图像信息隐藏算法四

钮心忻、杨榆、雷敏
北京邮电大学信息安全中心
yangyu@bupt.edu.cn

图像水印算法介绍

- 自适应隐写算法
  - PVD
  - EA
  - S-UNIWARD

PVD 隐写

- PVD（pixel-value differencing）
- 基于图像中相邻像素对差异的隐写

PVD隐写算法

- 将载体图像分成许多不交迭的小块，每个小块由两个相邻像素组成
  - 组成方式有很多种，如逐行或逐列 zigzag 扫描
- 秘密信息被隐藏在每个小块的两个像素灰度的差值中

PVD隐写算法

- 像素灰度级差值记为 $d_i=p(i+1)-p(i)$
- 将像素的灰度级最大差值区间分割为K个子区间，各子区间宽度D_i是2的整数次幂，即为2^n
  - 例如分成6个子区间：[0,7], [8,15], [16,31], [32,63], [64,127], [128,255]， $D_1=2^3$, $D_2=2^3$…
- 根据差值 $|d_i|$ 隶属的子区间$[L_j, U_j]$，由区间宽度$D_j = U_j - L_j + 1$确定在该小块中嵌入的比特数n，即$n=\log_2 D_j$
  - 嵌入方法：调整 p(i+1) 和（或） p(i)，使得差值$|d_i'|$与n比特消息m_i对应。

$$|d_i'| = L_j + \left(m_{i,o}m_{i,1}\dots m_{i,n-1}\right)_D, m_{i,k} \in \{0,1\}, k = 0,1,\dots n-1$$

PVD隐写算法

- 根据PVD算法，设k值为6，即灰度级差值区间[0,255]被划分为6个子区间[0,7], [8,15], [16,31], [32,63], [64,127], [128,255]，则
  - 像素对（1，8）和（8，18）分别能隐藏多少比特秘密信息?
  - 若秘密信息为111和000，则隐藏像素值对变为?

PVD隐写算法

- 根据PVD算法，设k值为6，即灰度级差值区间[0,255]被划分为6个子区间[0,7], [8,15], [16,31], [32,63], [64,127], [128,255]，则
  - 解1：值对（1,8）差值|d|=7，落入子区间[0,7]，区间宽度为8，因此可以隐藏$\log_2(8)=3$比特信息。
  - 类似的，值对（8,18）差值|d|=10，落入子区间[8,15]，区间宽度为8，因此可以隐藏$\log_2(8)=3$比特信息。

PVD隐写算法

- 根据PVD算法，设k值为6，即灰度级差值区间[0,255]被划分为6个子区间[0,7], [8,15], [16,31], [32,63], [64,127], [128,255]，则
  - 解2：值对(1,8) 的差值$|d|$落在区间[0,7],隐藏秘密信息$(111)_B$后应调整差值为

$$L + |d'| = 0 + (111)_B = 7$$

即隐藏秘密信息后，值对大小不变，仍为(1,8)

  - 值对(8,18)的差值$|d|$落在区间[8,15],隐藏秘密信息$(000)_B$后应调整差值为

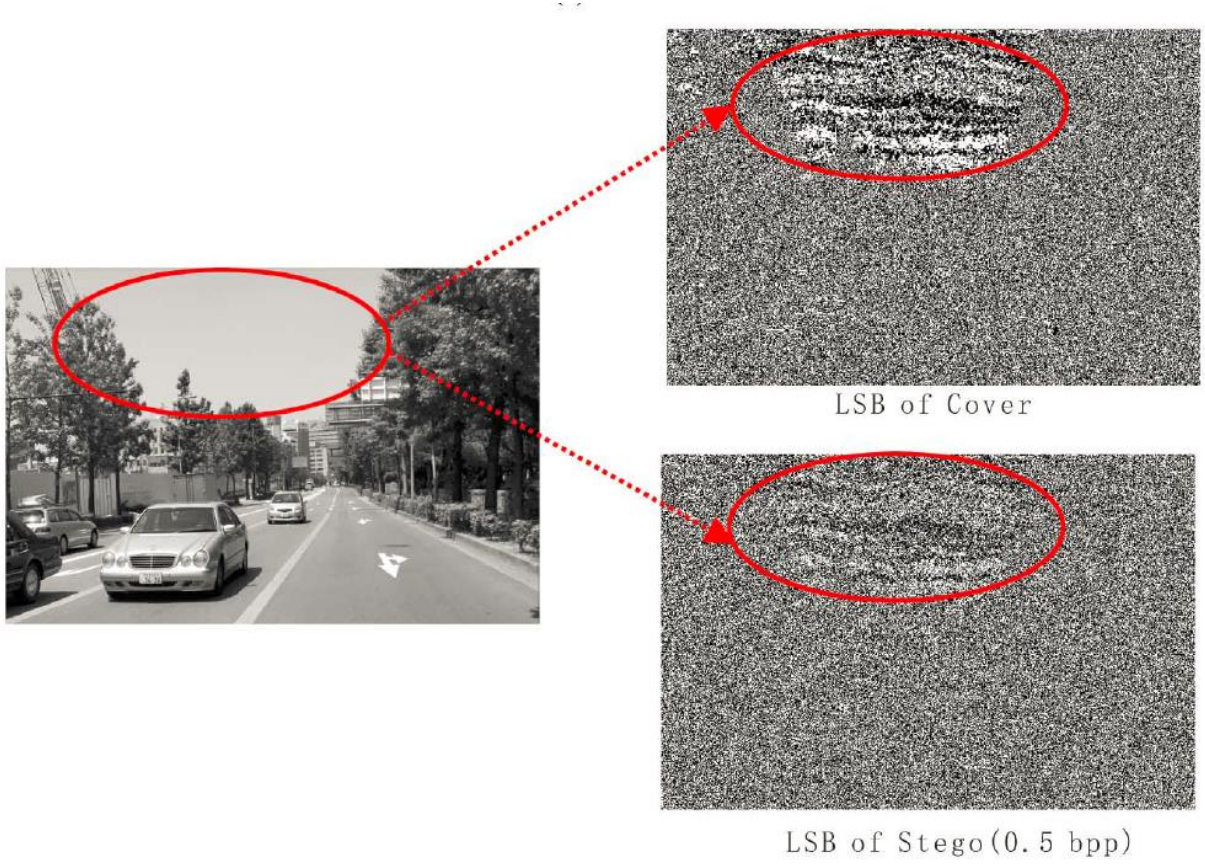
$$L + |d'| = 8 + (000)_B = 8$$

即隐藏秘密信息后，值对变为(8,16)

EA隐写算法

- EA（Edge Adaptive Image Steganography, 边缘自适应）
  - LSB和LSB Matching等算法忽略了图像内容自身纹理信息。
  - LSB Revisited和PVD利用了邻域像素对，但仍然没有充分结合载体自身特性。
  - 缺陷：即使只需要隐藏较少的秘密信息，图像中光滑、平坦的区域也可能被修改。

EA隐写算法



EA隐写算法

- EA（Edge Adaptive Image Steganography, 边缘自适应）
  - 对于一个$m \times n$图像，将其按$B_z \times B_z$大小分块，每个图像块根据秘钥随机择{0,90,180,270}度旋转。
  - 而后根据栅格（逐行）扫描方式排列为一维向量，并且，不重叠地将邻接像素两两组对，$V = \{(x_i, x_{i+1}) | i = 1,3,5, \cdots, m \times n - 1\}$(设$m \times n$为偶数)。

EA隐写算法

EA（Edge Adaptive Image Steganography,边缘自适应）

- 确定选取阈值T。已知$EU(t) = \{(x_i, x_{i+1}) || x_i - x_{i+1}| \geq t, \forall (x_i, x_{i+1}) \in V\}$，计算阈值T，使其满足:
- $T = \arg\max_t \{2 \times |EU(t)| \geq |M|\}$
- 其中，$|M|$为秘密消息长度。即寻找使得嵌入量不小于消息长度的最大值差作为阈值。T可选0,1,2，。。。。，31中任意数，其中，T选0时，退化为普通LSBMR（LSB-Matching Revisited）。

EA隐写算法

EA（Edge Adaptive Image Steganography,边缘自适应）

对集合$EU(T)=\{(x_i,x_{i+1})||x_i-x_{i+1}|\geq T,\forall(x_i,x_{i+1})\in V\}$的像素对进行隐写，隐写规则为：($r$随机取值为1或-1)

Case #1: $\mathrm{LSB}(x_i)=m_i$ & $f(x_i,x_{i+1})=m_{i+1}$

$$(x_i',x_{i+1}')=(x_i,x_{i+1});$$

Case #2: $\mathrm{LSB}(x_i)=m_i$ & $f(x_i,x_{i+1})\neq m_{i+1}$

$$(x_i',x_{i+1}')=(x_i,x_{i+1}+r);$$

Case #3: $\mathrm{LSB}(x_i)\neq m_i$ & $f(x_i-1,x_{i+1})=m_{i+1}$

$$(x_i',x_{i+1}')=(x_i-1,x_{i+1});$$

Case #4: $\mathrm{LSB}(x_i)\neq m_i$ & $f(x_i-1,x_{i+1})\neq m_{i+1}$

$$(x_i',x_{i+1}')=(x_i+1,x_{i+1})$$

$$f(a,b)=\mathrm{LSB}(\lfloor a/2\rfloor+b).$$

以$(x_i,x_{i+1})=(62,81),(m_i,m_{i+1})=(1,0)$,
$T=19$为例

$$\mathrm{LSB}(62)=0\neq m_i,\mathrm{LSB}\left(\left\lfloor\frac{(62-1)}{2}\right\rfloor+81\right)$$
$$=1\neq m_{i+1}.$$

$$(x_i',x_{i+1}')=(x_i+1,x_{i+1})=(63,81).$$

$$x_i=(x_{i,n-1},\cdots,x_{i,1},x_{i,0})$$
$$x_{i+1}=(x_{i+1,n-1},\cdots,x_{i+1,1},x_{i+1,0})$$
$$f(x_i,x_{i+1})=x_{i,1}\oplus x_{i+1,0}$$

EA隐写算法

- EA（Edge Adaptive Image Steganography,边缘自适应）
  - 隐藏后，邻域像素差值可能不满足要求，即 $|x_i' - x_{i+1}'| < T$，则需再次调整，调整规则为：

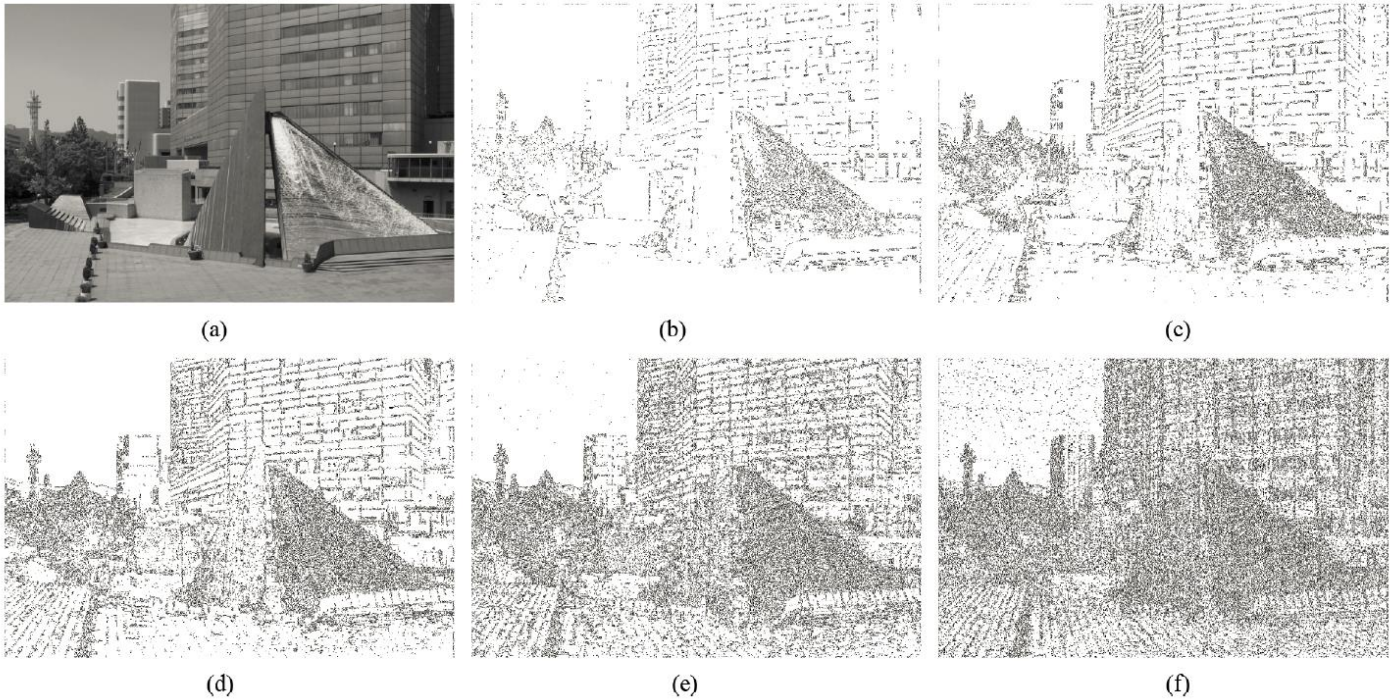
$$(x_i'', x_{i+1}'') = \arg\min_{(e_1,e_2)} \left\{ |e_1 - x_i| + |e_2 - x_{i+1}| \left| \begin{array}{c} e_1 = x_i' + 4k_1, \\ e_2 = x_{i+1}' + 2k_2, \\ |e_1 - e_2| \geq T, \\ 0 \leq e_1, e_2 \leq 255, \\ 0 \leq T \leq 31, \\ k_1, k_2 \in Z \end{array} \right. \right\}$$

前例中，$(x_i, x_{i+1}) = (62,81), (m_i, m_{i+1}) = (1,0), T = 19$，隐藏后，$(x_i', x_{i+1}') = (63,81), d = |63 - 81| = 18 \leq T$，需要调整。容易确定，$k_1 = 0, k_2 = 1$，则

$$x_i'' = x_i' + 4k_1 = 63 + 4 \times 0 = 63$$

$$x_{i+1}'' = x_{i+1}' + 2k_2 = 81 + 2 \times 1 = 83.$$

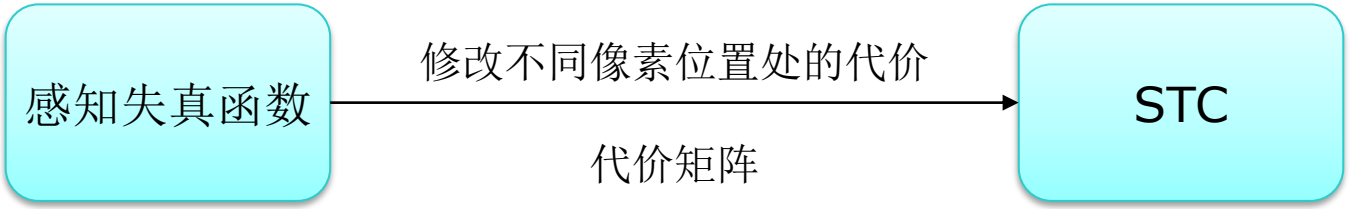
EA隐写算法



随着隐写的增加，更多的区域被自适应地选中用于隐写，且优先使用轮廓/边缘区域，平滑区域得到最大程度的维持。

UNIWARD隐写算法

- 基于STC的自适应隐写算法
  - 隐写算法的设计转变为失真函数的设计；
  - 失真函数明确了修改不同位置处像素所需付出的感知代价；
  - 基于代价，STC从所有码字中，挑选代价最小的码字作为隐写方案。

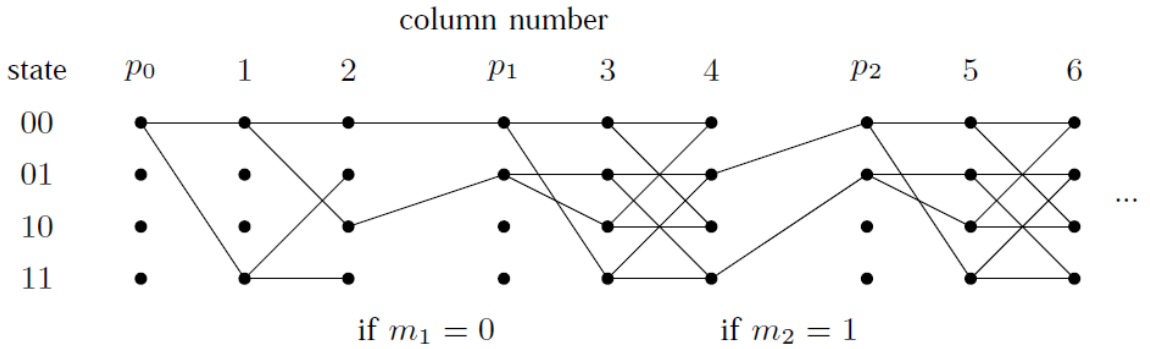


UNIWARD隐写算法

- 基于STC的自适应隐写算法
  - STC（SYNDROME-TRELLIS CODES）

$m_i = z_i \oplus z_{i+1} \oplus z_{i+2}, z_i \sim (m_i, m_{i+1})$

$$\hat{\mathbb{H}} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \qquad \mathbb{H} = \begin{pmatrix} 1 & 0 & & & & \\ 1 & 1 & 1 & 0 & & \\ & & 1 & 1 & 1 & 0 \\ & & & & 1 & 1 \\ & & & & & \ddots \end{pmatrix}$$



$m = (m_{M-1}, m_{M-2}, \cdots, m_1, m_0) = (0,1,1,\cdots)$

X → 隐藏 → Y → 提取

$z^{(X)} = LSB(X) = (z_{N-1}^{(X)}, z_{N-2}^{(X)}, \cdots, z_1^{(X)}, z_0^{(X)}) = (1,0,1,\cdots)$

$z^{(Y)} = LSB(Y) = (z_{N-1}^{(Y)}, z_{N-2}^{(Y)}, \cdots, z_1^{(Y)}, z_0^{(Y)}) = (1,1,1,\cdots)$

$m = Hz^{(Y)} = HLSB(Y) = (0,1,1,\cdots)$

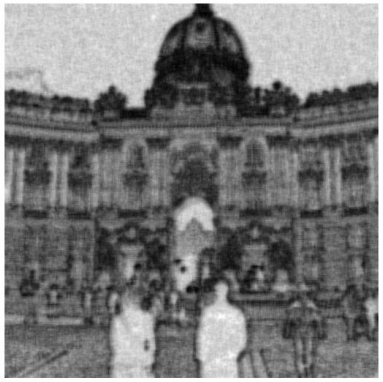
$$\mathcal{C}(\mathbf{m}) = \{\mathbf{z} \in \{0,1\}^n | \mathbb{H}\mathbf{z} = \mathbf{m}\}.$$

UNIWARD隐写算法

- UNIWARD
  - UNIversal Wavelet Relative Distortion，通用小波相对失真
  - S-UNIWARD（Spatial UNIWARD）
  - J-UNIWARD（Jpeg UNIWARD）
  - 通过小波分解描述特定方向上的纹理特性，引导STC的修改避开图像光滑区域。

UNIWARD隐写算法

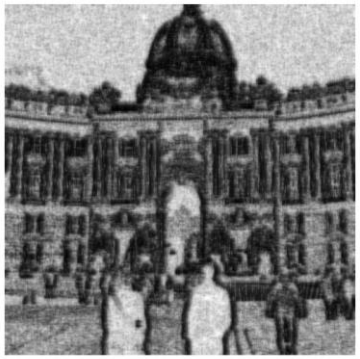
- S-UNIWARD

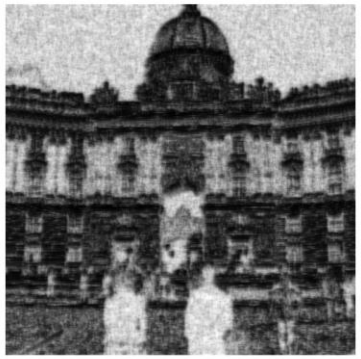


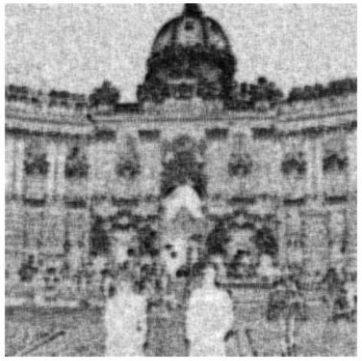
$$\mathcal{B} = \{\mathbf{K}^{(1)}, \mathbf{K}^{(2)}, \mathbf{K}^{(3)}\}. \quad \mathbf{W}^{(k)} = \mathbf{K}^{(k)} \star \mathbf{X},$$

$$\mathbf{K}^{(1)} = \mathbf{h} \cdot \mathbf{g}^{\mathrm{T}}, \ \mathbf{K}^{(2)} = \mathbf{g} \cdot \mathbf{h}^{\mathrm{T}}, \ \mathbf{K}^{(3)} = \mathbf{g} \cdot \mathbf{g}^{\mathrm{T}}.$$

$$D(\mathbf{X}, \mathbf{Y}) \triangleq \sum_{k=1}^{3} \sum_{u=1}^{n_1} \sum_{v=1}^{n_2} \frac{|W_{uv}^{(k)}(\mathbf{X}) - W_{uv}^{(k)}(\mathbf{Y})|}{\sigma + |W_{uv}^{(k)}(\mathbf{X})|},$$







隐写代价定义为载体和隐写对象之间的小波相对失真。

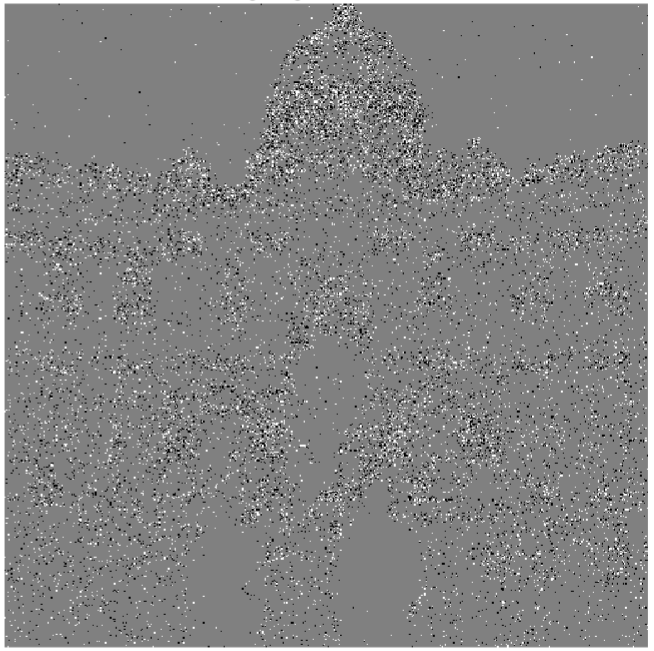
UNIWARD隐写算法

- S-UNIWARD

隐写位置显示，S-UNIWARD在纹理区域嵌入消息，图像的平滑区域总体得以维持不变。

cover

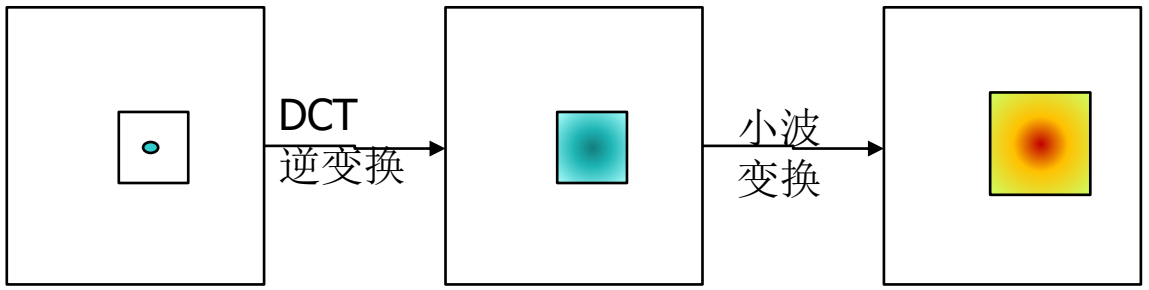
embedding changes: +1 = white, -1 = black



UNIWARD隐写算法

- 从空域到变换域的扩展

$$D(\mathbf{X},\mathbf{Y}) \triangleq D\left(J^{-1}(\mathbf{X}),J^{-1}(\mathbf{Y})\right) \qquad D(\mathbf{X},\mathbf{Y}) \triangleq \sum_{k=1}^{3}\sum_{u=1}^{n_1}\sum_{v=1}^{n_2}\frac{|W_{uv}^{(k)}(\mathbf{X})-W_{uv}^{(k)}(\mathbf{Y})|}{\sigma+|W_{uv}^{(k)}(\mathbf{X})|},$$

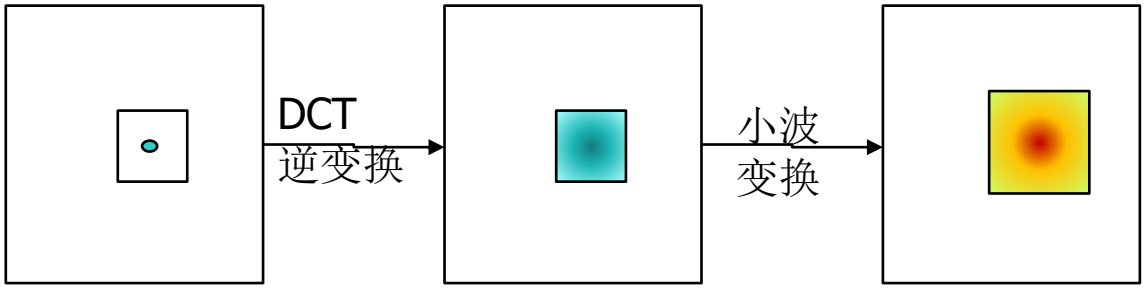


- 1个DCT系数发生变化，会影响到所在小块所有像素；
- 1个像素变换，会影响到整个区域小波系数；
- 可见，整个变化是非线性的。
- 运用加法原则简化分析。

$$D_{\mathrm{A}}(\mathbf{X},\mathbf{Y})=\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\rho_{ij}(\mathbf{X},Y_{ij})[X_{ij}\neq Y_{ij}], \quad \rho_{ij}(\mathbf{X},Y_{ij}) \triangleq D(\mathbf{X},\mathbf{X}_{\sim ij}Y_{ij}),$$

UNIWARD隐写算法

- 从空域到变换域的扩展



- 对于JPEG隐写，综合考虑计算复杂度和精度等要求，实际计算代价矩阵时，做了进一步简化。

$$\rho(X,Y_{ij}) \triangleq D\left(J^{-1}(X),J^{-1}(X_{\sim ij}Y_{ij})\right) \triangleq \sum_{k=1}^{3}\sum_{u=1}^{n_1}\sum_{v=1}^{n_2}\frac{\left|W_{u,v}^{(k)}(J^{-1}(X)) - W_{u,v}^{(k)}\left(J^{-1}(X_{\sim ij}Y_{ij})\right)\right|}{\sigma + \left|W_{u,v}^{(k)}(J^{-1}(X))\right|}$$

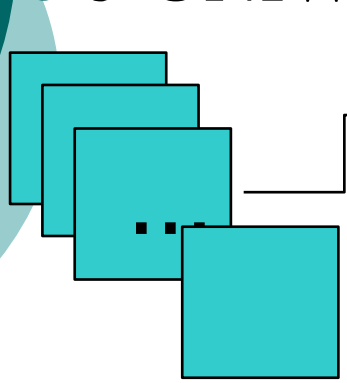
$$\approx \sum_{k=1}^{3}\sum_{u=1}^{n_1}\sum_{v=1}^{n_2}\frac{\left|W_{u,v}^{(k)}\left(J^{-1}\left((\Delta_{ij})\right)\right)\right|}{\sigma + \left|W_{u,v}^{(k)}(J^{-1}(X))\right|} \qquad (\Delta_{ij}) = 0_{\sim ij}1_{ij}$$

$$\sum_{k=1}^{3}\sum_{u=1}^{n_1}\sum_{v=1}^{n_2}\frac{\left|W_{u,v}^{(k)}\left(J^{-1}\left((\Delta_{ij})\right)\right)\right|}{\sigma+\left|W_{u,v}^{(k)}\left(J^{-1}(X)\right)\right|}$$

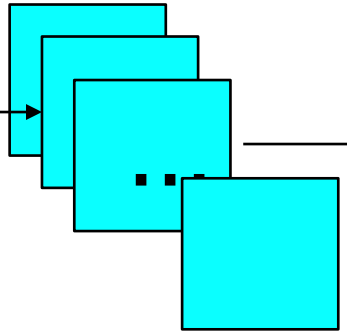
$$W_{u,v}^{(k)}\left(J^{-1}\left((\Delta_{ij})\right)\right),(\Delta_{ij})=0_{\sim ij}1_{ij}$$

UNIWARD隐写算法

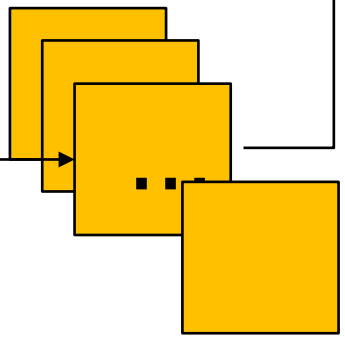
- J-UNIWARD



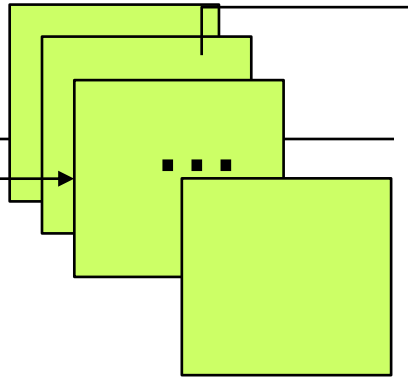
$$JB_{i,j}^{(k)}=\begin{cases}0;others\\1;8i+j=k\end{cases}$$



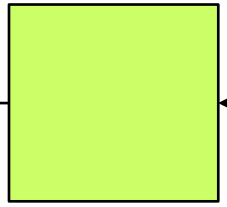
$$DB_{i,j}^{(k)}=\begin{cases}0;others\\\Delta_{i,j};8i+j=k\end{cases}$$



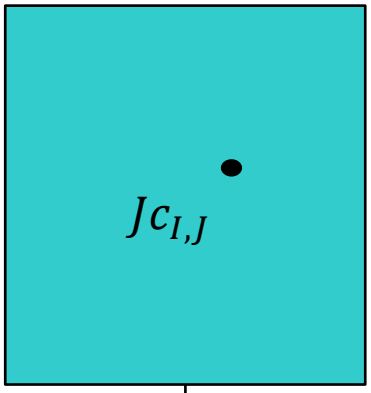
$$SB^{(k)}=DCT^{-1}\left(DB^{(k)}\right)$$

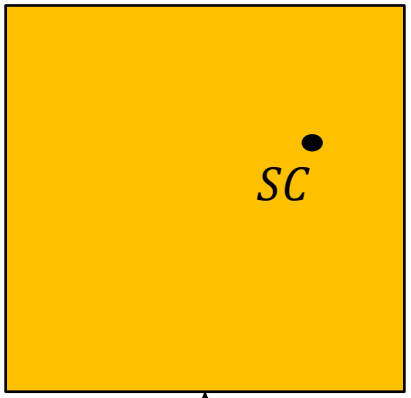


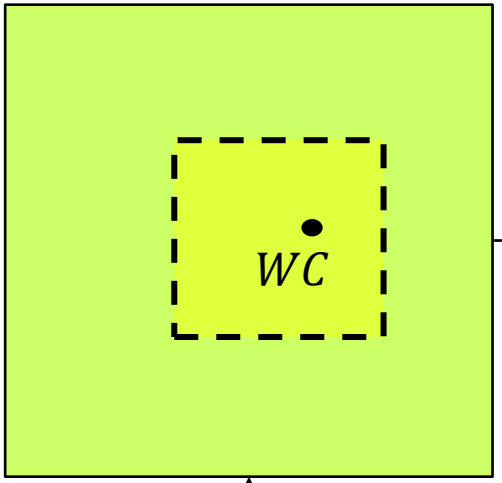
$$WB^{(k)}=SB^{(k)}\otimes K$$



$$WB^{(k)};\\k=(I\%8)\times 8+(J\%8)$$

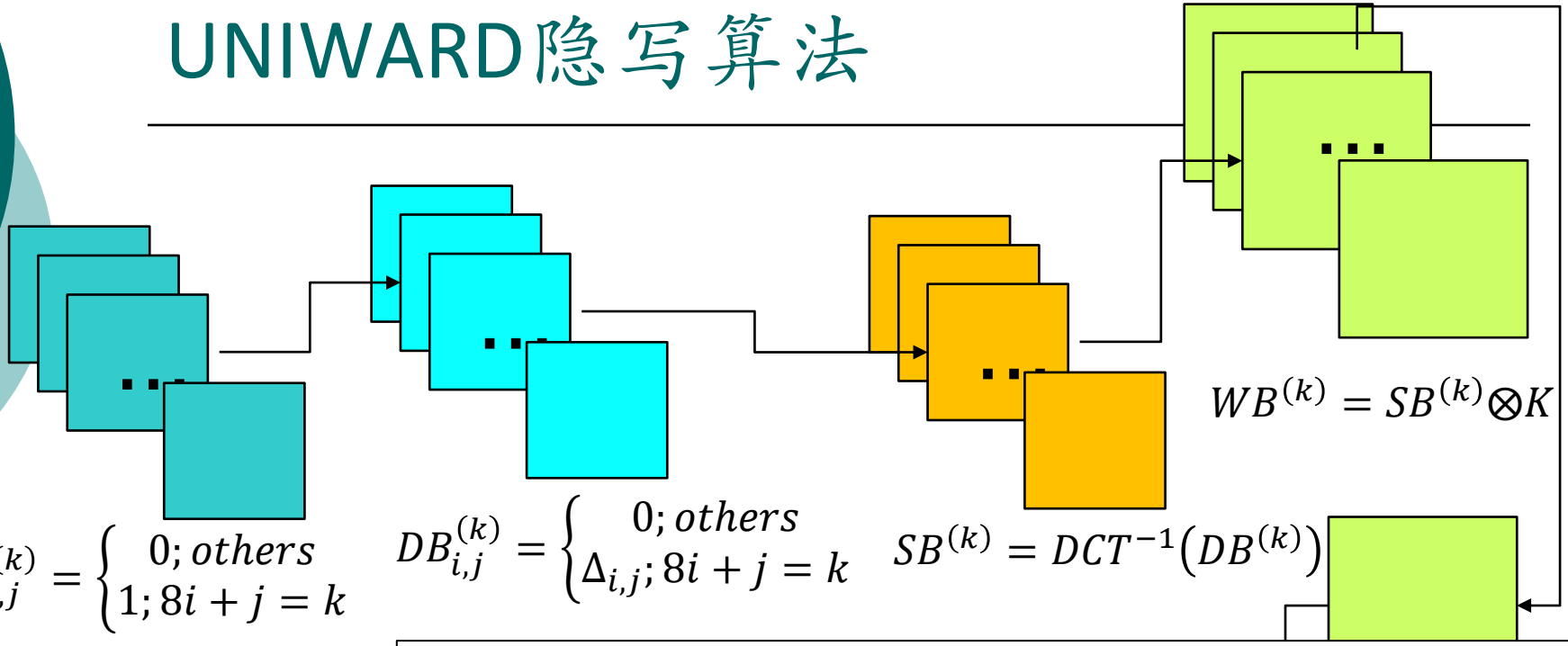






$$C_{I,J}=\sum_{u,v}\frac{WB_{u,v}^{(k)}}{\sigma+\left|WC_{u,v}\right|}$$

UNIWARD隐写算法



$$JB_{i,j}^{(k)} = \begin{cases} 0; others \\ 1; 8i + j = k \end{cases}$$

$$DB_{i,j}^{(k)} = \begin{cases} 0; others \\ \Delta_{i,j}; 8i + j = k \end{cases}$$

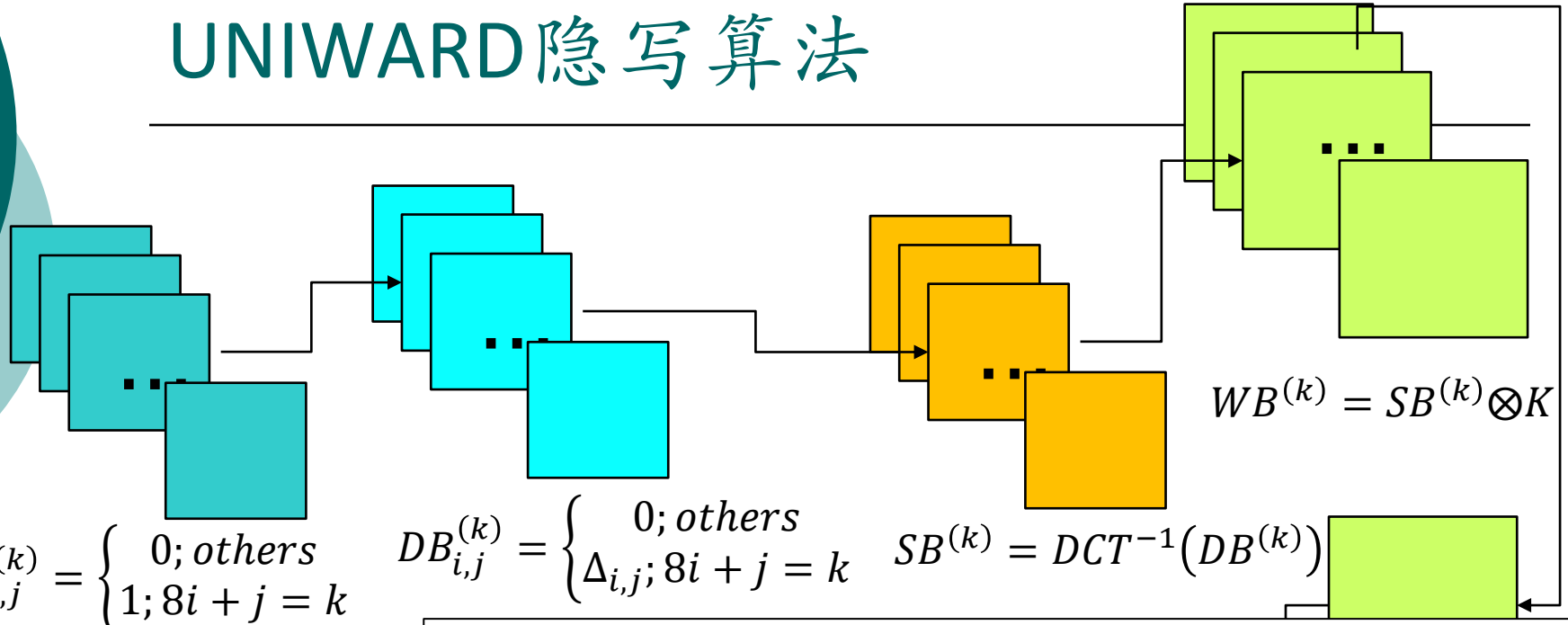
$$SB^{(k)} = DCT^{-1}\left(DB^{(k)}\right)$$

$$WB^{(k)} = SB^{(k)} \otimes K$$

像素扰动数组：仅1个位置JPEG系数隐写时，小块像素变化量。
testCoeffs：8×8，仅1个位置为1（有变化）其余位置为0的变化矩阵
spatialImpact：计算DCT逆变换得到像素扰动数组，数据类型为8×8 cell，每个cell为8×8矩阵，即像素扰动数组是一个4维数组：8×8×8×8

```
spatialImpact = cell(8, 8);
for bcoord_i=1:8
    for bcoord_j=1:8
        testCoeffs = zeros(8, 8);
        testCoeffs(bcoord_i, bcoord_j) = 1;
        spatialImpact{bcoord_i, bcoord_j} = idct2(testCoeffs)*C_QUANT(bcoord_i, bcoord_j);
    end
end
```

UNIWARD隐写算法



$$JB_{i,j}^{(k)} = \begin{cases} 0; others \\ 1; 8i + j = k \end{cases}$$

$$DB_{i,j}^{(k)} = \begin{cases} 0; others \\ \Delta_{i,j}; 8i + j = k \end{cases}$$

$$SB^{(k)} = DCT^{-1}\left(DB^{(k)}\right)$$

$$WB^{(k)} = SB^{(k)} \otimes K$$

小波扰动数组：仅1个位置JPEG系数隐写时，小波系数扰动量。包含水平、垂直和对角线共3个方向。

waveletImpact：数据类型为8×8 cell，每个cell为23×23矩阵。采用‘full’方式滤波，以便保留边界信息，输入矩阵为8×8，滤波器大小为16×16，卷积后大小为23×23，即输出矩阵大于输入矩阵。所以小波扰动数组为5维数组，3× 8×8 × 23×23

```
waveletImpact = cell(numel(F), 8, 8);
for Findex = 1:numel(F)
    for bcoord_i=1:8
        for bcoord_j=1:8
            waveletImpact{Findex, bcoord_i, bcoord_j} = imfilter(spatialImpact{bcoord_i, bcoord_j}, F{Findex}, 'full');
        end
    end
end
```

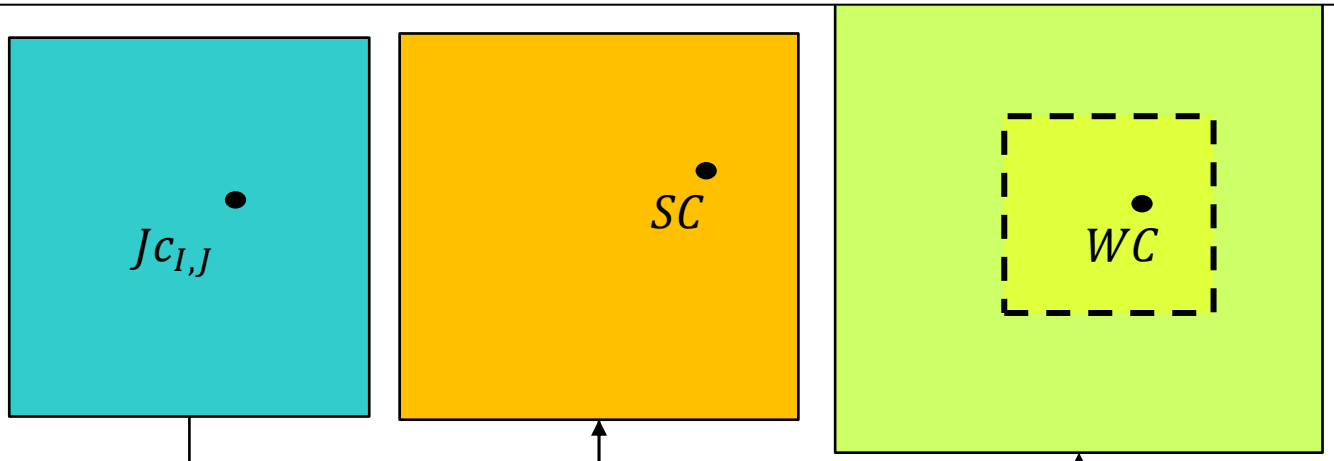
UNIWARD隐写算法

- J-UNIWARD

```
padSize = max([size(F{1}) ; size(F{2})]);
C_SPATIAL_PADDED = padarray(C_SPATIAL, [padSize padSize], 'symmetric');

RC = cell(size(F));
for i=1:numel(F)
    RC{i} = imfilter(C_SPATIAL_PADDED, F{i});
end
```

参考系数数组：计算载体三个方向小波系数，计算使用大小不变方式卷积，得到一个3维数组： 3×（M+ 2× padSize）×（N+ 2× padSzie）

像素填充矩阵：为保留边界信息，填充像素矩阵后再滤波。填充大小padSize由滤波器大小决定，当前参数为16。填充方式，对称，上下左右各填充padSize，矩阵大小变为（M+ 2× padSize）×（N+ 2× padSzie）



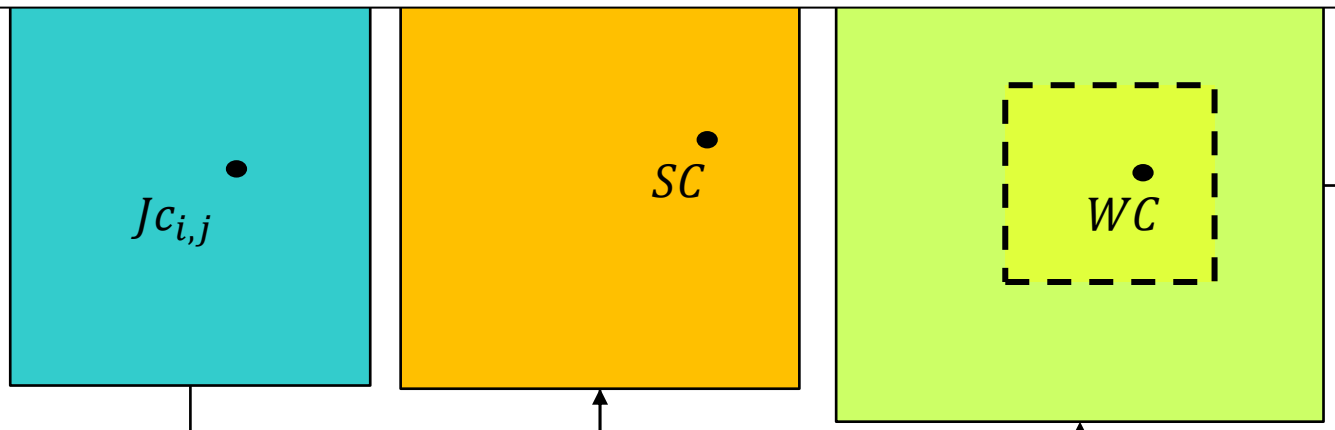
```
modRow = mod(row-1, 8)+1;
modCol = mod(col-1, 8)+1;

subRows = row-modRow-6+padSize:row-modRow+16+padSize;
subCols = col-modCol-6+padSize:col-modCol+16+padSize;
```

算法

小波扰动数组索引：JPEG以8×8的小块为处理单元进行压缩，只需存储64个位置的扰动信息即可，相应地，索引隐写带来的小波扰动时需要将坐标（row,col）映射到块内坐标（(row-1)%8+1, (col-1)%8+1 ）

参考系数数组索引：同一块不同位置系数被隐写时，计算相对小波失真用的载体参考小波系数是同一块：

1. 计算**块坐标**：块起始行为row-modRow，块结束行为row-modRow+7。列坐标类似。列坐标类似。
2. 计算**扩展块**坐标：经过DCT逆变换和小波变换，1个系数变化的影响范围延伸到块外，称为扩展块。扩展块起始行为row-modRow-6（上边界上延6行），块结束行为row-modRow+16（下边界下延9行）。列坐标类似。
3. **坐标平移**：计算参考系数数组时，在原始像素数组基础上，上下边界都填充了padSize行，所以索引时也应相应平移。列坐标类似。



$$WB^{(k)} = SB^{(k)} \otimes K$$

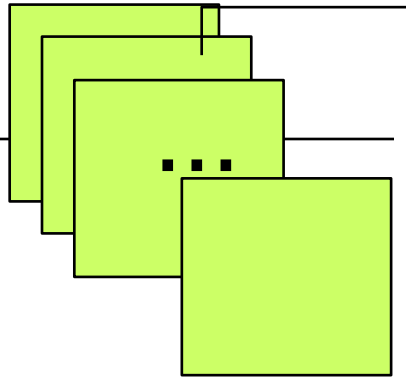
$$WB^{(k)};$$

$$k = (I\%8) \times 8 + (J\%8)$$

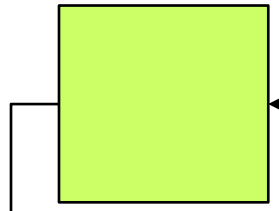
$$C_{I,J} = \sum_{u,v} \frac{WB_{u,v}^{(k)}}{\sigma + |WC_{u,v}|}$$

```
for fIndex = 1:3
    % compute residual
    RC_sub = RC{fIndex}(subRows, subCols);
    % get differences between cover and stego
    wavCoverStegoDiff = waveletImpact{fIndex, modRow, modCol};
    % compute suitability
    tempXi{fIndex} = abs(wavCoverStegoDiff) ./ (abs(RC_sub)+sgm);
end
rhoTemp = tempXi{1} + tempXi{2} + tempXi{3};
rho(row, col) = sum(rhoTemp(:));
```

代价计算：取一个参考系数扩展块RC_sub（23×23），取对应位置JPEG系数变化时的小波扰动waveletImpact{fIndex, modRow, modCol}（23×23），按点除计算绝对和得到小波相对失真。三个通道的小波相对失真之和为此处代价。（tempXi：3×1 cell（23×23），rhoTemp：23×23，rho：M×N）

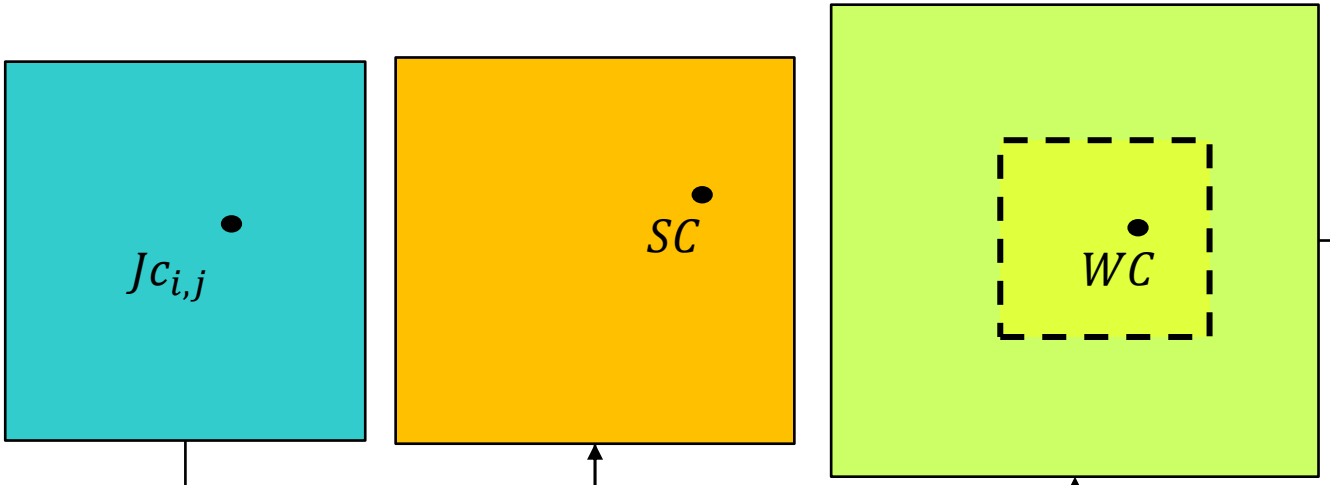


$$WB^{(k)} = SB^{(k)} \otimes K$$



$$WB^{(k)};$$

$$k = (I\%8) \times 8 + (J\%8)$$

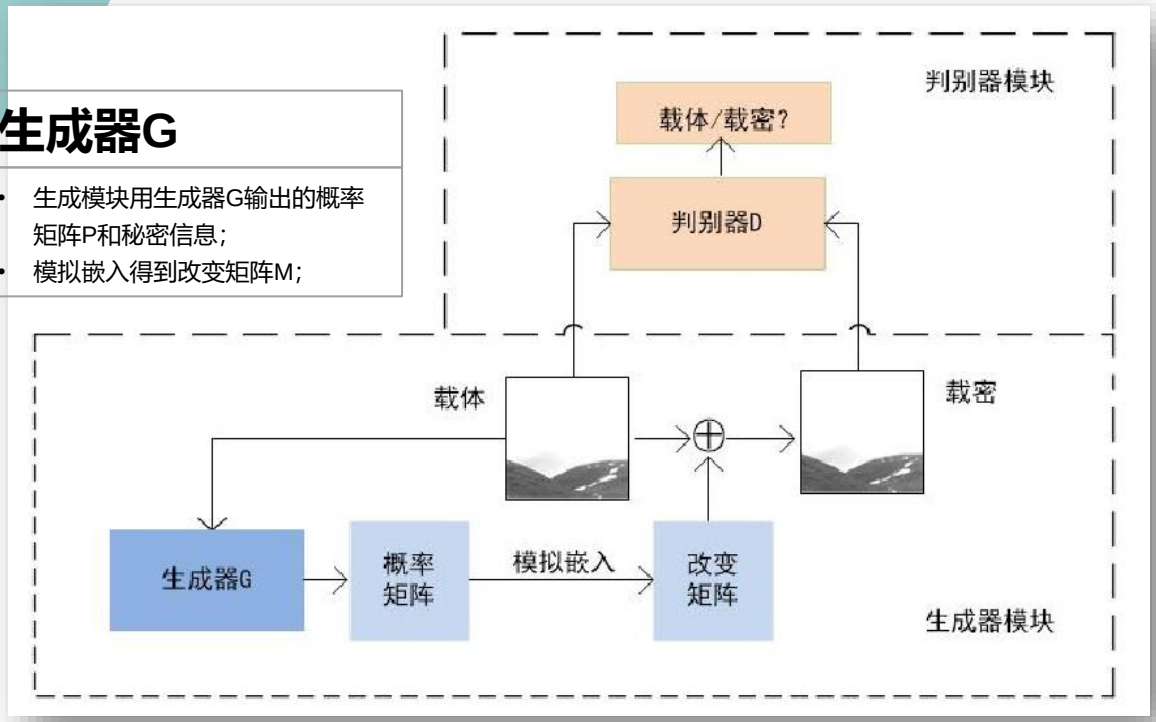


$$C_{I,J} = \sum_{u,v} \frac{WB_{u,v}^{(k)}}{\sigma + |WC_{u,v}|}$$

基于生成对抗网络的自适应隐写算法

ASDL-GAN

模型的本质是通过网络搜索合适的嵌入位置，再利用传统编码的方法嵌入秘密信息。虽然ASDL-GAN性能没有超越传统自适应隐写算法，但引入了自主学习和对抗理念，改变了传统的经验设计方式，推进了信息隐藏向高安全性进一步发展。



生成器G

- 生成模块用生成器G输出的概率矩阵P和秘密信息；
- 模拟嵌入得到改变矩阵M；

判别器D

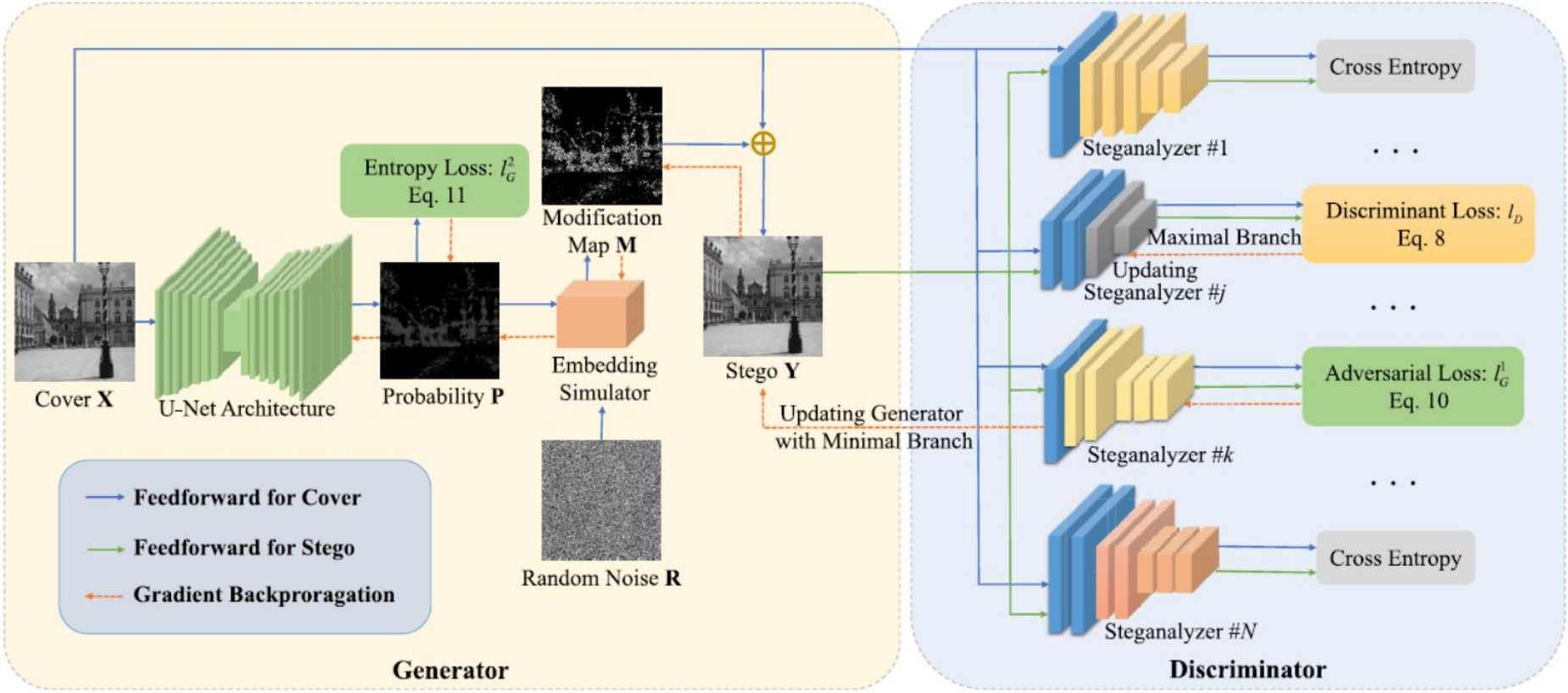
- 判别器模块则识别载体和载密图像，即隐写分析；.
- 根据判别器D的分类结果，生成器和判别器能够交替训练，更新网络参数；

改变矩阵M

- M中每个元素取0,-1,+1三值之一，指示了载体对应像素的改变量；.
- M与载体图像相加就可以得到载密（隐写）图像；.

基于生成对抗网络的自适应隐写算法

- Generative Multi-Adversarial Network (Steg-GMAN)

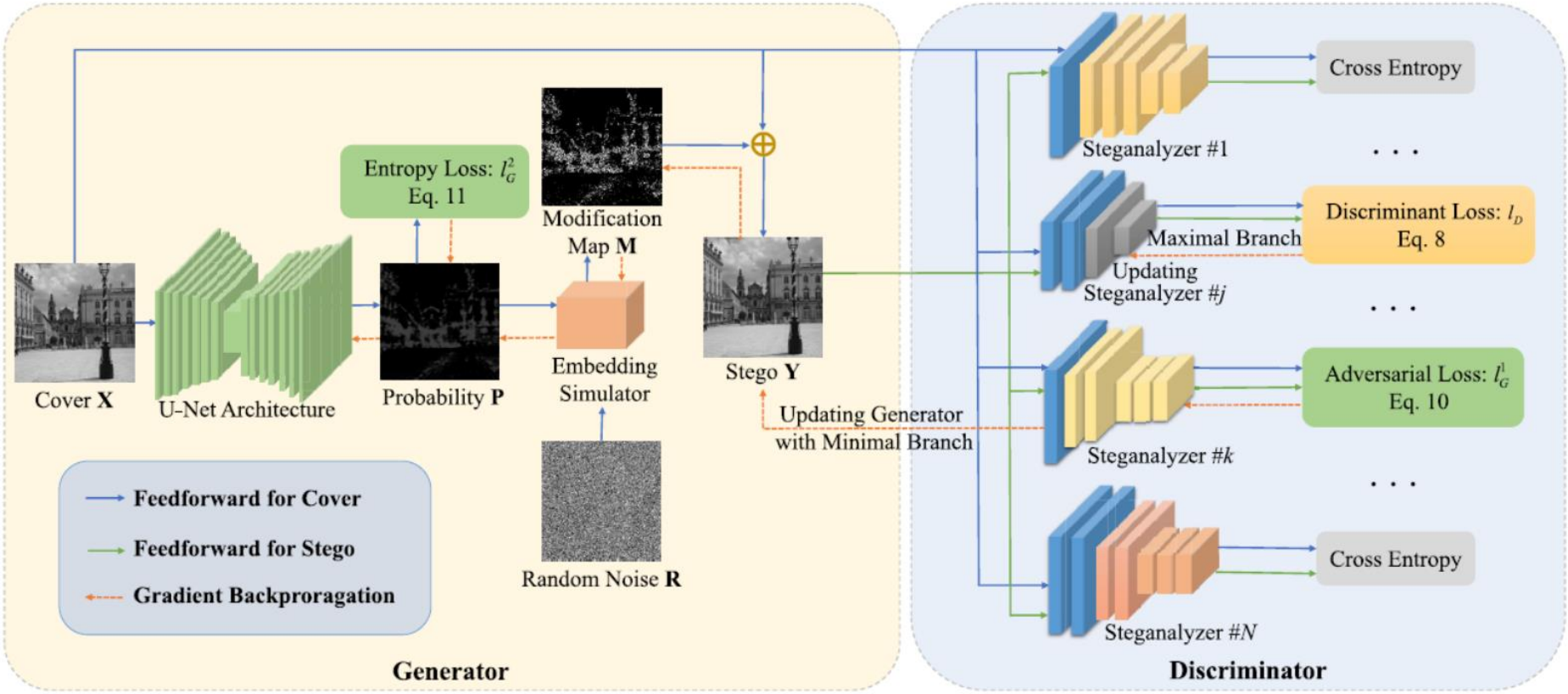


基于生成对抗网络的自适应隐写算法

- Steg-GMAN

第一步：生成概率图

输入：$\mathbf{X}=(x_{i,j})^{\mathrm{H}\times\mathrm{W}}$，输出：$\mathbf{P}=(p_{i,j})^{\mathrm{H}\times\mathrm{W}}$



$$m_{i,j} = \frac{1}{2}\tanh(\lambda(p_{i,j}^{+1} - r_{i,j})) - \frac{1}{2}\tanh(\lambda(p_{i,j}^{-1} - (1 - r_{i,j})))$$

基于生成对抗网络的自适应隐写算法

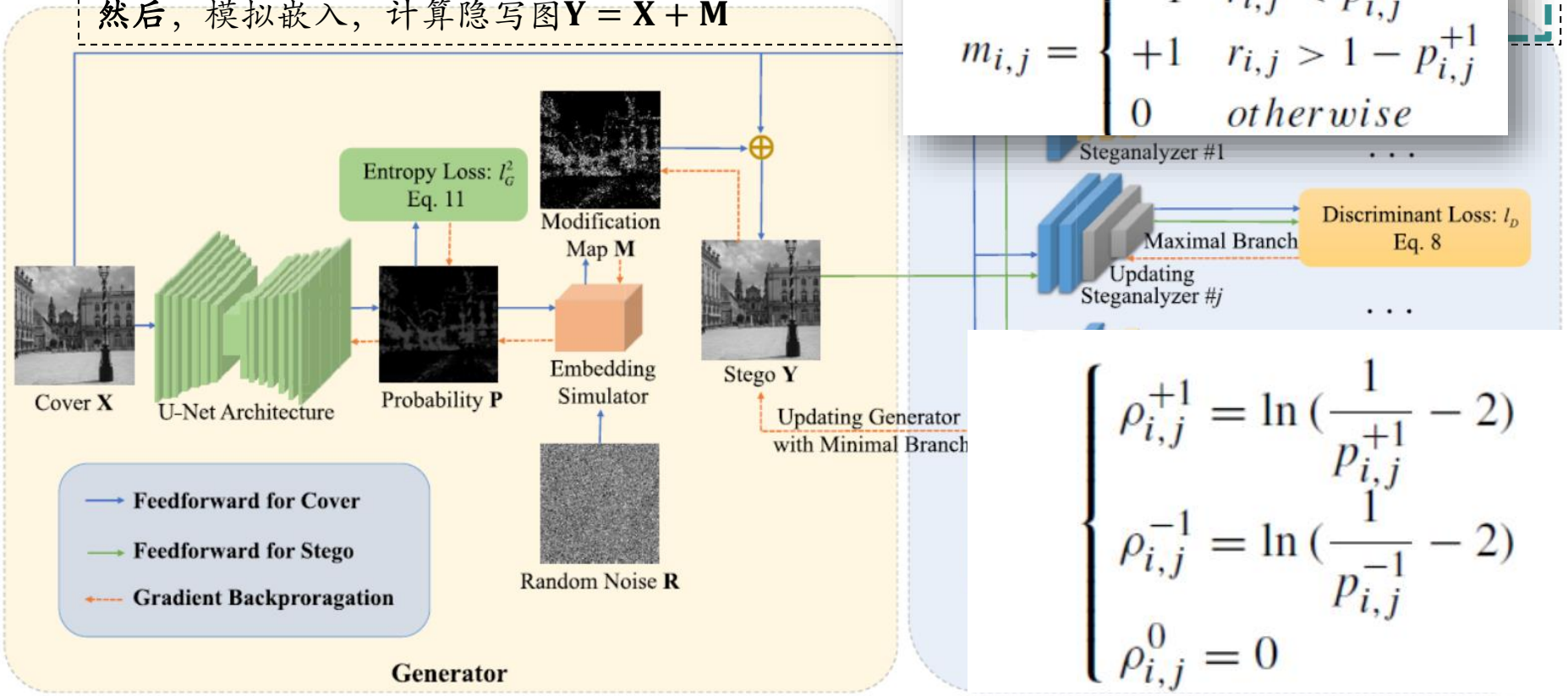
Steg-GMAN

第二步：模拟消息嵌入

首先，计算修改矩阵$\mathbf{M} = (m_{i,j})^{\mathrm{H}\times\mathrm{W}}$。令$p_{i,j}^{+1} = p_{i,j}^{-1} = \frac{1}{2}p_{i,j}$，并产生服从[0,1]上均匀分布的随机变量矩阵$\mathbf{R} = (r_{i,j})^{\mathrm{H}\times\mathrm{W}}$,比较三者关系确定修改方

然后，模拟嵌入，计算隐写图$\mathbf{Y} = \mathbf{X} + \mathbf{M}$

$$m_{i,j} = \begin{cases} -1 & r_{i,j} < p_{i,j}^{-1} \\ +1 & r_{i,j} > 1 - p_{i,j}^{+1} \\ 0 & otherwise \end{cases}$$



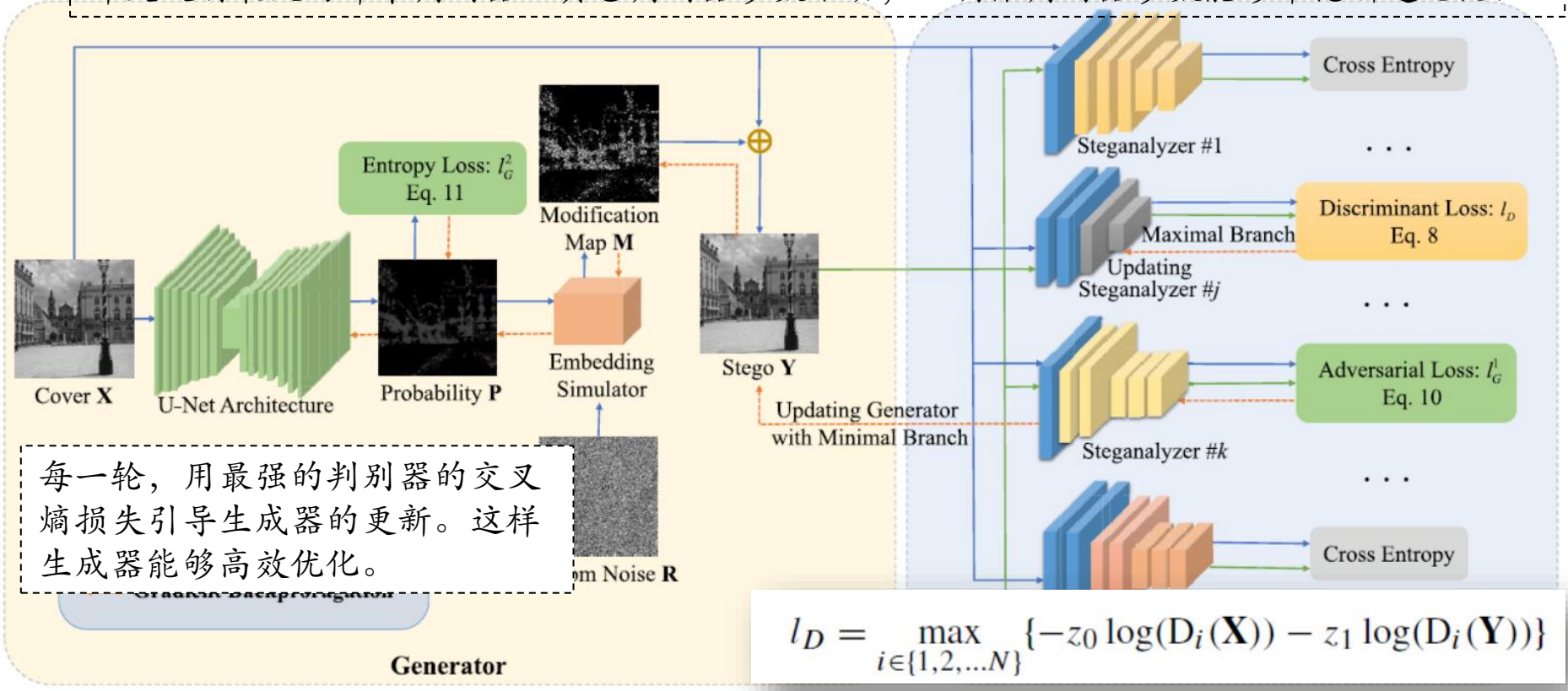
$$\begin{cases} \rho_{i,j}^{+1} = \ln\left(\frac{1}{p_{i,j}^{+1}} - 2\right) \\ \rho_{i,j}^{-1} = \ln\left(\frac{1}{p_{i,j}^{-1}} - 2\right) \\ \rho_{i,j}^{0} = 0 \end{cases}$$

基于生成对抗网络的自适应隐写算法

- Steg-GMAN

第三步：判别器的设计

计算各判别器的交叉熵$E_i = -z_0 \log\left(D_i(\mathbf{X})\right) - z_1 \log(D_i(\mathbf{Y}))$。每一轮，只更新最弱的判别器，即交叉熵最大的那个判别器。其它判别器参数不动，以确保判别器参数能够平稳渐进地优化。



每一轮，用最强的判别器的交叉熵损失引导生成器的更新。这样生成器能够高效优化。

$$l_D = \max_{i\in\{1,2,\ldots N\}} \{-z_0 \log(\mathrm{D}_i(\mathbf{X})) - z_1 \log(\mathrm{D}_i(\mathbf{Y}))\}$$

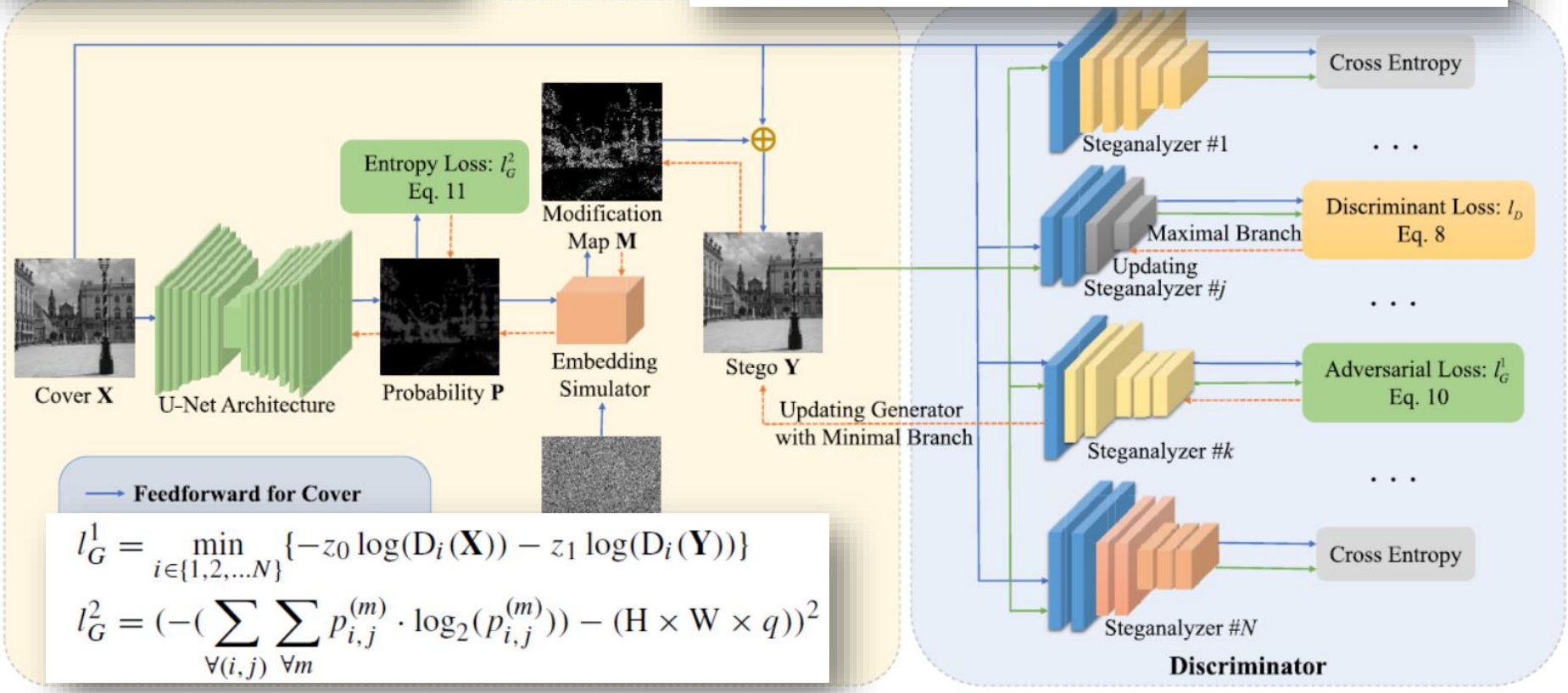
基于生成对抗网络的自适应隐写算法

- Steg-GMAN

损失函数的设计：判别器损失、生成器损失

$$l_G = -\alpha \cdot l_G^1 + \beta \cdot l_G^2$$

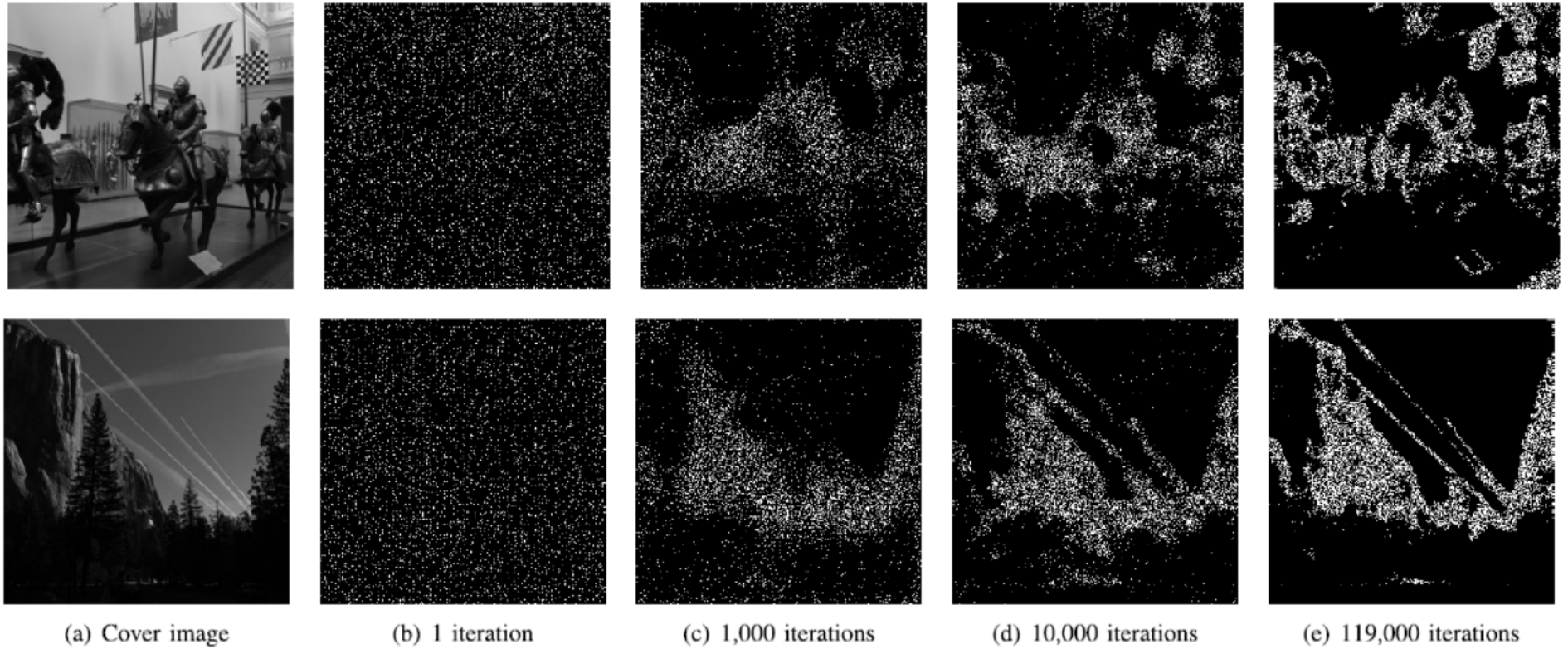
$$l_D = \max_{i\in\{1,2,\ldots N\}} \{-z_0 \log(\mathrm{D}_i(\mathbf{X})) - z_1 \log(\mathrm{D}_i(\mathbf{Y}))\}$$



$$l_G^1 = \min_{i\in\{1,2,\ldots N\}} \{-z_0 \log(\mathrm{D}_i(\mathbf{X})) - z_1 \log(\mathrm{D}_i(\mathbf{Y}))\}$$

$$l_G^2 = (-(\sum_{\forall(i,j)} \sum_{\forall m} p_{i,j}^{(m)} \cdot \log_2(p_{i,j}^{(m)})) - (\mathrm{H}\times\mathrm{W}\times q))^2$$

基于生成对抗网络的自适应隐写算法



(a) Cover image (b) 1 iteration (c) 1,000 iterations (d) 10,000 iterations (e) 119,000 iterations