Principy tvorby stegosystémů

Practical 1

Cíl: Stegoobjekty by měly zachovávat statistické vlastnosti nosičů.

Problém: Neznáme žádný jednoduchý a zároveň přesný statistický model pro digitální fotografie.

Problém: Digitální fotografie vykazují mnoho složitých lokálních vztahů kvůli povaze senzorů a kvůli dalšímu zpracování obrazu. (Viz akvizice obrazu)

Řešení:

1) Stegosystém, který zachovává nějaký zjednodušený statistický model nosiče.
2) Stegosystém, který napodobuje nějaký přirozený proces zpravidla šum.

   Problém: Šum vzniká ještě předtím, než vůbec dojde ke konverzi zachyceného signálu na digitální signál, ~~signál potom dále~~ obraz potom prochází demosaicováním korekcí barev a filtry. Na závěr může ještě projít ztrátovou kompresí.

3) Stegosystém odolný proti stávajícím útokům (známým)
4) ~~Minimalizovat počet změn při vkládání~~ Minimalizovat dopad změn ~~způsobených~~ vkládáním.

(1) 7.11 Příklad stegosystému zachovávajícího model nosiče

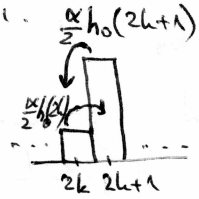
- Modelujeme kvantizované DCT koeficienty v obrázku JPEG jako nezávislé náhodné veličiny se stejným rozdělením.
- Cíl: zachovat histogram

Algoritmus OutGuess (vylepšení Jsteg)

1) Zaznamenáme histogram nosiče
2) Na základě stegoklíče zvolíme posloupnost DCT koeficientů do kterých budeme vkládat. Koeficienty s hodnotou 0 a 1 vynecháme. (Vkládání do 0 zanechává viditelné stopy)
3) V první fázi vkládáme do zvolených DCT koeficientů pomocí LSB embedding. (Tím pozměníme histogram)
4) V druhé fázi obnovíme původní histogram tím, že koeficienty, do kterých jsme nevkládali využijeme ke korekci histogramu.

Do jak velkého podílu α nenulových koeficientů budeme vkládat?
Musíme si nechat prostor na korekci.

- Co se děje při vkládání:



$\frac{\alpha}{2}$ koeficientů se přelije do vedlejšího sloupce.

Čili změna v histogramu v rámci k-tého páru je $\left|\frac{\alpha}{2}h_0(2k+1) - \frac{\alpha}{2}h_0(2k)\right|$

Menší sloupec vzroste a větší sloupec pohlesne.

- Korekce: musíme „přesunout" nevynulité koeficienty z menšího sloupce do většího abychom napravili změnu způsobenou vkládáním.

počet koeficientů použitelných pro korekci =

$$= (1-\alpha)\min\left(h_0(2k), h_0(2k+1)\right) \geq \frac{\alpha}{2}\left|h_0(2k+1) - h_0(2k)\right|$$

musí jich být dostatek, abychom mohli napravit změnu.

$$\frac{1}{\alpha} - 1 = \frac{1-\alpha}{\alpha} \geq \frac{\left|h_0(2k+1) - h_0(2k)\right|}{2\min\left(h_0(2k), h_0(2k+1)\right)}$$

$$\alpha \leq \frac{2\min\left(h_0(2k), h_0(2k+1)\right)}{\left|h_0(2k+1) - h_0(2k)\right| + 2\min\left(h_0(2k), h_0(2k+1)\right)} = \frac{2\min\left(h_0(2k), h_0(2k+1)\right)}{h_0(2k) + h_0(2k+1)}$$

$\forall k \in \mathbb{Z}$ (jmenovatel nenulový)

- Otázka je tedy, pro které k je výraz

můžu zkrátit $\frac{2\min\left(h_0(2k), h_0(2k+1)\right)}{h_0(2k) + h_0(2k+1)}$ minimální?

To lze také přeformulovat jako, pro které k je výraz $\frac{\min\left(h_0(2k), h_0(2k+1)\right)}{\max\left(h_0(2k), h_0(2k+1)\right)}$ ~~minimální~~ maximální (Vznikne odečtením 1 od předchozího.)

přeuspořádané hodnoty

- Ve fotografických se zpravidla jedná o pár $\{2k, 2k+1\} = \{-2, -1\}$

čili ~~Víme~~ máme α_{max}

čili máme $\alpha_{max} = \frac{2h_0(-2)}{h_0(-1) + h_0(-2)}$

Obvykle $\alpha_{max} = 0{,}2$ bpnc (bits per non-zero coefficient)

(3) ±1 embedding (v rastrovém obrázku) Practical 3

(7.3.1)

Problém s LSB embeddingu: Je asymetrické tj. při vkládání

sudé hodnoty vzrůstají

2k ⇄ 2k+1

liché hodnoty klesají

Řešení: Zprávu budeme nadále vkládat do LSB ovšem v případě, kdy se nejnižší bit v nosiči neshoduje s bitem zprávy provedeme nastavení bitu přičtením +1 nebo −1 k hodnotě, znaménko volíme náhodně. Čili $flip_{\pm 1}(x) = x + a$, kde $P(a=1) = 1/2 = P(a=-1)$.

- Výjimku nelze tvořit extremální hodnoty jako třeba 0 a 255, které lze pouze zvýšit nebo pouze snížit
- Nejsou známy žádné srovnatelně silné útoky na ±1 embedding, jako na LSB embedding.

(11.4) Útok: ±1 embedding se projevuje jako přidání náhodného šumu, tímto se vyhlazuje histogram.

- Hladkost histogramu můžeme měřit tím, že přejdeme k diskrétní fourierově transformaci histogramu H a poté spočítáme průměrnou hodnotu $|H|$. ~~(Bez +1 by vyšla 0 protože~~ (DFT je symetrické podle středu) Snaproximujeme polovinu spektra $\frac{\sum_{k=0}^{127} k\,|H(k)|}{\sum_{k=0}^{127} |H(k)|}$
- Problém: potřebujeme referenční množinu nosičů.
- ~~Problém~~ Výsledek: útok funguje pokud nosiče jsou bez šumu (např. díky softwarovému vyhlazení) s přirozeným šumem útok nerozpozná nosiče od stegoobrázků. množiny se příliš překrývají

Vylepšení: udělejme histogram ~~sousedních~~ sousedících pixelů:

$$h(i,j) = \#\{(x,y) \in P \mid x=i,\ y=j\}$$ (P je multimnožina sousedících párů pixelů)

Proveďme 2D-DFT → H

Spočítáme statistiku $\frac{\sum_{k,l=0}^{128} (k+l)\,|H(k,l)|}{\sum_{k,l=0}^{128} |H(k,l)|}$

Obrana proti histogramovému útoku na Jsteg.

Practical 4

- Zpráva bude opět uložena v LSB kvantizovaných DCT koeficientů.
- Vkládáme do nenulových AC koeficientů (tj. včetně 1) takto:
  - Má-li koeficient správný LSB, pak ho nenecháme.
  - V opačném případě dekrementujeme absolutní hodnotu koeficientu o 1.
- Problém: vložíme-li do koeficientu -1 nebo 1 hodnotu 0 pak výsledkem bude koeficient 0. Ale tento se při extrakci zprávy nečte. Tomuto fenoménu se říká „shrinkage" = „smrštění".
- Řešení: v případě smrštění musíme bit vložit znovu.
- Problém: Smrštění nastává pouze při vkládání bitu 0. Opětovné vkládání nulových bitů znamená, že celkově vkládáme více nul než jedniček.

Označme: $P(Z_i)$ pravděpodobnost, že v i-tém kroku vkládáme bit 0.
$P(S_i)$ pravděpodobnost, že v i-tém kroku dojde ke smrštění.

$$P(Z_i) = P(S_{i-1}) + \frac{1}{2}(1 - P(S_{i-1})), \quad P(Z_0) = 1/2.$$

$$P(S_i) = P(Z_i)\,\frac{h(-1)+h(1)}{\sum_{i\neq 0} h(i)}, \text{ kde } h \text{ je histogram AC koef.}$$

$$\Rightarrow \lim_{i\to\infty} P(Z_i) = \left(2 - \frac{h(-1)+h(1)}{\sum_{i\neq 0} h(i)}\right)^{-1} \left[P(Z_i) = \frac{1}{2}P(Z_{i-1})\,\frac{h(-1)+h(1)}{\sum_{i\neq 0} h(i)} + \frac{1}{2}\right]$$

Typicky $\dfrac{h(-1)+h(1)}{\sum_{i\neq 0} h(i)} \approx 0.5 \Rightarrow P(Z_i) \approx \frac{2}{3}$ a $1 - P(Z_i) \approx 1/3$

Vkládáme 2x více 0 než 1 !!!

Důsledek: u lichých koeficientů je pravděpodobnost změny 2x větší než u sudých.

Řešení: Předefinujeme pojem LSB pro záporná čísla:

$$LSB_{FS}(x) = \begin{cases} (1-x) \bmod 2 & \text{pro } x < 0 \\ x \bmod 2 & \text{jinak} \end{cases}$$

(Tuto metodu využívá algoritmus F5 společně s maticovým vkládáním)

Vložení 0 do 1 smršťuje a vložení bitu 1 do -1 smršťuje. Jelikož $h(1) \approx h(-1)$ dochází ke smrštění a při vkládání 0 a 1 se stejnou pravděpodobností.

Kapacita: $1 - \frac{1}{2}\,\frac{h(-1)+h(1)}{\sum_{i\neq 0} h(i)} \approx 0.75$ bpnc

bits per non-zero AC coefficient

(4) Minimalizace dopadu změn způsobených vkládáním | Practical 5

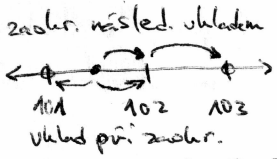
a) Vkládat do pixelů ve kterých bude změna nejméně znat.
Např. vkládat do oblastí s vysokou texturou (přesvícené oblasti najít nulovou texturu).

b) Schovat vkládání za zaokrouhlovací chybu.

- Vkládání při redukci hloubky barev.
  - Viděli jsme redukci hloubky s difúzí chyb a zaokrouhlováním pouze na barvy mající paritu odpovídající bitu zprávy.
  - V případě rastrového formátu lze využít např. při redukci z 48 bitů na 24 bitů. Vyjde-li např. odstín 101,5 můžeme zaokrouhlit na 101 nebo 102.

    Vyjde-li 101 ale my potřebujeme sudou hodnotu pak zaokrouhlíme na 102 nebo 100 ⇒ chyba zaokrouhlení je 1. (nejvýše)

    Kdybychom nejdříve zaokrouhlovali a pak vkládali může být chyba až 1,5: vyjde 101.5 zaokrouhlíme na 102 potom vkládáme bit 1 a ze 102 uděláme 103 ⇒ chyba 0.5+1.0 = 1.5.



- * Pro JPEG: Ve fázi kdy se kvantizují DCT koeficienty dochází též k zaokrouhlování.

c) Minimalizovat velikost změn.
Např. Vkládání s optimální volbou parity v paletovém formátu

d) Minimalizovat počet změn.

Jak je to s efektivitou metody (b)?

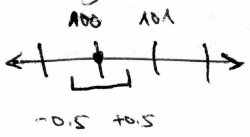
$$e = \frac{\#\text{ bitů vložených}}{\text{očekávaná distorze}} \qquad \text{distorze } d(x,y) = \sum_{i=1}^{n} (x_i - y_i)^2$$

Klasický LSB: $e = \frac{\alpha n}{\frac{\alpha}{2} n} = 2$ bity na jednotku distorze.

Distorze je ale ve skutečnosti větší, ~~vezm~~ uvažujeme-li nosič s nekonečnou hloubkou barev.
Na každý vložený bit připadá jedna distorze ~~$\frac{1}{2}$, ale navíc distorze~~ vyvolaná vkládáním ale navíc distorze vyvolaná samotnou kvantizací.

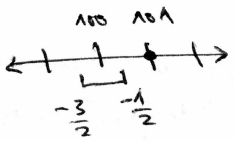
Klasický LSB embedding — redukce hloubky následovaný LSB embeddingem.

Distorze vyvolaná kvantizací bez flipu



$$\int_{-0.5}^{0.5} x^2\,dx = \left[\frac{x^3}{3}\right]_{-1/2}^{1/2} = \frac{1}{8\cdot 3} + \frac{1}{8\cdot 3} = \frac{1}{12}$$

Distorze vyvolaná kvantizací + flip



$$\int_{-\frac{3}{2}}^{-\frac{1}{2}} x^2\,dx = \left[\frac{x^3}{3}\right]_{-\frac{3}{2}}^{-\frac{1}{2}} = \frac{-1}{8\cdot 3} + \frac{27}{8\cdot 3} = \frac{13}{12}$$

celk. Distorze vyvolaná LSB embeddingem (na bit)

$$\frac{\frac{1}{12} + \frac{13}{12}}{2} = \frac{7}{12}$$ jednotek distorze na vkládaný bit

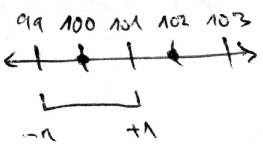
$$e = \frac{\alpha n}{\frac{7}{12}\alpha n} = \frac{12}{7} = 1.714$$ (bitů na jednotku distorze)

LSB embedding při redukci hloubky barev

Distorze na bit

Dejme tomu, že chci uložit 0.

$\frac{1}{3}$ jednotky distorze na vkládaný bit



$$\frac{1}{2}\int_{-1}^{1} x^2\,dx = \frac{1}{2}\left[\frac{x^3}{3}\right]_{-1}^{1} = \frac{1}{2}\left(\frac{1}{3} - \frac{-1}{3}\right) = \frac{1}{2}\cdot\frac{2}{3} = \frac{1}{3}$$

↑ může chápat: 1) délka intervalu je 2 ....
2) hustota pravděpodobnosti na intervalu $(-1,1)$ je $1/2$, (jinde je 0)

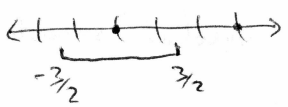
$$e = \frac{\alpha n}{\frac{1}{3}\alpha n} = 3$$ (bity na jednotku distorze)

Porovnání

$$\frac{3}{12/7} = \frac{7}{4} = 1.75$$

- Vkládání při zaokrouhlování zvyšuje efektivitu až o 75% oproti klasickému LSB embeddingu.
- Čili až o 75% víc bitů na jednotku distorze.

Ternární embedding



$$\frac{1}{3}\int_{-3/2}^{3/2} x^2\,dx = \frac{1}{9}\left(\frac{27}{8} + \frac{27}{8}\right) = \frac{3}{4}$$

$$e = \frac{\log_2 3^{\alpha n}}{\frac{3}{4}\alpha n} = \frac{4\log_2 3}{3} = 2.11$$