenej zo všetkých metód a takisto došlo k najväčšiemu rozmazaniu hrán oproti pôvodnému obrazu.

Dohromady sme každým spôsobom do istej miery dosiahli odstránenie šumu.

3.3 Porovnanie výsledkov

V tejto časti chceme porovnať kvalitu obrazov rekonštruovaných pomocou rôznych metód. Ich použitím sme často stratili vlastnosti pôvodného obrazu ako je jas a kontrast. Preto potrebujeme použiť iný spôsob porovnávanie ako je napr. Frobeniova norma rozdielu obrazov. Tento problém nemá jednoznačné riešenie, keďže

My použijeme spôsob uvedený v článku [7]. Označme x a y vektory, ktoré vzniknú usporiadaním pixelov v dvoch porovnávaných obrazoch. Berieme do úvahy tri vlastnosti obrazu – jas, kontrast a štruktúru. Využijeme tzv. **SSIM** (*structural similarity measurement*) *index*. Počítame podľa nasledujúceho vzorca:

$$\mathbf{SSIM}(x,y) = l(x,y) \cdot c(x,y) \cdot s(x,y), \quad (3.1)$$

kde $l(x,y)$ je funkcia porovnávajúca jas obrazov, $c(x,y)$ je funkcia porovnávajúca kontrast a $s(x,y)$ je funkcia porovnávajúca štruktúru. Sú vyskladané sa z priemerného jas a rozptylu rozdielu obrazu a jeho priemerného jas. Sú volené tak, aby **SSIM** mal nasledujúce vlastnosti:

- $\mathbf{SSIM}(x,y) = \mathbf{SSIM}(y,x)$
- $\mathbf{SSIM}(x,y) \leq 1$
- $\mathbf{SSIM}(x,y) = 1 \Leftrightarrow x = y$.

Teda čím je hodnota **SSIM** bližšie k 1, tým viac sa obraz podobá pôvodnému obrazu. Porovnávanie pomocou tohto indexu sme naimplementovali a každý rekonštruovaný obraz sme porovnali s jeho pôvodným obrazom bez šumu. V tabuľke nižšie ukážeme **SSIM** index obrazov, ktoré sme dostali odstránením šumu. V prvom stĺpci sa nachádzajú hodnoty pre obraz na konci kapitoly, v druhom priemer spracovaných obrazov.

	SSIM príklad	SSIM priemer
Obraz so šumom	0.9006	0.8869
CVXPY, $\lambda = 60$	0.8471	0.8718
CVXPY, $\lambda = 100$	0.9043	0.9258
CVXPY, $\lambda = 200$	0.9498	0.9504
CVXPY, $\lambda = 300$	0.9224	0.9150
Najmenšie štvorce, $\lambda = 1.4$	0.9543	0.9555
BP, exponenciály	0.9385	0.9227
BP, Studentovo rozdelenie	0.9470	0.9532

Tabuľka 3.1: Tabuľka porovnania SSIM indexov

Z týchto výsledkov vidíme, že našimi algoritmi sme dostali vždy lepší výsledok ako pre zašumený obraz. Takisto sa v podstate zhodujú s pozorovaniami v predchádzajúcej časti – kvalita obrazu odpovedá **SSIM** indexu. Najlepší výsledok sme dostali pre najmenšie štvorce a BP použitím Studentovho rozdelenia.



Obrázok 3.1: Pôvodný obraz



Obrázok 3.2: Zašumený obraz



Obrázok 3.3: $\lambda = 60$



Obrázok 3.4: $\lambda = 100$



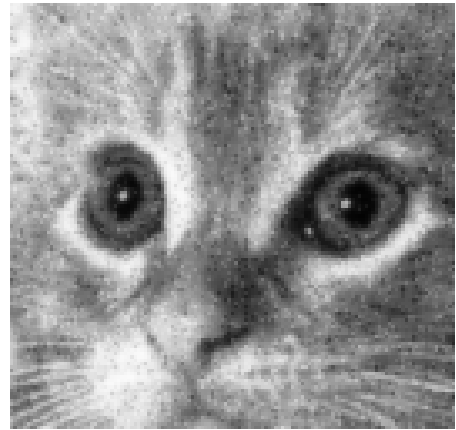
Obrázok 3.5: $\lambda = 200$



Obrázok 3.6: $\lambda = 300$



Obrázok 3.7: Zašumený obraz



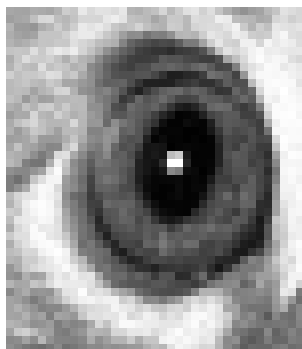
Obrázok 3.8: Najmenšie štvorce, $\lambda = 1.4$



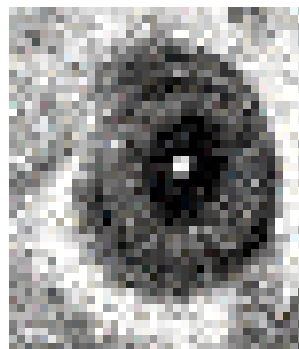
Obrázok 3.9: Belief propagation bez DFT, exponenciály



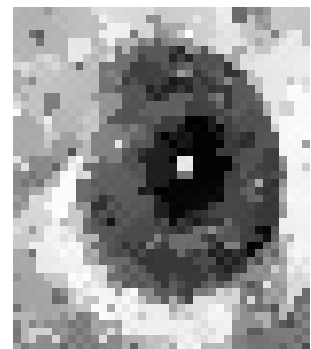
Obrázok 3.10: Belief propagation bez DFT, Studentovo rozdelenie



Obrázok 3.11: Pôvodný obraz



Obrázok 3.12: Obraz so šumom



Obrázok 3.13: BP s použitím DFT

Záver

V tejto práci sme popísali a implementovali tri spôsoby, ako pomocou grafického modelu rekonštruovať obraz. Každý z týchto spôsobov bol v odstránení šumu úspešný.

Hoci výsledky z hľadiska kvality výsledného obrazu boli porovnateľné, teoreticky komplikovanejšie spôsoby sa ukázali byť najmenej praktické. Najefektívnejšou metódou bola metóda najmenších štvorcov. Tá totiž zvládala aj obrazy väčších rozmerov a časovo pracovala omnoho rýchlejšie ako ostatné postupy.

Konvexnou optimalizáciou sme pre vhodne zvolený parameter dosiahli dobré výsledky v relatívne rýchlom čase, avšak nedokázali sme spracovať väčšie obrazy.

Pri rekonštrukcii cez belief propagation sme takisto boli obmedzení veľkosťou obrazu. Síce sme pri voľbe Studentovho rozdelenia ako faktorizačnej funkcie dosiahli podľa SSIM indexu najlepšie výsledky, čas spracovania sa pohyboval okolo pol hodiny. Použitím DFT sa algoritmus o niečo zrýchlil, ale boli sme schopní spracovať len obrazy ešte menších rozmerov ako pôvodne.

Zoznam použitej literatúry

- [1] Bernt Schiele. Probabilistic graphical models and their applications, Image processing. pages 1–30, 2016. <https://www.mpi-inf.mpg.de/fileadmin/inf/d2/GM/2016/gm-2016-1213-imageprocessing.pdf>.
- [2] Xiangyang Lan, Stefan Roth, Daniel Huttenlocher, and Michael J. Black. Efficient belief propagation with learned higher-order Markov random fields. In *Computer Vision – ECCV 2006*, pages 269–282, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [3] P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. *International Journal of Computer Vision*, 2006.
- [4] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*, pages 67–88. Cambridge University Press, 2004.
- [5] Jiří Matoušek and Jaroslav Nešetřil. *Kapitoly z diskrétní matematiky*, pages 95–104. Karolinum, 2010.
- [6] David Stanovský and Libor Barto. *Počítačová algebra*, pages 26–27. Matfyzpress, 2017.
- [7] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

Zoznam obrázkov

1.1	Príklad Bayesovej siete. Veličina A je závislá na veličinách B a C , C je závislá na veličinách B	5
1.2	Príklad Markovovho poľa. Veličina A je závislá na veličinách B a C , B je závislá na veličinách A, C a D , C je závislá na veličinách A a B a D je závislá na B	5
1.3	Graf príkladu	7
1.4	Príklad použitia IP na vrchol i , ktorý nie je list	10
1.5	Príklad použitia IP na vrchol i , ktorý je list	11
2.1	Vrcholy $t_{i,j}$	15
2.2	Vrcholy $t_{i,j}$ a $n_{i,j}$	15
2.3	Príklad obrazu a jeho histogramu	16
2.4	Pixel a jemu prislúchajúce funkcie	18
3.1	Pôvodný obraz	24
3.2	Zašumený obraz	24
3.3	$\lambda = 60$	24
3.4	$\lambda = 100$	24
3.5	$\lambda = 200$	24
3.6	$\lambda = 300$	24
3.7	Zašumený obraz	25
3.8	Najmenšie štvorce, $\lambda = 1.4$	25
3.9	Belief propagation bez DFT, exponenciály	25
3.10	Belief propagation bez DFT, Studentovo rozdelenie	25
3.11	Pôvodný obraz	25
3.12	Obraz so šumom	25
3.13	BP s použitím DFT	25