RESUME

- GAUTHIER Silvère -

Quelle est la problématique ?
Quel est l'objectif ?

Nombreuses sont les applications ayant émergé récemment et nécessitant des systèmes d'intégration d'un signal, parfois appelés « signaux intégrés » ou « tatouages », au sein d'un signal « hôte ». L'intégration doit obligatoirement être faite de sorte que le signal intégré soit caché, c'est-à-dire qu'il ne cause aucune dégradation sérieuse ou visible à son hôte. En même temps, l'intégration doit être robuste aux dégradations communes appliquées au signal hôte (par exemple un changement de contrastes pour une image). En effet, le tatouage doit survivre tant que le signal hôte survit. Dans la majorité des cas, ces dégradations sont dues à des processus bénins et aux transmissions, mais peuvent être également parfois le résultat d'attaques délibérées.

Contexte

Les scientifiques considèrent le problème de l'intégration d'un signal au sein d'un autre signal « hôte », porteur, afin de former un troisième signal « composite ». Ils introduisent de nouvelles méthodes de classes d'intégration, dont une appelée « quantization index modulation » (QIM). En utilisant un modèle déterministe pour évaluer les méthodes de tatouage numérique, ils montrent que QIM est « prouvablement efficace » contre les attaques arbitrairement limitées et pleinement informées, lesquelles surviennent dans beaucoup d'applications de copyright.

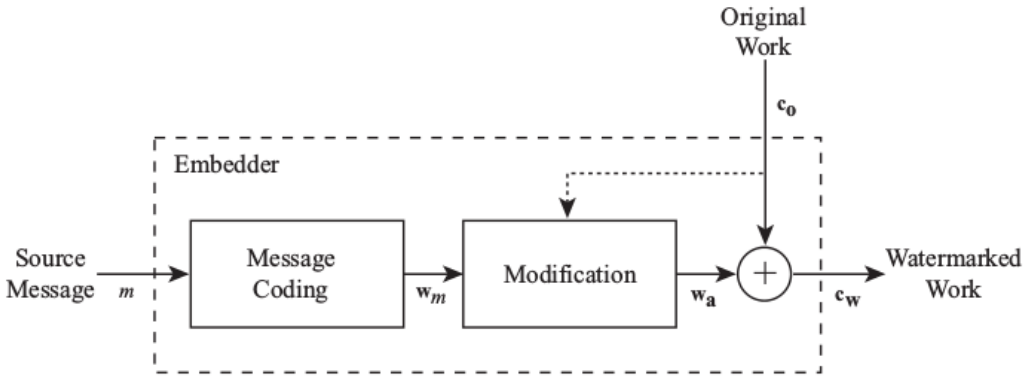


Fig. 1. Watermarking with informed embedding.

Voici un schéma représentant le principe du tatouage informé. Un message source m est codé en un signal w_m. Ensuite, le signal w_m est modifié en un signal w_a. Enfin, le signal modifié w_a est ajouté à l'image originale c_0 afin d'obtenir l'image tatouée c_w. Pour le tatouage à l'aveugle, l'étape de modification du signal w_m est indépendante de l'image originale, tandis que pour le tatouage informé, elle est fonction de cette image.

« Quantization Index Modulation »

Bref résumé de la méthode/application présentée

Cette méthode se déroule en deux grandes étapes : le codage et l'enfouissement.
- Le codage consiste à placer intelligemment les mots de code dans R2. Ces placements définiront un graphe sur lequel on appliquera l'algorithme de Voronoï.
- L'enfouissement consiste à placer y au mieux dans la cellule de Voronoï la mieux adaptée.

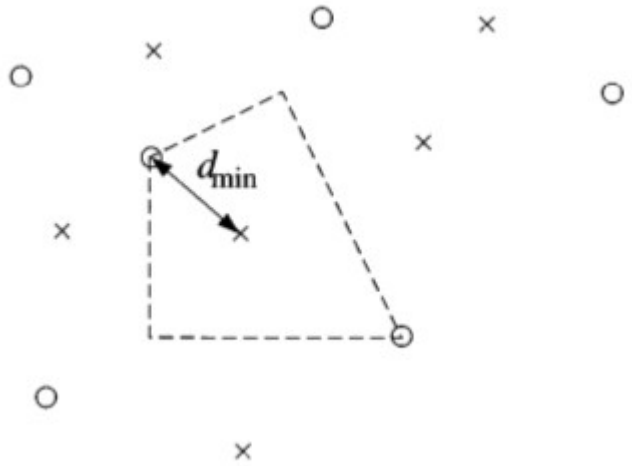
Au départ, nous devons construire un dictionnaire contenant tous les messages possibles.

Construction du dictionnaire

On utilise ici deux quantificateurs entrelacés (ayant des niveaux de reconstruction qui ne se recouvrent pas) qui correspondront aux 0 et aux 1 de notre signal. L'idée est alors de créer des sous-dictionnaires, à raison d'un pour chaque message possible. L'union de tous ces sous-dictionnaires forme alors le dictionnaire global. On peut ensuite utiliser un bruit uniforme qui servira de clef secrète par la suite.

Les sous-dictionnaires sont en réalité les niveaux de reconstruction des quantificateurs, sachant que les mots de code ne se recouvrent pas.

Déroulement de l'algorithme d'insertion



Sur ce schéma, les X et les O correspondent à deux quantificateurs différents. Ici, le X correspond au 1 et le O correspond au 0. La distance d_{min} permet de mesurer la robustesse de l'algorithme aux perturbations et la taille des cellules de Voronoï (une seule est schématisée ici) donne le degré de distorsion.

Donc si m=1, on quantifiera le signal hôte avec le X le plus proche, et si m=0, on quantifiera le signal hôte avec le O le plus proche.

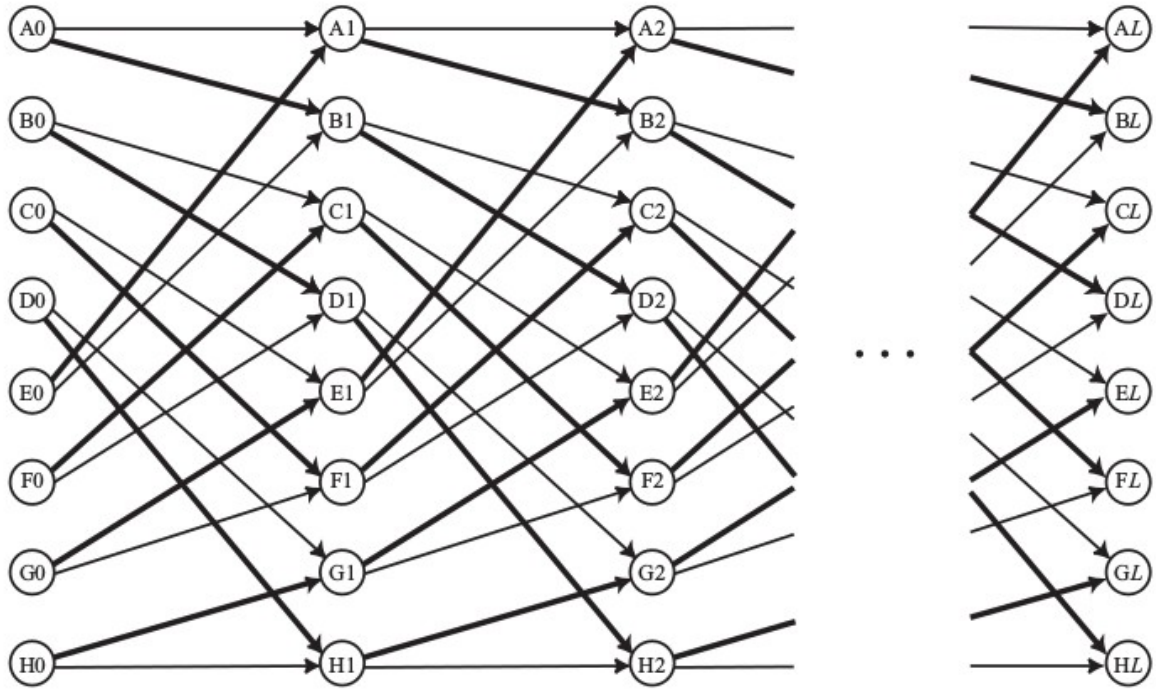
Déroulement de l'algorithme de décodage

Pour décoder le message, à partir du graphe donné ci-dessus, on peut calculer une estimation du message en utilisant la règle du plus proche voisin.

« Informed Coding and Embedding to Design a Robust, High Capacity Watermark »

Bref résumé de la méthode/application présentée

Déroulement de l'algorithme d'insertion



Voici un 8-treillis. C'est un graphe composé de 8 nœuds de départ notés {A0, B0, … , H0}. Ensuite, pour chaque bit du message, on créera 8 nouveaux nœuds. Au bit i du message m, nous seront donc dans un des nœuds {Ai, Bi, … , Hi}. Comme nous pouvons le voir dans ce treillis poiur un message de longueur L, il existe deux types d'arêtes. Les arêtes en gras correspondent à un bit à 1 et les autres à un bit à 0. Chaque bit du message m fera donc traverser l'arc correspondant à sa valeur.

La particularité ici de cette technique est d'utiliser un 8-treillis de telle sorte que deux messages très proches en binaire seront assez éloignés dans leur codage afin d'éviter les ambiguïtés et les erreurs de détection. En effet, on remarque que deux messages identiques partant de A0 et allant jusqu'à AL prendront le même chemin. Par contre, si jamais un des deux messages contient un bit à 1, par exemple de A0 à B1, la seule manière de retomber sur un état A est de passer de A0 à B1, puis de B1 à C2, puis de C2 à E3, puis de E3 à A4. Cela nécessite donc quatre étapes. Voilà pourquoi cette méthode permet une bonne robustesse, puisque deux messages très similaires seront différents d'au moins quatre états dans le treillis.

On en conclut ici que chaque message a un unique chemin dans le treillis et inversement.

Déroulement de l'algorithme de décodage

Pour décoder le message, à partir du treillis donné ci-dessus, on peut calculer une estimation du message en utilisant le chemin ayant la meilleure corrélation avec celui-ci.