

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/51958966>

# Estimation of the Embedding Capacity in Pixel-pair based Watermarking Schemes

Article · November 2011

Source: arXiv

---

CITATION

1

---

READS

58

4 authors, including:



**Rishabh Iyer**

University of Washington Seattle

50 PUBLICATIONS 693 CITATIONS

SEE PROFILE



**Rushikesh Borse**

MIT Academy of Engineering

9 PUBLICATIONS 233 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



Competitive analysis of existing image quality assessment methods [View project](#)



Pixel-QUAD based Reversible Data Hiding for High Embedding Capacity [View project](#)

# Estimation of the Embedding Capacity in Pixel-pair based Watermarking Schemes

Rishabh Iyer, Rushikesh Borse, Ronak Shah and Subhasis Chaudhuri

November 28, 2011

## Abstract

Estimation of the Embedding capacity is an important problem specifically in reversible multi-pass watermarking and is required for analysis before any image can be watermarked. In this paper, we propose an efficient method for estimating the embedding capacity of a given cover image under multi-pass embedding, without actually embedding the watermark. We demonstrate this for a class of reversible watermarking schemes which operate on a disjoint group of pixels, specifically for pixel pairs. The proposed algorithm iteratively updates the co-occurrence matrix at every stage, to estimate the multi-pass embedding capacity, and is much more efficient vis-a-vis actual watermarking. We also suggest an extremely efficient, pre-computable tree based implementation which is conceptually similar to the co-occurrence based method, but provides the estimates in a single iteration, requiring a complexity akin to that of single pass capacity estimation. We also provide bounds on the embedding capacity. We finally show how our method can be easily used on a number of watermarking algorithms and specifically evaluate the performance of our algorithms on the benchmark watermarking schemes of Tian [11] and Coltuc *et al* [6].

## 1 Introduction

Reversible Watermarking [4] is a technique used to preserve the copyright of digital data (image, audio and video), while at the same time it ensures exact recoverability of the watermark as well as the cover image. This is mainly significant in applications concerning military and medical image processing, legal and multimedia archiving of valuable original works, etc. We briefly describe below some prominent schemes in reversible watermarking.

### 1.1 Past work

There are many algorithms proposed for reversible watermarking, described comprehensively in the survey papers [18, 8]. There are four major techniques of embedding watermarks in reversible watermarking schemes, namely: histogram bin shifting, lossless data compression, expansion and mapping based techniques and prediction based techniques. Histogram bin shifting based techniques [21] suffer from the basic limitation of low embedding capacity, while data compression based techniques [16] mostly involve computationally expensive algorithms. There have been several algorithms proposed and implemented based on transforms on groups of pixels [1, 5, 6, 7, 11, 14, 22],

because of the basic advantage of high embedding capacity and a modest computation cost. Majority of these techniques operate on a pair of pixels. The first of these was proposed by Tian [11] and subsequently extended by [1, 7, 14]. These techniques are location map based and hence require an additional step of data compression. Recently reversible contrast mapping (RCM) based methods [2, 5, 6] have been used to efficiently embed data without using a location map. Prediction based techniques [7, 19] have also been suggested and they use information from the neighboring pixels to embed information.

## 1.2 Multi-pass capacity estimation

Loosely the embedding capacity of an image can be described as the size of the largest watermark which can be embedded into that image. Each watermarking technique has a maximum possible embedding capacity over a single pass, and hence often it is necessary to go for multiple passes to embed a much larger watermark into the given image. Recall that multipass embedding involves at every stage, successively embedding the watermark bits into the already watermarked image from the previous stage. Consequently any watermarking application would require an estimation of the number of passes of watermarking possible as well as an analysis of the feasibility of inserting a watermark of specific length into a given image. For this purpose it is necessary to calculate the embedding capacity beforehand. In practical settings, it may be necessary to find such estimates repeatedly for different configurations of the watermark and the cover image. Hence it may not be feasible to actually embed the watermark in a given image, and to check if the watermark and the cover image are compatible for watermarking. Since most watermarking algorithms are quite slow, the possibility of having to check for multiple iterations of embedding before choosing the right watermark would make the task computationally quite demanding. In particular, these watermarking schemes tend to be quite complex and involve data-compression stages which makes the task of embedding computationally expensive. Hence we require efficient estimation algorithms for the computation of embedding capacity of different watermarking schemes. The problem of multi-pass embedding capacity estimation has not been studied much in the literature despite this being needed before any cover image or watermark could be selected for embedding. Although Kalker [12] talks about capacity bounds based on dirty water codes and Hamming codes, it mainly focuses on capacity bounds for an allowable control distortion. Li *et al.* [13] also talk about image independent embedding capacity, but the focus lies in finding the minimum possible embedding capacity for any image. Thus there is an urgent need to develop appropriate techniques to compute the embedding capacity in multi-pass watermarking schemes. For a given image, the single pass embedding capacity is simple to estimate and can be directly computed by considering pixel pairs eligible for watermarking. Multi-pass embedding capacity estimation is however challenging since the subsequent passes depend not just on the cover image but also on the watermarks embedded in the previous iterations. Hence in the rest of the paper, we focus on developing computationally efficient techniques to estimate the embedding capacity in multi-pass embedding.

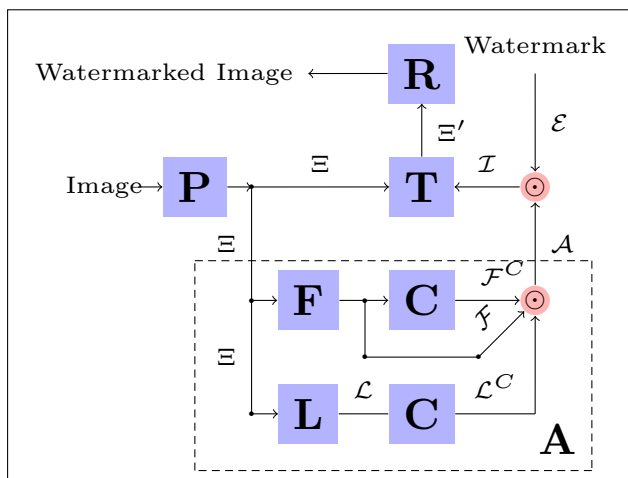


Figure 1: A block diagram depicting the procedure for this class of watermarking schemes. Here  $\mathbf{P}$  denotes the disjoint partitioning block,  $\mathbf{F}$  denotes the Flag stream generator,  $\mathbf{L}$  represents the location map bitstream generator,  $\mathbf{R}$  is the reconstruction block,  $\mathbf{C}$  represents the compression method and  $\mathbf{T}$  denotes the transform block.

## 2 Framework, Notation and Problem Definition

### 2.1 Framework and Notation

We show in this paper that for a select class of watermarking algorithms it is indeed possible to provide good estimates of the embedding capacity over multiple passes. This is a class of transforms which operate on independent groups or blocks of pixels, generally known as the expansion and mapping based algorithms. In this paper, we propose algorithms for pixel-pair based methods, since for majority of these techniques [2, 5, 7, 6, 11, 14, 22], the independent group or blocks of pixels are pixel pairs. In other words the entire image is partitioned into disjoint pairs of pixels. We further show that the proposed algorithms can also be extended easily to schemes [1] which operate on larger groups of pixels (for example pixel triplets and quadruplets).

We now introduce the notation we will use throughout this paper. Let  $\mathbb{D}$  represent the domain of the pixel pairs, i.e  $\{\mathbb{D} = [0, L] \times [0, L]\}$ , where  $L = 255$  for an 8-bit Image. Further let  $\xi$  represent a pixel pair  $(x, y)$ , where  $x$  and  $y$  refer to the pixel intensities. The procedure involved in pixel-pair based watermarking schemes is depicted in fig. 1. As illustrated in fig. 1, the input image is first partitioned through a disjoint partitioning block  $\mathbf{P}$ , into a set of disjoint pixel pairs, represented as  $\Xi = \{\xi_1, \xi_2, \dots, \xi_N\}$ . These pixel pairs are generally adjacent to each other, either horizontally, vertically or diagonally. Let  $N$  be the total number of pixel pairs in the image. For ease of notation we will sometimes drop the subscript. We further represent  $\xi' = (x', y')$ , as the transformed pixel pair, after watermarking the pixel pair  $\xi = (x, y)$ .

Recall that in order to prevent overflow and underflow we have the constraints:  $0 < x' < L, 0 < y' < L$ . Thus not every pixel pair is embeddable. Further in order to control the distortion, some additional restrictions are imposed on the embeddable domain [6, 11, 22]. Correspondingly only

those pixel pairs  $\xi$ , such that:  $|x - y| < \theta_h$  for some threshold  $\theta_h$  are considered for embedding. Thus let the corresponding domain of embeddable pixel pairs be  $\mathbb{D}_{\mathcal{T}} \subseteq \mathbb{D}$ . Let  $\mathbb{B}$  be the set of all possible bitstreams, which are embedded into an image. In order to maintain reversibility of the watermarking algorithms, many times some additional data is required to be embedded along with the watermark bitstream [8]. This data is called the auxiliary data stream denoted by  $\mathcal{A} \in \mathbb{B}$ , and hence the disjoint pixel pairs  $\Xi$  are analyzed through an auxiliary data block  $\mathbf{A}$  (shown as a densely dashed block in fig. 1), to construct the auxiliary data stream  $\mathcal{A}$ , needed to be embedded along with the image. Thus within the embeddable pixel pairs, both the watermark and the auxiliary information have to be embedded. There are two types of auxiliary data used in watermarking algorithms belonging to this class. They are the flag bits and the location map. Thus the auxiliary block  $\mathbf{A}$  consists of the flag bitstream generator  $\mathbf{F}$  and the compressed location map stream generator  $\mathbf{L}$ . Flag bits are typically required to be embedded along with the watermark to ensure reversibility. In most algorithms these are either in the form of LSB bits of some pixels which need to be stored, or flags containing information regarding some pixel pairs. We represent the domain of pixel pairs which contribute towards the flag bit stream as  $\mathbb{D}_{\mathcal{F}} \subseteq \mathbb{D}$ . In other words for every pixel pair belonging to  $\mathbb{D}_{\mathcal{F}}$ , we need to store bits either in the form of flags or LSB bits. Generally  $\mathbb{D}_{\mathcal{F}} \subset \mathbb{D}$ , as not every pixel pair requires a flag bit to be stored. Let  $f_{\xi}$  be the flag contributed by the pixel pair  $\xi$ . In most algorithms  $f_{\xi}$  is a binary bit, and atmost one bit needs to be stored for every pixel pair  $\xi$ . Further these bits need to be stored only for pixel pairs  $\xi \in \mathbb{D}_{\mathcal{F}}$  and hence for every  $\xi \notin \mathbb{D}_{\mathcal{F}}$ ,  $f_{\xi} = \{\phi\}$ . Here  $\phi$  represents a nullbit. Let further  $\eta(f_{\xi})$  represent the number of bits contributed by  $\xi$ . Hence  $\eta(f_{\xi}) = 0$ , if  $f_{\xi} = \{\phi\}$ . We can then represent the flag bitstream as  $\mathcal{F} = \mathbf{F}(\Xi) = \{f_{\xi_1}, f_{\xi_2}, \dots, f_{\xi_N}\}$ . For those  $\xi_j \notin \mathbb{D}_{\mathcal{F}}$ ,  $f_{\xi_j} = \{\phi\}$  and they do not contribute towards  $\mathcal{F}$ . The location map on the other hand is a binary map which stores information for every pixel pair in the image. For example in Tian's scheme [11], the location map consists of information whether a pixel pair is expandable or not. Let  $l_{\xi}$  denote the location map bit corresponding to the pixel pair  $\xi$ . We denote  $\mathcal{L} = \mathbf{L}(\Xi) = \{l_{\xi_1}, l_{\xi_2}, \dots, l_{\xi_N}\}$  as the location map bit stream. Unlike the flagbit stream, the location map is required for every pixel pair in the image (i.e  $\eta(l_{\xi}) = 1, \forall \xi \in \mathbb{D}$ ) and hence it cannot be directly embedded in the form of the auxiliary data since its size is the same as the total number of pixel pairs in the image. Thus it has to be compressed using a compressing method  $\mathbf{C}$ . We then represent the compressed location map as  $\mathcal{L}^C = \mathbf{C}(\mathcal{L})$ . Sometimes the flag bit stream may also be compressed and we represent the compressed flag bit stream as  $\mathcal{F}^C = \mathbf{C}(\mathcal{F})$ . Hence the auxiliary data stream  $\mathcal{A} = \mathcal{F} \odot \mathcal{L}^C \odot \mathcal{F}^C$ . Here  $\odot$  indicates the concatenation of bitstreams. Hence we have:  $\eta(\mathcal{A}) = \eta(\mathcal{F}) + \eta(\mathcal{L}^C) + \eta(\mathcal{F}^C)$ . Note that one or more of these may be null streams, depending on the watermarking scheme used. As we shall see later, the sizes of the compressed location map and flag bits may depend on the number of ones and zeros in those streams. We assume  $\omega(\cdot)$  denotes the number of ones in a bitstream and hence  $\omega(\mathcal{L})$  and  $\omega(\mathcal{F})$  denote the number of ones in the location map and flag bitstream respectively. Let  $\mathbb{D}_{\mathcal{L}}^1$  be the region of pixel pairs where the location map bit is 1. In other words:  $\mathbb{D}_{\mathcal{L}}^1 = \{\xi \in \mathbb{D} | l_{\xi} = 1\}$ . Similarly we can define  $\mathbb{D}_{\mathcal{F}}^1 = \{\xi \in \mathbb{D}_{\mathcal{F}} | f_{\xi} = 1\}$  as the region of pixel pairs where the flag bit is 1.

We represent the watermark bitstream as  $\mathcal{E} \in \mathbb{B}$ . Correspondingly we can construct the total embeddable bitstream  $\mathcal{I} = \mathcal{E} \odot \mathcal{A}$ , with  $\eta(\mathcal{I}) = \eta(\mathcal{E}) + \eta(\mathcal{A})$ . Since we are interested in finding the embedding capacity, we assume that  $\mathcal{E}$  represents the largest possible bitstream, and correspondingly the embedded bits  $\mathcal{I}$ , will be embedded in every embeddable pixel pair. Thus  $\eta(\mathcal{I})$  is equal to the total number of embeddable pixel pairs in the image. Let  $i_{\xi}$  denote the bit embedded in

the pixel pair  $\xi$ . Further we assume  $\eta(i_\xi)$  denotes the number of bits embedded in  $\xi$ . Given the embeddable bitstream  $\mathcal{I}$ , we can then construct an ordered sequence of bits, to be embedded into the image such that:  $\mathcal{I} = \{i_{\xi_1}, i_{\xi_2}, \dots, i_{\xi_N}\}$  with,  $i_{\xi_j} = \phi$ , if  $\xi_j \notin \mathbb{D}_{\mathcal{I}}$  or  $\eta(i_{\xi_j}) = 0$ .

Each pixel pair  $\xi = (x, y)$  is mapped to another pixel pair  $\xi' = (x', y')$  through a transform:  $\xi' = T(\xi, i_\xi)$ . Let  $\mathbf{T}$  denote the transform block. Let  $\xi^0 = T(\xi, 0)$ ,  $\xi^1 = T(\xi, 1)$  and  $\xi^\phi = T(\xi, \phi)$ . Hence  $\xi' = \{\xi^0, \xi^1, \xi^\phi\}$ . Thus we embed the bitstream  $\mathcal{I}$  into the pixel pairs,  $\Xi$ , to obtain the set of transformed pixel pairs:  $\Xi' = \mathbf{T}(\Xi, \mathcal{I}) = \{T(\xi_1, i_{\xi_1}), T(\xi_2, i_{\xi_2}), \dots, T(\xi_N, i_{\xi_N})\} = \{\xi'_1, \xi'_2, \dots, \xi'_N\}$ . Finally the watermarked image can be obtained by reconstructing the image from the set of pixel pairs  $\Xi'$ , through a image reconstruction block  $\mathbf{R}$ . This entire procedure is illustrated in fig.1. The maximum embedding capacity for these algorithms is 0.5 bpp since we can embed atmost a bit in each pixel pair. However the auxiliary data eats up this capacity. Hence  $\eta(\mathcal{E}) = \eta(\mathcal{I}) - \eta(\mathcal{A})$  represents the size of the largest possible watermark embeddable in the image and is the embedding capacity of the image.

In multipass embedding, the entire embedding stage depicted in fig. 1, is repeated at every stage on the watermarked image from the previous stage. Further note that throughout this paper we also assume that at every subsequent pass of embedding the bitstream will be embedded in the same set of disjoint pixel pairs, which were selected in the first pass of embedding. We represent the number of passes in multi-pass watermarking as  $P$ . In multipass embedding, let  $\mathcal{I}_k, \mathcal{A}_k, \mathcal{E}_k, \mathcal{L}_k, \mathcal{F}_k$ , etc. refer to the corresponding bitstream of the  $k^{th}$  stage of watermarking. In other words,  $\mathcal{I}_0 = \{i_{\xi_0}, i_{\xi_1}, \dots, i_{\xi_N}\}, \mathcal{I}_1 = \{i_{\xi'_0}, i_{\xi'_1}, \dots, i_{\xi'_N}\}$  and so on. We represent the final concatenated bitstreams after  $P$  passes as  $\mathcal{I}, \mathcal{A}, \mathcal{E}, \mathcal{F}, \mathcal{L}$ , etc. For example  $\mathcal{I} = \mathcal{I}_0 \odot \mathcal{I}_1 \odot \dots \odot \mathcal{I}_{P-1}$ . Also let  $p$  be the fraction of the number of ones in the embedded bitstream to the total size of the bitstream.

In this paper, we provide a general framework of algorithms applicable to any watermarking scheme fitting in the above mentioned class. We do not confine our analysis to any particular watermarking scheme and try to keep our algorithms as general as possible. However we give some examples below of few watermarking schemes which fit into this framework and briefly describe the embedding regions and the auxiliary data required by them.

- Tian [11]: This is the first paper on difference expansion. It embeds a watermark bit into the difference of pixels and requires a location map. Here the set of embeddable pixel pairs is exactly those pixels which are changeable. Further the auxiliary data comprises here of the compressed location map and flag bits in the form of LSB bits. The domains  $\mathbb{D}_{\mathcal{I}}, \mathbb{D}_{\mathcal{F}}, \mathbb{D}_{\mathcal{L}}^1$  and  $\mathbb{D}_{\mathcal{F}}^1$  for Tian's scheme are shown in fig. 2. Here we have  $f_\xi = \text{LSB}(|x - y|)$ .
- Thodi *et al* [7]: This is an extension of Tian's algorithm. It uses a combination of histogram bin shifting and difference expansion. The two main algorithms here are difference expansion with histogram shifting using overflow map (DE-HS-OM), and difference expansion with histogram expansion using flag bits (DE-HS-FB). The method is similar to Tian's but they select locations for embedding by defining non overlapping regions in the histogram of the expandable differences. The regions here are similar to those of Tian's and we do not show them separately.
- Coltuc *et al* [6]: They use a reversible contrast mapping (RCM) method of embedding watermark using a simple integer transform on pairs of pixels. Again the domains  $\mathbb{D}_{\mathcal{I}}, \mathbb{D}_{\mathcal{F}}$  and

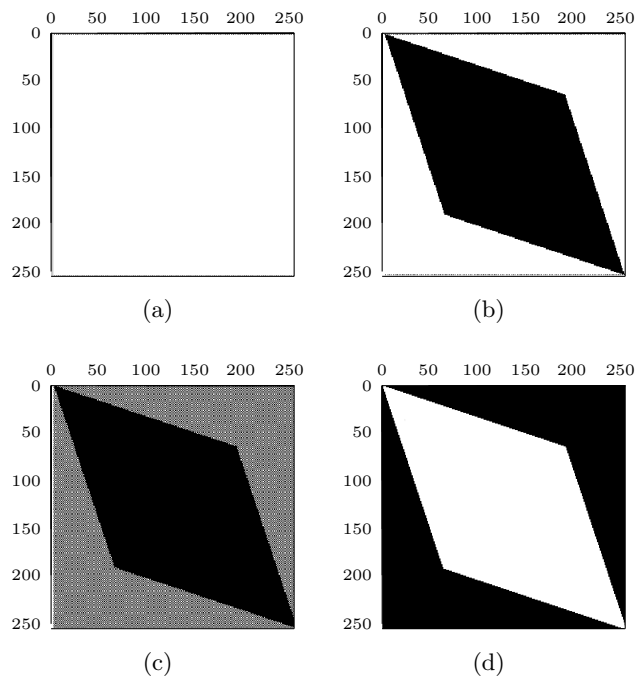


Figure 2: Illustrations of the regions of  $\mathbb{D}_{\mathcal{I}}$ ,  $\mathbb{D}_{\mathcal{F}}$ ,  $\mathbb{D}_{\mathcal{F}}^1$ , and  $\mathbb{D}_{\mathcal{L}}^1$ , for the scheme of Tian [11]. The regions are plotted as  $255 \times 255$  matrices and the white region represents the corresponding domains. Note that here  $\mathbb{D}_{\mathcal{I}}$  is almost the entire region  $\mathbb{D}$ . We assume  $\theta_h = 255$ .

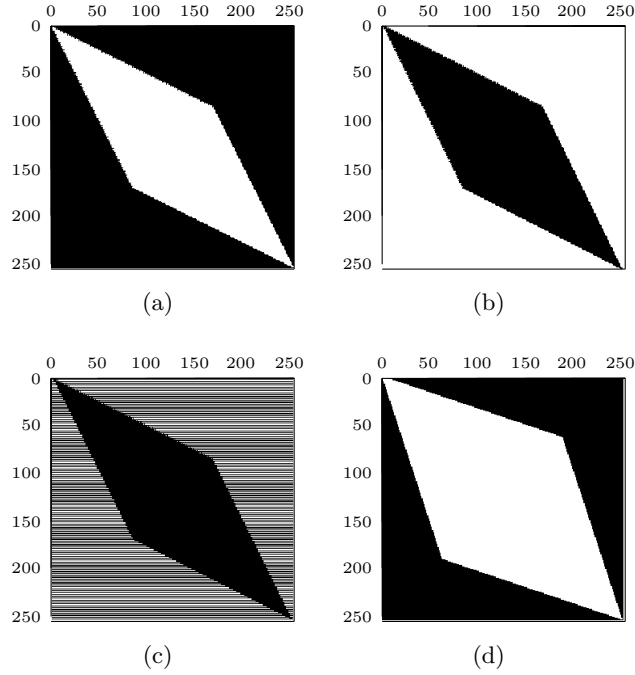


Figure 3: The regions in above two and the bottom-left figures represent  $\mathbb{D}_{\mathcal{I}}, \mathbb{D}_{\mathcal{F}}$  and  $\mathbb{D}_{\mathcal{F}}^1$  respectively for Coltuc's scheme. The bottom left Image represents the region  $\mathbb{D}_{\mathcal{I}}$  for the scheme of Weng *et al.* The regions are plotted as  $255 \times 255$  matrices, and the white region represents the corresponding domains.

$\mathbb{D}_{\mathcal{F}}^1$  for Coltuc's scheme are shown in fig. 5a, 5b and 3c. Further we have here:  $f_{\xi} = \text{LSB}(x)$ .

- Weng *et al* [22]: The concept of invariability of the sum of pixel pairs and pairwise difference adjustment (PDA) is exploited to embed a watermark in a pixel pair. This method requires location map compression. Again the different domains for the scheme of Weng are shown in fig. 3d.

As it is clear from the above summary, majority of the techniques are location map based and require additional data compression. The compression of the location map significantly increases the complexity of the algorithms, further emphasizing the need to provide efficient multi-pass embedding capacity estimates as described in section 1. Each watermarking algorithm requires atleast two iterations over the entire image to first identify the embeddable regions and get the auxiliary information and then actually embedding the watermark, in addition to a possibly extra iteration to compress the location map. This is also evident from the watermarking procedure shown in fig. 1. In particular the location map compression is a computationally expensive task. Thus the watermarking schemes have a lot of associated overhead. The RCM based method however does not require any location map, and is comparatively the most efficient algorithm. We however show that our estimation method performs much better than even Coltuc's method computationally, while still providing reasonably precise estimates.



## 2.2 Problem Definition

We assume that the probability that a given bit in the bitstream is 1 is  $p$ , and is known. We claim here that due to the block based approach of this class of watermarking schemes, the embedding capacity for a given image and watermark depends only on  $p$  and not on the actual bitstream itself. Infact we experimentally verify this in section 5.1. Hence we pose our problem as providing good estimates of the total embedding capacity  $\eta(\mathcal{E})$ , in multi-pass watermarking schemes given  $p$  and the number of passes  $P$ . Estimating the embedding capacity  $\eta(\mathcal{E})$  requires estimation of total number of embeddable pixel pairs  $\eta(\mathcal{I})$ , along with the auxiliary data size  $\eta(\mathcal{A})$ , required to ensure reversibility.

## 2.3 Our Contributions

We exploit specific properties of the pixel pair based watermarking schemes to efficiently estimate the multi-pass embedding capacity in computational costs significantly lower than actual watermarking. We first propose a co-occurrence based method which iteratively updates the co-occurrence matrix of the image to effectively estimate both the embedding information and the auxiliary data size required at every pass. Subsequently we propose a pre-computable tree based implementation which concisely represents the multi-pass structure for every pixel pair. We then prove the equivalence between these two methods. We then use these algorithms to estimate the embedding capacity of an image for a given value of  $p$ . Lastly we also propose methods to estimate bounds on the embedding capacity. Though this is mainly a theoretical paper, we perform a number of experiments and evaluate our algorithms on the watermarking schemes of Tian and Coltuc *et al.* We show that these estimates are reasonably close to the actual capacities of these images.

# 3 Embedding capacity estimation

In this section we present methods to estimate the embedding capacities, given the bitstream distribution ( $p$ ). The pixel pair tree method though a very fast estimation procedure, requires an offline stage and some additional memory. The co-occurrence matrix method is, however, an iterative method, but is amenable to considering different probabilities at every iteration.

## 3.1 Proposed Algorithms

Let  $\mathcal{B} \in \mathbb{B}$  represent a general bitstream, to be embedded into an image. It could represent the embedded bitstream ( $\mathcal{I}$ ), watermark ( $\mathcal{E}$ ), flag bit stream ( $\mathcal{F}$ ) or the location map bit stream ( $\mathcal{L}$ ). There are typically two problems of interest related to this. One is estimating the size of the bitstream  $\eta(\mathcal{B})$ , like estimating the number of embeddable pixel pairs  $\eta(\mathcal{I})$  or the size of the flag bit stream  $\eta(\mathcal{F})$ . The other is estimating the number of ones in the bitstream  $\omega(\mathcal{B})$ , which is relevant in estimating the compression of the bitstream, as we shall see later. For example, to estimate the size of the compressed streams  $\eta(\mathcal{L}^C)$  or  $\eta(\mathcal{F}^C)$  we would need to estimate the number of ones in the corresponding streams,  $\omega(\mathcal{F})$  and  $\omega(\mathcal{L})$ , respectively. Also we assume that  $\mathcal{B}_k$  denotes the bitstream corresponding to the  $k^{th}$  stage of embedding. Let  $b_\xi$  represent the bits contributed by the pixel pair  $\xi$ , and  $\eta(b_\xi)$  represent the number of bits contributed by the pixel pair  $\xi$  towards the

bitstream  $\mathcal{B}$  in a single pass. Since we are concerned mainly with pixel pair based watermarking schemes,  $b_\xi$  is a bit and  $\eta(b_\xi)$  is either 0 or 1, depending whether  $\xi$  contributes towards  $\mathcal{B}$  or not. Again for example, when  $\mathcal{B}$  is  $\mathcal{I}, \mathcal{F}$  and  $\mathcal{L}$ ,  $b_\xi$  is  $i_\xi, f_\xi$  and  $l_\xi$  respectively. Further,  $\mathbb{D}_\mathcal{B}$  represents the region of pixel pairs which contribute towards the bitstream and  $\mathbb{D}_\mathcal{B}^1$  represents the pixel pairs where  $b_\xi = 1$ . In other words  $\mathbb{D}_\mathcal{B} = \{\xi \in \mathbb{D} | b_\xi \neq \phi\}$  and  $\mathbb{D}_\mathcal{B}^1 = \{\xi \in \mathbb{D}_\mathcal{B} | b_\xi = 1\}$ .

### 3.1.1 Co-occurrence Matrix based method

**Definition 1.** (*Pair-wise co-occurrence Matrix*). We define the pairwise co-occurrence matrix  $C$  of size  $L \times L$ , similar to the conventional co-occurrence matrix [10], as the population (distribution) of co-occurring pixel pairs in an Image. In our context, it represents a count of the number of times a pixel pair occurs in the image. Thus given the disjoint pairs of pixels  $\Xi = \{\xi_1, \xi_2, \dots, \xi_N\}$ , we can define it as:

$$C(\xi) = \sum_{j=1}^N I(\xi = \xi_j) \text{ where } I(\cdot) \text{ is the indicator function.} \quad (1)$$

In this paper all subsequent usage of the term co-occurrence matrix would actually mean a pairwise co-occurrence matrix. We now provide a scheme to iteratively update the co-occurrence matrix and estimate the size of the bitstream at a given stage using the corresponding co-occurrence matrix at that stage. We start with a co-occurrence matrix  $C_0$  initially, calculated directly from the cover image. The initial image and its corresponding co-occurrence matrix represent the  $0^{th}$  stage. We then iteratively update the co-occurrence matrix at every stage (pass of embedding) using the following scheme: Let the co-occurrence matrix at the  $k^{th}$  stage be  $C_k$ . Then for every pixel pair  $\xi \in \mathbb{D}_\mathcal{I}$ ,  $p$  fraction of the total number of these pairs of  $C_k(\xi)$  will become  $\xi^1$ , while  $1 - p$  of these will transform to  $\xi^0$  in  $C_{k+1}$ . For those pixel pairs not embeddable,  $\xi$  is transformed to  $\xi^\phi$ . This is elaborated in detail in Algorithm 1.

---

**Algorithm 1** Statistical estimation of the embedding capacity from the co-occurrence matrix for a given probability of watermark  $p$  and number of passes  $P$ .

---

Find the pair-wise co-occurrence matrix  $C_0$  from the given image using Equation (1).

$k = 0$ .

**repeat**

    Set all entries of  $C_{k+1}$  to 0.

**for**  $\xi \in \mathbb{D}_\mathcal{I}$  **do**

$C_{k+1}(\xi^0) \leftarrow C_{k+1}(\xi^0) + (1 - p)C_k(\xi)$ .

$C_{k+1}(\xi^1) \leftarrow C_{k+1}(\xi^1) + pC_k(\xi)$ .

**end for**

**for**  $\xi \notin \mathbb{D}_\mathcal{I}$  **do**

$C_{k+1}(\xi^\phi) \leftarrow C_{k+1}(\xi^\phi) + C_k(\xi)$ .

**end for**

$k \leftarrow k + 1$

**until**  $k < P$

---

The co-occurrence matrix at the  $k^{th}$  stage can be used to estimate the embedding capacity for the  $(k + 1)^{th}$  pass. For example the embedding capacity of the first pass can be estimated from the initial co-occurrence matrix  $C_0$ . Thus we can write:

$$\eta(\mathcal{B}_k) = \sum_{\xi \in \mathbb{D}} C_k(\xi) \eta(b_\xi) = \sum_{\xi \in \mathbb{D}_{\mathcal{B}}} C_k(\xi). \quad (2)$$

It is clear that for a  $P$  pass watermarking it is sufficient to estimate the co-occurrence matrix upto the  $P - 1^{th}$  stage. Thus the total size of the bitstream  $\eta(\mathcal{B})$  can be estimated as:

$$\eta(\mathcal{B}) = \sum_{k=0}^{P-1} \eta(\mathcal{B}_k) = \sum_{k=0}^{P-1} \sum_{\xi \in \mathbb{D}_{\mathcal{B}}} C_k(\xi). \quad (3)$$

We can similarly find the number of ones in the bitstream  $\omega(\mathcal{B}_k)$ , by replacing  $\eta(b_\xi)$  by  $b_\xi$ . Thus we have:

$$\omega(\mathcal{B}_k) = \sum_{\xi \in \mathbb{D}} C_k(\xi) b_\xi = \sum_{\xi \in \mathbb{D}_{\mathcal{B}}^1} C_k(\xi). \quad (4)$$

In typical images the co-occurrence matrix is diagonally dominant with only about 10 % of the entries non-diagonal. Typically the disjoint set of pixel pairs are chosen as the neighbouring pairs of pixels and hence both the pixels are very similar in magnitude and correspondingly the possible set of pixel pairs is actually a small fraction of the total size. Hence one may use the standard sparse matrix representations [9, 17] to make our algorithms computationally more efficient.

Note that the co-occurrence based method is also an iterative procedure similar to any watermarking scheme. It is however computationally much more efficient than any stage-wise watermarking scheme for the following reasons: 1) The size of the co-occurrence matrix ( $256 \times 256$ ) is smaller than that of any typical image and hence the computation is much lesser. 2) A large overhead is associated with actual embedding due to the many iterations involved in calculating and embedding the auxiliary data, along with the required compression algorithm. 3) The sparse matrix representation of the co-occurrence matrix further improves the computation. In fact it is many times faster than actual embedding, as we show in the timing analysis in the results.

### 3.1.2 Pixel Pair Tree based method

**Definition 2.** (*Pixel-pair tree*). The pixel-pair tree for a pair  $\xi$  is defined as a tree which starts with the pixel pair  $\xi$  and traces a specific path based on the subsequent embedded bits as this pixel pair evolves and represents all feasible paths of watermarking for a given number of passes.

A pixel pair tree for a pixel pair (10, 12) embedded using the watermarking scheme of Coltuc [6], is shown in fig. 4. This tree concisely represents the entire life cycle of a particular pixel pair upto  $P$  passes. We begin by first defining the notation we will use for the pixel pair tree. Let  $\mathbf{S}_\xi$  represent the set of all paths for a given pixel pair. Note that we are assuming here that this is a  $P$ -stage watermarking. For example in fig. 4 the possible paths for the pair (10,12) upto  $P = 4$  passes are  $\{(10,12), (9,14), (5,18), (4,18), (4,18)\}$ ,  $\{(10,12), (9,14), (5,19), (4,19), ((4,19))\}$  and so on. Let  $s$  denote a particular path in  $\mathbf{S}_\xi$  and let  $s[k]$  denote the pixel pair at the  $k^{th}$  stage of embedding in the path  $s$ . Again, consider the path  $s = \{(10, 12), (9, 14), (5, 18), (4, 18), (4, 18)\}$ , then we have  $s[0]$  as

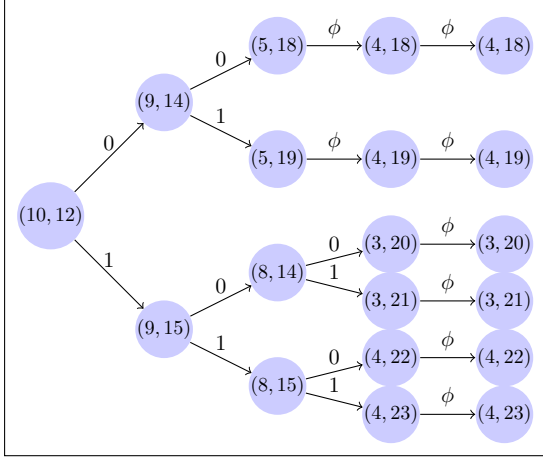


Figure 4: An illustration of the pixel-pair tree for the pixel pair (10, 12) for Coltuc's method [6]. A similar tree can be constructed for any block based watermarking scheme.

the pixel pair (10,12),  $s[1]$  as (9,14),  $s[2]$  as (5,18) and so on. Similarly,  $i_{s[0]}$ ,  $i_{s[1]}$  and  $i_{s[2]}$  are 0,0 and  $\phi$  respectively and  $\eta(i_{s[0]})$ ,  $\eta(i_{s[1]})$  and  $\eta(i_{s[2]})$  are 1,1 and 0 respectively. Further let  $s_j$  denote a path in the pixel pair tree  $\mathbf{S}_{\xi_j}$  (the pixel pair tree for the pair  $\xi_j$ ). Then we define  $\mathbf{s} = \{s_1, s_2, \dots, s_N\}$  as a particular path configuration chosen for the pixel pairs  $\Xi = \{\xi_1, \xi_2, \dots, \xi_N\}$  and let  $\mathbb{S}$  represent the set of all possible path configurations  $\mathbf{s}$ . Let  $p_s$  denote the path specific probability of the path  $s$ . It represents the probability that the pixel pair will evolve through the specific path  $s$ . Then clearly, we can write  $p_s = \prod_{k=0}^{P-1} p_{s[k]}$ , where  $p_{s[k]}$  denotes the probability of transition from  $s[k]$  to  $s[k+1]$  given whether  $s[k]$  is embeddable. Thus

$$p_{s[k]} = \begin{cases} p, & \text{if } i_{s[k]} = 1 \text{ \& } s[k] \in \mathbb{D}_{\mathcal{I}} \\ 1 - p, & \text{if } i_{s[k]} = 0 \text{ \& } s[k] \in \mathbb{D}_{\mathcal{I}} \\ 1, & \text{if } s[k] \notin \mathbb{D}_{\mathcal{I}} \end{cases} \quad (5)$$

Let  $\eta(\mathbf{s}, \mathcal{B})$  represent the size of the bitstream obtained through a specific path configuration  $\mathbf{s} = \{s_1, s_2, \dots, s_N\}$ . Further let  $\eta(\xi, s, \mathcal{B})$  denote the total number of bits contributed by a specific pixel pair  $\xi$  through the path  $s$ . In other words  $\eta(\xi, s, \mathcal{B}) = \sum_{k=0}^{P-1} \eta(b_{s[k]})$  and  $\eta(\mathbf{s}, \mathcal{B}) = \sum_{j=1}^N \eta(\xi_j, s_j, \mathcal{B})$ .

The estimate of the length of the bitstream  $\mathcal{B}$  can then be given as  $\eta(\mathcal{B}) = \mathbf{E}_{\mathbf{s} \in \mathbb{S}}(\eta(\mathbf{s}, \mathcal{B}))$ , where  $\mathbf{E}_{\mathbf{s} \in \mathbb{S}}(\eta(\mathbf{s}, \mathcal{B}))$  represents the total expected size of  $\mathcal{B}$  by considering every possible path configuration  $s_1 \in \mathbf{S}_{\xi_1}$ ,  $s_2 \in \mathbf{S}_{\xi_2}$ ,  $\dots$ ,  $s_N \in \mathbf{S}_{\xi_N}$ , which we can also write as  $\mathbf{s} \in \mathbb{S}$ . We then write this expectations  $\mathbf{E}_{\mathbf{s} \in \mathbb{S}}(\eta(\mathbf{s}, \mathcal{B}))$ , as the sum of the expected number of bits ( $\mathbf{E}_{s \in \mathbf{S}_{\xi}}(\eta(\xi, s, \mathcal{B}))$ ) contributed by every pixel pair  $\xi$ . Since we have considered here a  $P$  stage watermarking, for simplicity of notation, we

represent this expectation as  $\mathbf{E}_P^\eta(\xi, \mathcal{B})$ . Thus we can write:

$$\begin{aligned}
\eta(\mathcal{B}) &= \mathbf{E}_{\mathbf{s} \in \mathcal{S}}(\eta(\mathbf{s}, \mathcal{B})) = \sum_{j=1}^N \mathbf{E}_{s_j \in \mathcal{S}_{\xi_j}}(\eta(\xi_j, s_j, \mathcal{B})) \\
&= \sum_{j=1}^N \mathbf{E}_P^\eta(\xi_j, \mathcal{B}) = \sum_{j=1}^N \sum_{s_j \in \mathcal{S}_{\xi_j}} p_{s_j} \eta(\xi_j, s_j, \mathcal{B}) \\
&= \sum_{j=1}^N \sum_{s_j \in \mathcal{S}_{\xi_j}} \left( \prod_{k=0}^{P-1} p_{s_j[k]} \right) \sum_{k=0}^{P-1} \eta(b_{s_j[k]}) \tag{6}
\end{aligned}$$

In a similar manner we can also find the size of the bitstream  $\eta(\mathcal{B}_k)$  as an expectation over every possible path in the pixel pair tree. We use the notations similar to our above derivations, with each quantity now representing its stage-wise equivalent. However note that here  $\eta(\xi, s, \mathcal{B}_k) = \eta(b_{s[k]})$ . Further since we are interested in estimating the size of  $\mathcal{B}_k$ , the pixel pair tree extends only till the  $k^{\text{th}}$  stage and hence  $p_s = \prod_{m=0}^k p_{s[m]}$ . Again for convenience we use  $\mathbf{E}^\eta(\xi, \mathcal{B}_k)$  instead of  $\mathbf{E}_{\mathbf{s} \in \mathcal{S}_\xi}(\eta(\xi, s, \mathcal{B}_k))$  and we can write:

$$\begin{aligned}
\eta(\mathcal{B}_k) &= \mathbf{E}_{\mathbf{s} \in \mathcal{S}}(\eta(\mathbf{s}, \mathcal{B}_k)) = \sum_{j=1}^N \mathbf{E}_{s_j \in \mathcal{S}_{\xi_j}}(\eta(\xi_j, s_j, \mathcal{B}_k)) \\
&= \sum_{j=1}^N \mathbf{E}^\eta(\xi_j, \mathcal{B}_k) = \sum_{j=1}^N \sum_{s_j \in \mathcal{S}_{\xi_j}} \eta(\xi_j, s_j, \mathcal{B}_k) p_{s_j} \\
&= \sum_{j=1}^N \sum_{s_j \in \mathcal{S}_{\xi_j}} \eta(b_{s_j[k]}) \left( \prod_{m=0}^k p_{s_j[m]} \right) \tag{7}
\end{aligned}$$

Thus we summarize the expressions for  $\mathbf{E}_P^\eta(\xi, \mathcal{B})$  and  $\mathbf{E}^\eta(\xi, \mathcal{B}_k)$  as:

$$\mathbf{E}_P^\eta(\xi, \mathcal{B}) = \sum_{s \in \mathcal{S}_\xi} \left( \prod_{k=0}^{P-1} p_{s[k]} \right) \sum_{k=0}^{P-1} \eta(b_{s[k]}) \tag{8}$$

$$\mathbf{E}^\eta(\xi, \mathcal{B}_k) = \sum_{s \in \mathcal{S}_\xi} \eta(b_{s[k]}) \left( \prod_{m=0}^k p_{s[m]} \right) \tag{9}$$

We can then find the number of ones in the bitstreams  $\mathcal{B}_k$  and  $\mathcal{B}$ . This problem is very similar to that of finding the size of the bitstreams, since now instead of summing over  $\eta(b_{s[k]})$  we need to sum over the actual bits contributed  $b_{s[k]}$ . Hence we do not again reformulate the problem, but use the

same procedure above, by just replacing  $\eta(b_{s[k]})$  by  $b_{s[k]}$  and give the final results as:

$$\begin{aligned}\omega(\mathcal{B}) &= \sum_{j=1}^N \mathbf{E}_P^\omega(\xi_j, \mathcal{B}) = \sum_{j=1}^N \sum_{s_j \in \mathbf{S}_{\xi_j}} \prod_{k=0}^{P-1} p_{s_j[k]} \sum_{k=0}^{P-1} b_{s_j[k]} \\ \omega(\mathcal{B}_k) &= \sum_{j=1}^N \mathbf{E}^\omega(\xi_j, \mathcal{B}_k) = \sum_{j=1}^N \sum_{s_j \in \mathbf{S}_{\xi_j}} b_{s_j[k]} \prod_{m=0}^k p_{s_j[m]}\end{aligned}\quad (10)$$

As evident from above, the computation of the number of ones of a bitstream is exactly the same as that of computing the size of that bitstream except for replacing  $\eta(b_{s[k]})$  by  $b_{s[k]}$ . In the rest of the analysis we provide some important theorems and properties related to the total size of the bitstreams as well as the size of the bitstreams at every stage. These properties however also hold for the number of ones in the these bitstreams.

- It is easy to show that:  $\mathbf{E}_P^\eta(\xi, \mathcal{B}) = \sum_{k=0}^{P-1} \mathbf{E}^\eta(\xi, \mathcal{B}_k)$ .
- The formulations above are in terms of the image pixel pair (equations (7) and (6)), and we call them the image pixel-pair based formulation. We can alternatively also reformulate them in terms of the co-occurrence matrix. From the definition of the co-occurrence matrix, it is clear that we can write-

$$\eta(\mathcal{B}_k) = \sum_{\xi \in \mathbb{D}} C_0(\xi) \mathbf{E}^\eta(\xi, \mathcal{B}_k) \quad (11)$$

$$\eta(\mathcal{B}) = \sum_{\xi \in \mathbb{D}} C_0(\xi) \mathbf{E}_P^\eta(\xi, \mathcal{B}) \quad (12)$$

- The co-occurrence based algorithm and the tree based algorithm provide exactly the same estimates. In other words, the estimates provided by equations (2) and (3) are exactly the same as the ones provided in equation (7) and (6). We prove this formally in theorem- 3 (provided in Appendix-A).
- The formulation of  $\mathbf{E}_P^\eta(\xi, \mathcal{B})$  and  $\mathbf{E}^\eta(\xi, \mathcal{B}_k)$  requires an exhaustive search over every possible path making them exponential. Hence we need to provide a more efficient estimation mechanism. We observe that there exists a recursive relationship, which we describe in the following theorem. For simplicity we represent  $\eta(b_{s[0]})$  as  $\eta(b_\xi)$  since  $s[0] = \xi$ .

**Theorem 1.** Equation (9) can be reformulated in a recursive manner (for  $k > 0$ ) as:

$$\mathbf{E}^\eta(\xi, \mathcal{B}_k) = \begin{cases} (1-p) \mathbf{E}^\eta(\xi^0, \mathcal{B}_{k-1}) + p \mathbf{E}^\eta(\xi^1, \mathcal{B}_{k-1}), & \text{if } \xi \in \mathbb{D}_{\mathcal{I}} \\ \mathbf{E}^\eta(\xi^\phi, \mathcal{B}_{k-1}), & \text{if } \xi \notin \mathbb{D}_{\mathcal{I}} \end{cases} \quad (13)$$

with the base case of:  $\mathbf{E}^\eta(\xi, \mathcal{B}_0) = \eta(b_\xi)$ . Equation (8) can also be reformulated (for  $P > 1$ ) as:

$$\mathbf{E}_P^\eta(\xi, \mathcal{B}) = \begin{cases} \eta(b_\xi) + p \mathbf{E}_{P-1}^\eta(\xi^1, \mathcal{B}) + (1-p) \mathbf{E}_{P-1}^\eta(\xi^0, \mathcal{B}), & \text{if } \xi \in \mathbb{D}_{\mathcal{I}} \\ \eta(b_\xi) + \mathbf{E}_{P-1}^\eta(\xi^\phi, \mathcal{B}), & \text{if } \xi \notin \mathbb{D}_{\mathcal{I}} \end{cases} \quad (14)$$

with the base case:  $\mathbf{E}_1^\eta(\xi, \mathcal{B}) = \eta(b_\xi)$ .

*Proof.* The base cases can be derived from equations (9) and (8). We first prove it for the case of the stage-wise expectation. Then we consider for  $k > 0$ , the case where  $\xi$  is embeddable. We define  $\mathbf{S}_{\xi^0}$  and  $\mathbf{S}_{\xi^1}$  as the set of possible paths for the pixel pairs  $\xi^0$  and  $\xi^1$ , respectively in  $P - 1$  passes. Also let  $s^0, s^1$  represent a possible path in  $\mathbf{S}_{\xi^0}$  and  $\mathbf{S}_{\xi^1}$ , respectively. The main idea here is that we break up the set of paths  $\mathbf{S}_{\xi}$  into 2 groups, one of which starts with  $\xi$  and consists of the paths in  $\mathbf{S}_{\xi^0}$ , while the other also starts with  $\xi$  but consists of those in  $\mathbf{S}_{\xi^1}$ . Then we can rewrite equation (9) as:

$$\mathbf{E}^\eta(\xi, \mathcal{B}_k) = \sum_{s^1 \in \mathbf{S}_{\xi^1}} p \prod_{m=0}^{k-1} p_{s^1[m]} \eta(b_{s^1[k-1]}) + \sum_{s^0 \in \mathbf{S}_{\xi^0}} (1-p) \prod_{m=0}^{k-1} p_{s^0[m]} \eta(b_{s^0[k-1]}) \quad (15)$$

Note that the set of pixel pairs  $\forall s \in \mathbf{S}_{\xi}, s[k]$  are captured fully by the pixel pairs  $\forall s^1 \in \mathbf{S}_{\xi^1}, s^1[k-1]$  and  $\forall s^0 \in \mathbf{S}_{\xi^0}, s^0[k-1]$ . Also recognize that within each summation, we have  $\mathbf{E}^\eta(\xi^0, \mathcal{B}_{k-1})$  and  $\mathbf{E}^\eta(\xi^1, \mathcal{B}_{k-1})$  embedded and hence we can rewrite equation (15) as:

$$\mathbf{E}^\eta(\xi, \mathcal{B}_k) = p \mathbf{E}^\eta(\xi^0, \mathcal{B}_{k-1}) + (1-p) \mathbf{E}^\eta(\xi^1, \mathcal{B}_{k-1})$$

We can similarly prove this for the case when  $\xi$  is not embeddable.

We now consider the case for the total expectation of a given pixel pair. Again we use the same idea as above and rewrite equation (8) as:

$$\mathbf{E}_P^\eta(\xi, \mathcal{B}) = \sum_{s^1 \in \mathbf{S}_{\xi^1}} p \prod_{k=0}^{P-2} p_{s^1[k]} \left( \sum_{k=0}^{P-2} \eta(b_{s^1[k]}) + \eta(b_\xi) \right) + \sum_{s^0 \in \mathbf{S}_{\xi^0}} (1-p) \prod_{k=0}^{P-2} p_{s^0[k]} \left( \sum_{k=0}^{P-2} \eta(b_{s^0[k]}) + \eta(b_\xi) \right)$$

We observe that  $\eta(b_\xi)$  occurs in every term. Again we recognize that within each of the summations are  $\mathbf{E}_{P-1}^\eta(\xi^0, \mathcal{B})$  and  $\mathbf{E}_{P-1}^\eta(\xi^1, \mathcal{B})$ . Thus we can write:

$$\mathbf{E}_P^\eta(\xi, \mathcal{B}) = (1-p) \mathbf{E}_{P-1}^\eta(\xi^0, \mathcal{B}) + p \mathbf{E}_{P-1}^\eta(\xi^1, \mathcal{B}) + \eta(b_\xi) \left( p \sum_{s^1 \in \mathbf{S}_{\xi^1}} \prod_{k=0}^{P-2} p_{s^1[k]} + (1-p) \sum_{s^0 \in \mathbf{S}_{\xi^0}} \prod_{k=0}^{P-2} p_{s^0[k]} \right) \quad (16)$$

Since the paths in  $\mathbf{S}_{\xi^0}$  and  $\mathbf{S}_{\xi^1}$  form a complete subtree,  $\sum_{s^1 \in \mathbf{S}_{\xi^1}} \prod_{k=0}^{P-2} p_{s^1[k]} = 1$  and  $\sum_{s^0 \in \mathbf{S}_{\xi^0}} \prod_{k=0}^{P-2} p_{s^0[k]} = 1$ . Thus we observe that equation (16) transforms to equation (14) for the case when  $\xi$  is embeddable. We can similarly handle the case when  $\xi$  is not embeddable.  $\square$

Note that we can similarly reformulate  $\mathbf{E}^\omega(\xi, \mathcal{B}_k)$  and  $\mathbf{E}_P^\omega(\xi, \mathcal{B})$  recursively and the expression is similar to (13) and (14) except for replacing  $\eta(b_\xi)$  by  $b_\xi$ .

It is clear from equation (14) that  $\mathbf{E}_P^\eta(\xi, \mathcal{B})$  or the total expected number of bits for a given pixel pair is a polynomial in  $p$ . Hence in order to make the offline stage independent of  $p$ , we compute the polynomial coefficients of  $\mathbf{E}_P^\eta(\xi, \mathcal{B})$  for every pixel pair and store them. Thus we can directly use these stored coefficients to compute  $\mathbf{E}_P^\eta(\xi, \mathcal{B})$  for any given  $p$  without having to recompute them every time. In addition the set of coefficients is typically iteratively computed for many values of  $P$ . Hence we can use a simple 'memoization' [15] while computing the total expectation for a given pixel pair  $\mathbf{E}_P^\eta$  and using the previously computed values of  $\mathbf{E}_{P-1}^\eta$ , in equation (14). Thus the tree

formulation of equation (8) can be computed in linear time using these simple tricks. The same idea can be extended for computing the stage-wise expectation.

Thus there are two stages of computation, i.e the online stage and offline. The offline stage is image independent and we compute the coefficients of the polynomials, described in equation (14) and equation (13). These can be iteratively computed for various values of  $P$  in linear time. The online stage consists of just a single iteration to use the values of  $\mathbf{E}_P^\eta(\xi, \mathcal{B})$  or  $\mathbf{E}^\eta(\xi, \mathcal{B}_k)$  depending on what has to be estimated.

Thus this method provides an extremely efficient implementation to estimate both the size and the number of ones in a bitstream. Importantly it can compute this in a single iteration, although the embedding is multi-pass. Correspondingly, we can find the multi-pass embedding capacity, in a time complexity comparable to that of single pass estimation. Thus it is more efficient than the co-occurrence based estimation framework. The only drawback is that it requires an offline stage and additional memory to store the total or the stage-wise expectation for every possible pixel pair.

### 3.2 Compressed bit-streams

Compressed bitstreams sometimes occur as auxiliary data, either as a compressed location map or compressed flag bits. Correspondingly it is necessary to estimate the size of these compressed streams. Let  $\mathcal{B}^C$  be the bitstream obtained by compressing  $\mathcal{B}$ . Further let  $Cf(\cdot)$  represent the compression factor (a number between 0 and 1) for a bitstream and a given compression scheme. We can then find the size of the compressed bitstream as:

$$\eta(\mathcal{B}^C) = \sum_{k=0}^{P-1} \eta(\mathcal{B}_k) Cf(\mathcal{B}_k) \quad (17)$$

However, we cannot find the exact bitstreams at every pass  $\mathcal{B}_k$ , just using the bit probability, since they depend on the exact watermark sequence. Further we would have to actually embed the watermark in order to find the transformed pixel pairs and hence our statistical framework cannot be extended to this. However the Kolmogorov complexity of a binary bitstream of length  $n$ , with  $r$  number of ones has been shown [3] to be bounded by  $nH_0(\frac{r}{n})$ , where  $H_0(\cdot)$  refers to the entropy [20] of a bitstream. Correspondingly the compression of these bitstreams, can be estimated by just finding the number of ones  $\omega(\mathcal{B}_k)$  in these bitstreams at every stage. These estimates though approximate are still quite good and useful in practice. In the last section we proposed algorithms to estimate  $\omega(\mathcal{B}_k)$ . Hence we can estimate the size of the compressed bitstream at every stage  $\eta(\mathcal{B}_k^C)$  as well as the total compressed bitstream size  $\eta(\mathcal{B}^C)$  as:

$$\eta(\mathcal{B}_k^C) = \eta(\mathcal{B}_k) H_0\left(\frac{\omega(\mathcal{B}_k)}{\eta(\mathcal{B}_k)}\right) \quad (18)$$

$$\eta(\mathcal{B}^C) = \sum_{k=0}^{P-1} \eta(\mathcal{B}_k) H_0\left(\frac{\omega(\mathcal{B}_k)}{\eta(\mathcal{B}_k)}\right). \quad (19)$$

### 3.3 Embedding Capacity Estimation

Using the tools we have developed in the earlier sections, we are now in a position to estimate the embedding capacity. In particular, we use the co-occurrence or the tree based algorithm to find the



total number of embeddable pixel pairs ( $\eta(\mathcal{I})$ ) and the auxiliary data size. Hence we can find the total and the stage-wise embedding capacity as:

$$\eta(\mathcal{E}_k) = \eta(\mathcal{I}_k) - \eta(\mathcal{F}_k) - \eta(\mathcal{F}_k^C) - \eta(\mathcal{L}_k^C) \quad (20)$$

$$\eta(\mathcal{E}) = \eta(\mathcal{I}) - \eta(\mathcal{F}) - \eta(\mathcal{F}^C) - \eta(\mathcal{L}^C) \quad (21)$$

Note that generally all these types of auxiliary data will not occur together in any watermarking scheme and hence one or more of these will be null streams. We give below estimates of each of  $\eta(\mathcal{I}_k)$ ,  $\eta(\mathcal{F}_k)$ ,  $\eta(\mathcal{F}_k^C)$  and  $\eta(\mathcal{L}_k^C)$  in terms of the co-occurrence and tree based algorithms.

$$\eta(\mathcal{I}_k) = \sum_{\xi \in \mathbb{D}_{\mathcal{I}}} C_k(\xi) = \sum_{j=1}^N \mathbf{E}^\eta(\xi_j, \mathcal{I}_k) \quad (22)$$

$$\eta(\mathcal{F}_k) = \sum_{\xi \in \mathbb{D}_{\mathcal{F}}} C_k(\xi) = \sum_{j=1}^N \mathbf{E}^\eta(\xi_j, \mathcal{F}_k) \quad (23)$$

Further we can find the estimates of the number of ones in  $\mathcal{F}$  and  $\mathcal{L}$  as:

$$\omega(\mathcal{F}_k) = \sum_{\xi \in \mathbb{D}_{\mathcal{F}}} C_k(\xi) = \sum_{j=1}^N \mathbf{E}^\omega(\xi_j, \mathcal{F}_k) \quad (24)$$

$$\omega(\mathcal{L}_k) = \sum_{\xi \in \mathbb{D}_{\mathcal{L}}} C_k(\xi) = \sum_{j=1}^N \mathbf{E}^\omega(\xi_j, \mathcal{L}_k) \quad (25)$$

Finally we can find  $\eta(\mathcal{F}_k^C)$  and  $\eta(\mathcal{L}_k^C)$  as:

$$\eta(\mathcal{F}_k^C) = \eta(\mathcal{F}_k) H_0\left(\frac{\omega(\mathcal{F}_k)}{\eta(\mathcal{F}_k)}\right), \quad \eta(\mathcal{L}_k^C) = N H_0\left(\frac{\omega(\mathcal{L}_k)}{N}\right) \quad (26)$$

We have  $\eta(\mathcal{L}_k) = N$ , since the size of the location map is exactly that of the number of pixel pairs. We can then find  $\eta(\mathcal{I})$ ,  $\eta(\mathcal{F})$ ,  $\eta(\mathcal{F}^C)$  and  $\eta(\mathcal{L}^C)$  by summing each of the above obtained expressions from  $k = 1$  to  $P - 1$ . The tree based estimates for  $\eta(\mathcal{I})$  and  $\eta(\mathcal{F})$  can directly be obtained however, by just replacing the stage-wise expectation  $\mathbf{E}^\eta$  by the total expectation  $\mathbf{E}_P^\eta$ .

### 3.4 Change in probability of ones due to auxiliary data stream

One subtle point worth mentioning here is that the embedded bitstream contains both the auxiliary data as well as the watermark. We have until now represented  $p$  as the probability of the bit '1' in the embedded bit stream. Let  $p_W$  represent the probability of the bit '1' in the watermark. Then  $p \approx p_W$  since the watermark is the major component of the embedded bitstream atleast for the initial passes. However, since a few watermarking schemes (for example Coltuc's method) significantly depend on the probability of the embedded bits, it may be necessary to find at every stage the probability of the bit '1' in the embedded bitstream. For this we need to estimate the probability of '1' in the auxiliary bitstream, which we denote by  $p_A$ . In order to estimate  $p_A$ , we

need to estimate the number of ones in the auxiliary data stream. In the case where the auxiliary data is represented as compressed bit streams, we can assume that the compressed bitstream will be random, and the number of ones and zeros are the same. Hence we can assume the probability of the bits contributed by these compressed bitstreams is 0.5. The number of ones in the flag bit stream can be computed, as  $\omega(\mathcal{F}_k)$ , and hence the probability  $p_A$  can be found as:

$$p_{A_k} = \frac{\omega(\mathcal{F}_k) + 0.5(\eta(\mathcal{L}_k^C) + \eta(\mathcal{F}_k^C))}{\eta(\mathcal{F}_k) + \eta(\mathcal{L}_k^C) + \eta(\mathcal{F}_k^C)} \quad (27)$$

We can then easily modify algorithm-1, and use  $p = p_k$  at every stage of the algorithm, where  $p_k$  can be defined as:

$$p_k = \frac{(\eta(\mathcal{I}_k) - \eta(\mathcal{A}_k)) \cdot p_W + \eta(\mathcal{A}_k) \cdot p_{A_k}}{\eta(\mathcal{I}_k)} \quad (28)$$

Here  $\mathcal{A}$  contains combinations from the flag bit stream, compressed location map and compressed flag stream. We call this method the Co-occurrence based adaptive probability (CAP) algorithm. Using the co-occurrence matrix at the  $k^{th}$  stage  $C_k$ , we first estimate  $\eta(\mathcal{I}_k)$  and  $\eta(\mathcal{A}_k)$  at every iteration using techniques discussed in section 3.3. These estimates are then used in equations (27), (28) to update the probabilities  $p_k$ . Finally using the updated probabilities  $p_k$  and methods given in algorithm-I, we estimate  $C_{k+1}$ . Thus we can modify the iterative co-occurrence based method to consider weighted probabilities at every iteration. The tree based implementation, however, inherently considers a single probability and is not amenable to a change in probability at every iteration and hence cannot be used to provide accurate estimates. However this change is necessary only for those schemes which significantly depend on the probability of the watermark. Further even for such schemes, the estimates obtained by considering only the watermark probabilities are observed to be practically very useful in most cases, without the need to consider the updated probabilities  $p_k$ . We show in results that the error magnitude is quite small and we achieve descent approximations by only considering  $p \approx p_W$ . Thus unless extremely accurate estimates are required, we may continue to use the tree based algorithm in section 3.1.2 to achieve quick estimates.

### 3.5 Estimating the optimal number of passes

The methods we have discussed upto this point have concentrated on providing estimates of the embedding capacity for a given number of passes  $P$ . We can however find the overall embedding capacity and the optimal number of passes as well. For typical watermarking schemes the stage-wise embedding capacity keeps on decreasing. This is due to an increase in the number of pixel pairs not eligible for embedding at every successive pass of watermarking. Further at every stage the overhead in the form of the size of the auxiliary data also generally increases. Thus the optimal number of passes is reached when the stage-wise embedding capacity  $\mathcal{E}_k$  touches zero.

### 3.6 Generalization to watermarking schemes operating on groups of pixels

We have considered upto now watermarking schemes operating on pairs of pixels only, and hence we have assumed  $\xi = (x, y)$  in our algorithms. However our methods can be extended to consider  $n$ -tuple of pixels  $\xi = (x^1, x^2, \dots, x^n)$ . Consider the case of  $n = 3$  or triplets of pixels [1]. Here in every triplet 2 bits can be embedded, so we need to consider 4 cases of embedding 00, 01, 10 and

11. Let the transformed pixel triplets be  $\xi^{00}, \xi^{01}, \xi^{10}$  and  $\xi^{11}$ . If a pixel triplet is not embeddable it will transform to  $\xi^\phi$ . Then we can easily extend the co-occurrence based method, by considering here that  $(1-p)^2, p(1-p), p(1-p)$  and  $p^2$  fraction of the pixel triplets of  $C_k(\xi)$  transform to  $C_{k+1}(\xi^{00}), C_{k+1}(\xi^{01}), C_{k+1}(\xi^{10})$  and  $C_{k+1}(\xi^{11})$  respectively. The tree based method can similarly be estimated to consider a tree of groups of pixels instead of pixel pairs. The tree will have more branches. For example in the case of pixel triplets, each embeddable pixel triplet will have 4 possible paths to  $\xi^{00}, \xi^{01}, \xi^{10}$  and  $\xi^{11}$  respectively. We can however define a recursive formulation similar to equation (14) only replacing the equation for the embeddable case as:  $\mathbf{E}_P^\eta(\xi, \mathcal{B}) = \eta(b_\xi) + p^2 \mathbf{E}_{P-1}^\eta(\xi^{11}, \mathcal{B}) + p(1-p)(\mathbf{E}_{P-1}^\eta(\xi^{01}, \mathcal{B}) + \mathbf{E}_{P-1}^\eta(\xi^{10}, \mathcal{B})) + (1-p)^2 \mathbf{E}_{P-1}^\eta(\xi^{00}, \mathcal{B})$ . Our methods can similarly be extended to deal with any arbitrary  $n$ , where-in every embeddable pixel  $n$ -tuple can get converted to  $2^{n-1}$  possible pixel  $n$ -tuples since within every such tuple  $n-1$  bits can be embedded.

## 4 Bounds on the Embedding Capacity

In this section we provide the bounds on the embedding capacity. The maximum embedding capacity of a given image can be thought of as the largest possible watermark which can be embedded into that image. In other words no watermark with a size is larger than this, can be embedded completely into that image. We use the same tree based implementation discussed in section 3.1.2 to find the maximum possible embedding capacity, by considering every possible type of watermark. We denote the largest size as  $\eta^{\max}(\cdot)$ . Then using the notation developed in section 3.1.2, we can write:

$$\eta^{\max}(\mathcal{E}) = \max_{\mathbf{s} \in \mathbb{S}} \left[ \eta(\mathcal{I}, \mathbf{s}) - \eta(\mathcal{F}, \mathbf{s}) - \eta(\mathcal{L}^C, \mathbf{s}) - \eta(\mathcal{F}^C, \mathbf{s}) \right] \quad (29)$$

Recall that  $\mathbf{s} \in \mathbb{S}$  here represents  $s_1 \in \mathbf{S}_{\xi_1}, s_2 \in \mathbf{S}_{\xi_2}, \dots, s_N \in \mathbf{S}_{\xi_N}$ . Note that though not shown,  $\eta^{\max}(\mathcal{E})$  is a function of  $P$ , and it represents the maximum possible embedding capacity obtained in  $P$  passes. In order to find the maximum possible embedding capacity in the image, we need to find  $\max_P \eta^{\max}(\mathcal{E})$ . As discussed in section 3.5, the total embedding capacity initially increases till the optimal number of passes after which it begins to decrease. Hence we can simply start from  $P = 0$ , till the value of  $P$  where  $\eta^{\max}(\mathcal{E})$  begins to decrease. The corresponding value of  $\eta^{\max}(\mathcal{E})$  will be the maximum possible embedding capacity for that image.

**Theorem 2.** *The maximum possible embedding capacity  $\eta^{\max}(\mathcal{E})$  upto  $P$  passes can be upper bounded by:*

$$\eta^{\max}(\mathcal{E}) \leq \sum_{j=1}^N (\mathbf{M}_P^\eta(\xi_j, \mathcal{I}) - \mathbf{M}_P^\eta(\xi_j, \mathcal{F})) - \sum_{k=0}^{P-1} N H_0 \left( \frac{\sum_{j=1}^N \mathbf{M}^\omega(\xi_j, \mathcal{L}_k)}{N} \right) - \sum_{k=0}^{P-1} \left( \sum_{j=1}^N \mathbf{L}^\eta(\xi_j, \mathcal{F}_k) \right) \min(\alpha_k, \beta_k) \quad (30)$$

with:

$$\alpha_k = H_0 \left( \frac{\sum_{j=1}^N \mathbf{M}^\omega(\xi_j, \mathcal{F}_k)}{\sum_{j=1}^N \mathbf{M}^\omega(\xi_j, \mathcal{F}_k) + \mathbf{L}^\omega(\xi_j, \bar{\mathcal{F}}_k)} \right), \quad \beta_k = H_0 \left( \frac{\sum_{j=1}^N \mathbf{L}^\omega(\xi_j, \mathcal{F}_k)}{\sum_{j=1}^N \mathbf{L}^\omega(\xi_j, \mathcal{F}_k) + \mathbf{M}^\omega(\xi_j, \bar{\mathcal{F}}_k)} \right) \quad (31)$$

and for a general bitstream  $\mathcal{B}$ :

$$\mathbf{M}_P^\eta(\xi, \mathcal{B}) = \max_{\mathbf{s} \in \mathbf{S}_\xi} \sum_{k=0}^{P-1} \eta(b_{s[k]}), \quad \mathbf{M}^\omega(\xi, \mathcal{B}_k) = \max_{\mathbf{s} \in \mathbf{S}_\xi} b_{s[k]} \quad (32)$$

$$\mathbf{L}^\eta(\xi, \mathcal{B}_k) = \min_{\mathbf{s} \in \mathbf{S}_\xi} \eta(b_{s[k]}), \quad \mathbf{L}^\omega(\xi, \mathcal{B}_k) = \min_{\mathbf{s} \in \mathbf{S}_\xi} b_{s[k]}. \quad (33)$$

Further the bitstream  $\bar{\mathcal{F}}_k$  is formed by inverting every bit in  $\mathcal{F}_k$ .

*Proof.* We start with equation (29) and slowly relax the constraints as follows:

$$\begin{aligned} \eta^{\max}(\mathcal{E}) &= \max_{\mathbf{s} \in \mathbb{S}} \left[ \eta(\mathcal{I}, \mathbf{s}) - \eta(\mathcal{F}, \mathbf{s}) - \eta(\mathcal{L}^C, \mathbf{s}) - \eta(\mathcal{F}^C, \mathbf{s}) \right] \\ &\leq \max_{\mathbf{s} \in \mathbb{S}} \left[ \eta(\mathcal{I}, \mathbf{s}) - \eta(\mathcal{F}, \mathbf{s}) \right] - \min_{\mathbf{s} \in \mathbb{S}} \eta(\mathcal{L}^C, \mathbf{s}) - \min_{\mathbf{s} \in \mathbb{S}} \eta(\mathcal{F}^C, \mathbf{s}). \end{aligned}$$

We now consider the first two terms in the above expression. We can expand it as follows:

$$\begin{aligned} \max_{\mathbf{s} \in \mathbb{S}} \left[ \eta(\mathcal{I}, \mathbf{s}) - \eta(\mathcal{F}, \mathbf{s}) \right] &= \max_{\mathbf{s} \in \mathbb{S}} \sum_{j=1}^N \left[ \eta(\xi_j, s_j, \mathcal{I}) - \eta(\xi_j, s_j, \mathcal{F}) \right] \\ &= \sum_{j=1}^N \max_{s_j \in \mathbf{S}_{\xi_j}} \left[ \eta(\xi_j, s_j, \mathcal{I}) - \eta(\xi_j, s_j, \mathcal{F}) \right] = \sum_{j=1}^N \max_{s_j \in \mathbf{S}_{\xi_j}} \left[ \sum_{k=0}^{P-1} \eta(i_{s[k]}) - \sum_{k=0}^{P-1} \eta(f_{s[k]}) \right] \\ &= \sum_{j=1}^N (\mathbf{M}_P^\eta(\xi_j, \mathcal{I}) - \mathbf{M}_P^\eta(\xi_j, \mathcal{F})). \end{aligned} \quad (34)$$

Now we consider the term involving the compressed location map. Since this term appears with a negative sign, finding an upper bound on  $\eta^{\max}(\mathcal{E})$  is equivalent to finding a lower bound of this term. Hence we first replace the compression term by the entropy of the location map bitstream at every stage, which is a lower bound to the size of the compressed bitstream. Hence we can write:

$$\begin{aligned} \min_{\mathbf{s} \in \mathbb{S}} \eta(\mathcal{L}^C, \mathbf{s}) &\geq \min_{\mathbf{s} \in \mathbb{S}} \sum_{k=0}^{P-1} NH_0\left(\frac{\omega(\mathcal{L}_k, \mathbf{s})}{N}\right) \geq \min_{\mathbf{s} \in \mathbb{S}} \sum_{k=0}^{P-1} NH_0\left(\frac{\sum_{j=1}^N l_{s_j[k]}}{N}\right) \\ &\geq \sum_{k=0}^{P-1} \min_{\mathbf{s} \in \mathbb{S}} NH_0\left(\frac{\sum_{j=1}^N l_{s_j[k]}}{N}\right). \end{aligned}$$

We then further simplify the above expression, using the lemma 1 (Provided in Appendix-B). Observe that  $\mathcal{L}_k$  is typically dominated by ones initially since most of the pixel pairs are embeddable. Further compression is only possible till  $\omega(\mathcal{L}_k, \mathbf{s}) \geq N/2$  since, as the fraction of the number of ones reaches 0.5, the compression factor reaches 1 and hence embedding is no longer possible. Thus

$\frac{\omega(\mathcal{L}_k, \mathbf{s})}{N} > 0.5, \forall k$  and hence using lemma 1 directly with  $f(\mathbf{s}) = \frac{\omega(\mathcal{L}_k, \mathbf{s})}{N}$  we have:

$$\begin{aligned} \min_{\mathbf{s} \in \mathbb{S}} \eta(\mathcal{L}^C, \mathbf{s}) &\geq \sum_{k=0}^{P-1} NH_0\left(\frac{\max_{\mathbf{s} \in \mathbb{S}} \sum_{j=1}^N l_{s_j[k]}}{N}\right) \geq \sum_{k=0}^{P-1} NH_0\left(\frac{\sum_{j=1}^N \max_{s_j \in \mathbb{S}_{\xi_j}} l_{s_j[k]}}{N}\right) \\ &\geq \sum_{k=0}^{P-1} NH_0\left(\frac{\sum_{j=1}^N \mathbf{M}^\omega(\xi_j, \mathcal{L}_k)}{N}\right) \end{aligned} \quad (35)$$

We now finally consider the term involving the compressed flag bitstream. Again similar to the compressed location map size estimation, we can write:

$$\begin{aligned} \min_{\mathbf{s} \in \mathbb{S}} \eta(\mathcal{F}^C, \mathbf{s}) &\geq \min_{\mathbf{s} \in \mathbb{S}} \sum_{k=0}^{P-1} \eta(\mathcal{F}_k, \mathbf{s}) H_0\left(\frac{\omega(\mathcal{F}_k, \mathbf{s})}{\eta(\mathcal{F}_k, \mathbf{s})}\right) \\ &\geq \sum_{k=0}^{P-1} \min_{\mathbf{s} \in \mathbb{S}} \eta(\mathcal{F}_k, \mathbf{s}) H_0\left(\frac{\omega(\mathcal{F}_k, \mathbf{s})}{\eta(\mathcal{F}_k, \mathbf{s})}\right) \\ &\geq \sum_{k=0}^{P-1} \min_{\mathbf{s} \in \mathbb{S}} \eta(\mathcal{F}_k, \mathbf{s}) \min_{\mathbf{s} \in \mathbb{S}} H_0\left(\frac{\omega(\mathcal{F}_k, \mathbf{s})}{\eta(\mathcal{F}_k, \mathbf{s})}\right) \end{aligned}$$

We now consider each of the terms above separately. Consider the first term in the above expression.

$$\begin{aligned} \min_{\mathbf{s} \in \mathbb{S}} \eta(\mathcal{F}_k, \mathbf{s}) &= \min_{\mathbf{s} \in \mathbb{S}} \sum_{j=1}^N \eta(f_{s_j[k]}) \\ &= \sum_{j=1}^N \min_{s_j \in \mathbb{S}_{\xi_j}} \eta(f_{s_j[k]}) = \sum_{j=1}^N \mathbf{L}^\eta(\xi_j, \mathcal{F}_k) \end{aligned} \quad (36)$$

Now we consider the second term. Again we take the min term inside the entropy. However unlike the case of the compressed location map, we cannot comment on the value of  $\frac{\omega(\mathcal{F}_k, \mathbf{s})}{\eta(\mathcal{F}_k, \mathbf{s})}$ , whether it lies within  $[0.5, 1]$  or not. Hence we cannot use lemma-1 directly. However we use lemma 2 (Again provided in Appendix B), and define  $h(\mathbf{s}) = \frac{\omega(\mathcal{F}_k, \mathbf{s})}{\eta(\mathcal{F}_k, \mathbf{s})}$  to obtain:

$$\min_{\mathbf{s} \in \mathbb{S}} H_0\left(\frac{\omega(\mathcal{F}_k, \mathbf{s})}{\eta(\mathcal{F}_k, \mathbf{s})}\right) \geq \min\left(H_0\left(\max_{\mathbf{s} \in \mathbb{S}} \frac{\omega(\mathcal{F}_k, \mathbf{s})}{\eta(\mathcal{F}_k, \mathbf{s})}\right), H_0\left(\min_{\mathbf{s} \in \mathbb{S}} \frac{\omega(\mathcal{F}_k, \mathbf{s})}{\eta(\mathcal{F}_k, \mathbf{s})}\right)\right) \quad (37)$$

Consider the first term within the entropy. We use a trick here of defining  $f_{s[k]}^- = 1 - f_{s[k]}$ , if  $s[k] \in \mathbb{D}_{\mathcal{F}}$  and 0 otherwise. Then observe that  $\eta(f_{s_j[k]}) = f_{s_j[k]} + f_{s_j[k]}^-$ . Thus we can write:

$$\begin{aligned}
H_0\left(\max_{\mathbf{s} \in \mathbb{S}} \frac{\omega(\mathcal{F}_k, \mathbf{s})}{\eta(\mathcal{F}_k, \mathbf{s})}\right) &= H_0\left(\max_{\mathbf{s} \in \mathbb{S}} \frac{\sum_{j=1}^N f_{s_j[k]}}{\sum_{j=1}^N \eta(f_{s_j[k]})}\right) = H_0\left(\max_{\mathbf{s} \in \mathbb{S}} \frac{1}{1 + \frac{\sum_{j=1}^N \bar{f}_{s_j[k]}}{\sum_{j=1}^N f_{s_j[k]}}}\right) \\
&= H_0\left(\frac{1}{1 + \min_{\mathbf{s} \in \mathbb{S}} \frac{\sum_{j=1}^N \bar{f}_{s_j[k]}}{\sum_{j=1}^N f_{s_j[k]}}}\right) \geq H_0\left(\frac{1}{1 + \frac{\min_{\mathbf{s} \in \mathbb{S}} \sum_{j=1}^N \bar{f}_{s_j[k]}}{\max_{\mathbf{s} \in \mathbb{S}} \sum_{j=1}^N f_{s_j[k]}}}\right) \\
&\geq H_0\left(\frac{1}{1 + \frac{\sum_{j=1}^N \min_{s_j \in \mathbb{S}_{\xi_j}} \bar{f}_{s_j[k]}}{\sum_{j=1}^N \max_{s_j \in \mathbb{S}_{\xi_j}} f_{s_j[k]}}}\right) \geq H_0\left(\frac{1}{1 + \frac{\sum_{j=1}^N \mathbf{L}^\omega(\xi_j, \bar{\mathcal{F}}_k)}{\sum_{j=1}^N \mathbf{M}^\omega(\xi_j, \mathcal{F}_k)}}\right) \quad (38)
\end{aligned}$$

Similarly consider the second term and using derivations similar to above, we can obtain:

$$H_0\left(\min_{\mathbf{s} \in \mathbb{S}} \frac{\omega(\mathcal{F}_k, \mathbf{s})}{\eta(\mathcal{F}_k, \mathbf{s})}\right) \geq H_0\left(\frac{1}{1 + \frac{\sum_{j=1}^N \mathbf{M}^\omega(\xi_j, \bar{\mathcal{F}}_k)}{\sum_{j=1}^N \mathbf{L}^\omega(\xi_j, \mathcal{F}_k)}}\right) \quad (39)$$

Thus the theorem is proved by combining all the above equations.  $\square$

Estimation of the maximum embedding capacity, requires estimation of the quantities in equation (32). They however require exhaustive search over the pixel pair tree for every pixel pair, due to which the computation becomes extremely expensive. We can however recursively reformulate these expressions, similar to equations (13) and (14). Thus we can recursively compute  $\mathbf{M}_P^\eta(\xi, \mathcal{B})$  for  $P > 1$  as:

$$\mathbf{M}_P^\eta(\xi, \mathcal{B}) = \begin{cases} \eta(b_\xi) + \max(\mathbf{M}_{P-1}^\eta(\xi^0, \mathcal{B}), \mathbf{M}_{P-1}^\eta(\xi^1, \mathcal{B})), & \text{if } \xi \in \mathbb{D}_{\mathcal{I}} \\ \eta(b_\xi) + \mathbf{M}_{P-1}^\eta(\xi^\phi, \mathcal{B}), & \text{if } \xi \notin \mathbb{D}_{\mathcal{I}} \end{cases} \quad (40)$$

The base case here is  $\mathbf{M}_1^\eta(\xi, \mathcal{B}) = \eta(b_\xi)$ . Similarly we reformulate the expression for  $\mathbf{M}^\omega(\xi, \mathcal{B}_k)$  for  $k > 0$  as

$$\mathbf{M}^\omega(\xi, \mathcal{B}_k) = \begin{cases} \max(\mathbf{M}^\omega(\xi^0, \mathcal{B}_{k-1}), \mathbf{M}^\omega(\xi^1, \mathcal{B}_{k-1})), & \text{if } \xi \in \mathbb{D}_{\mathcal{I}} \\ \mathbf{M}^\omega(\xi^\phi, \mathcal{B}_{k-1}), & \text{if } \xi \notin \mathbb{D}_{\mathcal{I}} \end{cases} \quad (41)$$

Again the base case here is  $\mathbf{M}^\omega(\xi, \mathcal{B}_0) = b_\xi$ .

Note that the expression for  $\mathbf{L}^\omega(\xi, \mathcal{B}_k)$  is similar to equation (41) only replacing the  $\max(\cdot)$  by  $\min(\cdot)$ . The above expressions can easily be proved using methods similar to those used in theorem-1. Also we can use tricks like memoization etc. similar to those discussed in section-3.1.2, to efficiently estimate the maximum embedding capacity.

## 5 Results and Discussion

The presented concept has been experimented on several commonly known images as possible cover images. Images were chosen with varying gray level distributions. We however present results here only for Lena and Boat images for brevity (shown in fig. 5). A general analysis for several other

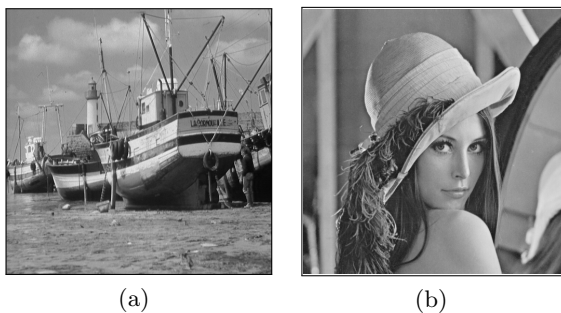


Figure 5: Above are the Boat (left) and Lena (right) images on which we have performed our analysis.

Table 1: The standard deviation of the total embedding capacity at the end of various watermarking stages (in  $10^{-2}$  bpp).

Algo	Stage-1	Stage-2	Stage-3
Coltuc	0.3	1.1	1.6
Tian	0.3	1.3	1.5

images has been provided as additional material<sup>1</sup>. We compute the embedding capacity for these cover images for watermarks of varying number of ones and zeros and compare them to the actual capacity as obtained through direct embedding of these watermarks in the corresponding cover image. We also find the bounds on the embedding capacity for these images. Lastly the codes used in the quantitative evaluation of our results shall be made available once our paper has been accepted.

### 5.1 Dependence of Embedding capacity on probability of watermark alone

In this section, we statistically verify our claim, that the multipass embedding capacity of pixel-pair based watermarking schemes depends only on the probability mass function of the embedding bitstream and not on the actual bitstream itself. In particular, we verify this for the Lena image, for the schemes of Coltuc [6] and Tian [11], and observe that the variance of the embedding capacity obtained by taking 100 different watermarks with  $p = 0.6$  is negligible. The values of the standard deviations of the bpp's at various stages of watermarking are shown in Table I.

### 5.2 Estimation and bounds of embedding capacity

We demonstrate in this section, how the embedding capacity estimates obtained using our algorithms, are reasonably close to those obtained by actual embedding a watermark, for the schemes of Tian [11] and Coltuc [6]. In particular we compare the capacities obtained by actually watermarking (AW) with the estimates obtained by the tree based algorithm (TA), along with the Co-occurrence matrix based adaptive probability algorithm (CAP) discussed in section 3.4. Finally

<sup>1</sup>Additional results are provided in '<http://www.ee.iitb.ac.in/~sc/main/misc/results.html>'

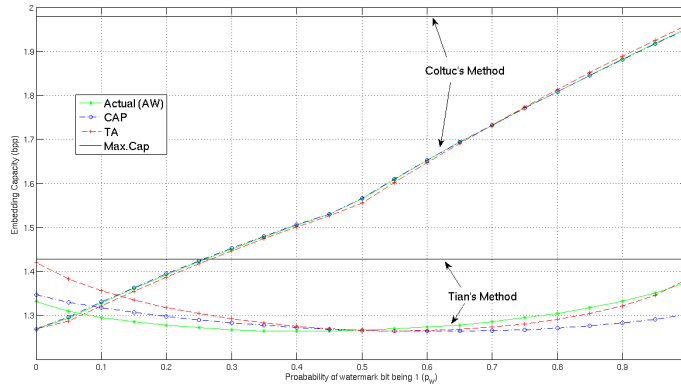


Figure 6: The capacity estimation for Lena image obtained through Tian’s and Coltuc’s schemes. For each of these watermarking schemes, the results of AW, CAP, TA and Maxcap are shown.

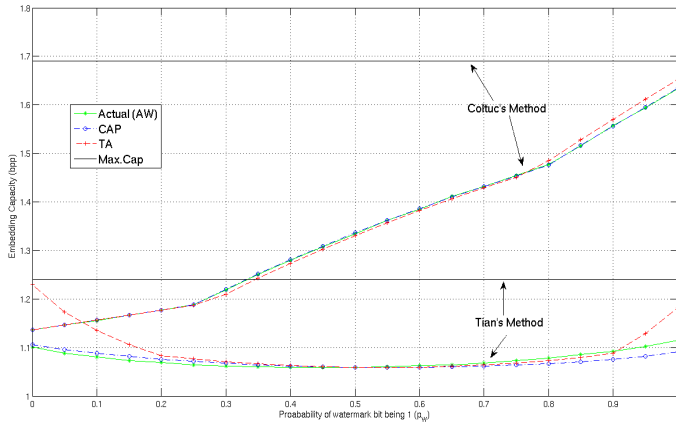


Figure 7: The results for Boat image obtained through Tian’s and Coltuc’s schemes.

we also compute the maximum possible embedding capacity (MaxCap), discussed in section IV. The CAP algorithm is precise since it considers weighted probabilities discussed in section 3.4, while TA, though not as accurate is extremely fast . We have computed here the optimal embedding capacity, as discussed in section 3.5. Lastly observe that we compute these for various watermarks, characterized by the fraction of the number of ones in the watermark ( $p_W$ ), and are plotted in fig. 6 and fig. 7.

### 5.2.1 Coltuc et al.

Coltuc’s algorithm does not require any location map, and requires flag bits to be embedded along with the watermark. Hence the estimates provided are very precise and accurate. In particular, the estimates provided by the CAP algorithm are extremely accurate and consistent with the actual capacity for all probabilities. Further the estimates provided by the tree based algorithm (TA) assuming  $p = p_W$ , are reasonably close as evident in fig. 6 and fig. 7.



Table 2: Timing analysis for different algorithms (time taken in seconds) for execution in MATLAB for a 3.2 GHz PC with 2 GB RAM.

Algo	Image Size	AW	CAP	TA
Coltuc	256 x 256	11.21	0.17	0.06
	512 x 512	31.32	0.35	0.19
	1024 x 1024	96.65	0.43	0.25
Tian	256 x 256	37.23	0.21	0.14
	512 x 512	99.32	0.55	0.31
	1024 x 1024	172.12	0.73	0.38

### 5.2.2 Tian

Tian’s algorithm requires a compression stage, and correspondingly we estimate the compression, using entropy based method discussed in section 3.2. In order to achieve higher embedding, typically the flag bits are also compressed. Further we assume that the bitstreams are compressed using arithmetic coding. As evident from fig. 6 and fig. 7, the estimates provided for Tian’s schemes are not as precise as the ones provided by coltuc, mainly due to our approximation of the compression of the flag and location map bit streams. Nevertheless, except for the extreme probabilities, the estimates for most watermark bit probabilities are quite precise and useful. Notwithstanding this, it is also evident that CAP estimates are better than those of the TA algorithm.

### 5.3 Comparative Analysis

The proposed algorithms not only provide useful estimates of embedding capacity but are also extremely fast. The CAP algorithm is a more general algorithm since it is easily amenable to updating probabilities at every stage, and takes account of the contribution of the auxiliary bits. The tree based algorithm is however not amenable to updating probabilities at every iteration, but is extremely fast. Further note that the timings given in Table-2 for actual watermarking (AW), are the ones obtained by the most efficient look-up-table based implementation of Coltuc and Tian’s algorithm. It is evident that our algorithms are computationally much more efficient than these low cost implementations. In particular, the tree based algorithm (TA) is extremely fast, nearly 200 to 400 times faster than actual watermarking, while the CAP algorithm, though iterative is also reasonably fast, and is about 100 times faster than the actual watermarking. The main reason for this is the computational efficiency gained through the sparse representation of the co-occurrence matrix and the simple computations of our algorithms, vis-a-vis the complexity of actual watermarking due to the numerous iterations of collecting, embedding and possibly compressing the auxiliary data along with the watermark. Also as evident from the timings, Tian’s algorithm is computationally expensive since it involves a data compression stage.

## 6 Conclusion

This is mainly a theoretical paper, centered around multipass embedding capacity estimation for mapping and expansion based algorithms. We provide very general purpose algorithms, which can be applied to any watermarking scheme, operating on blocks of pixels. In particular, we implemented a co-occurrence based and a tree based algorithm, to efficiently estimate the capacity, at computational costs substantially lesser than actually embedding the watermark. We further show in Appendix-A, that both of these algorithms provide the same estimates. From our results, it is evident that the tree based method is computationally more efficient compared to co-occurrence based one since it requires only a single iteration. The co-occurrence based method on the other hand is however a more general framework and can be extended to consider different watermark probabilities at every iteration and correspondingly can be used to provide accurate estimates by considering the probabilities of the auxiliary bits as well.

## 7 Acknowledgement

We would like to thank Subhasis Das, Siddhant Agrawal and Prof. Sibi-Raj Pillai from IIT Bombay for suggestions and discussions. Finally we gratefully acknowledge partial funding from Bharati Centre for Communication.

## References

- [1] A.M.Alattar. Reversible watermark using the difference expansion of a generalized integer transform. *IEEE Trans. Image Processing*, 13(8):1147–1156, August 2004.
- [2] R. Borse and S. Chaudhuri. Computation of embedding capacity in reversible watermarking schemes. *Proceedings of the ACM's ICVGIP-10, Chennai, India*, 2010.
- [3] T. Cover, J. Thomas, and MyiLibrary. *Elements of information theory*, volume 6. Wiley Online Library, 1991.
- [4] I. Cox. *Digital Watermarking and Steganography*. Morgan Kaufmann, Second Edition, New York, 2008.
- [5] D.Coltuc and J.-M. Chassery. High capacity reversible watermarking. *Proceedings of the IEEE International Conference on Image Processing ICIP*, pages 2565–2568, 2006.
- [6] D.Coltuc and J.-M. Chassery. Very fast watermarking by reversible contrast mapping. *IEEE Signal processing Letters*, 14(4):255–258, 2007.
- [7] D.M.Thodi and J.J.Rodriquez. Expansion embedding techniques for reversible watermarking. *IEEE Trans. Image Processing*, 16(3):721–730, 2007.
- [8] J. Feng, I. Lin, C. Tsai, and Y. Chu. Reversible watermarking: current status and key issues. *International Journal of Network Security*, 12(3):161–171, 2006.

- [9] G. Golub and C. Van Loan. Matrix computations 3rd edition. *The John Hopkins University, Baltimore*, 1996.
- [10] R. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Transactions on systems, man and cybernetics*, 3(6):610–621, 1973.
- [11] J.Tian. Reversible data embedding using a difference expansion. *IEEE Trans. Circuits Systems for Video Technology*, 13(8):890–896, August 2003.
- [12] T. Kalker and F. M.J.Willems. Capacity bounds and constructions for reversible data-hiding. *IEEE International Conference on DSP*, 2002.
- [13] C.-T. Li. Reversible watermarking scheme with image-independent embedding capacity. *IEE Trans on Vision and Image Processing*, 152(6)(779-786), 2005.
- [14] L.Kamastra and H.Heijmans. Reversible data embedding into images using wavelet techniques and sorting. *IEEE Trans. Image Processing*, 14(12):2082–2090, December 2005.
- [15] D. Michie. Memo functions and machine learning. *Nature*, 218(1):19–22, 1968.
- [16] M.U.Celik and G.Sharma. Lossless generalized-lsb data embedding. *IEEE Transactions on Image Processing*, 14(2):253–266, 2005.
- [17] S. Pissanetzky. *Sparse matrix technology*. Academic Press London, 1984.
- [18] C. Roberto, F. Francesco, and B. Rudy. Reversible watermarking techniques: An overview and a classification. *EURASIP Journal on Information Security*, 2010.
- [19] V. Sachnev. Reversible watermarking algorithm using sorting and prediction. *IEEE Trans. Circuits Systems for Video Technology*, 19(7):989–999, July 2009.
- [20] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(10):623–656.
- [21] C. D. Vleeschouwer, J. E. Delaigle, and B. Macq. Circular interpretation of histogram for reversible watermarking. *in Proceedings of the IEEE 4th Workshop on Multimedia Signal Processing*, pages 345–350, Oct 2001.
- [22] S. Weng, Y. Zhao, J. Pan, and R. Ni. Reversible watermarking based on invariability and adjustment on pixel pairs. *Signal Processing Letters, IEEE*, 15:721–724, 2008.

## A Appendix

**Theorem 3.** *Under multi-pass embedding, the embedding capacity estimated by the co-occurrence based algorithm (algorithm-1) is exactly the same as the one estimated by the pixel-pair tree based algorithm. In particular we prove that:*

$$1) \sum_{\xi \in \mathbb{D}} C_k(\xi) b_\xi = \sum_{\xi \in \mathbb{D}} C_0(\xi) \sum_{s \in \mathbf{S}_\xi} \left( \prod_{m=0}^k p_{s[m]} \right) b_{s[k]}$$

$$2) \sum_{k=0}^{P-1} \sum_{\xi \in \mathbb{D}} C_k(\xi) b_\xi = \sum_{\xi \in \mathbb{D}} C_0(\xi) \sum_{s \in \mathbf{S}_\xi} \left( \prod_{k=0}^{P-1} p_{s[k]} \right) \sum_{k=0}^{P-1} b_{s[k]}$$

*Proof.* For every pixel pair  $\xi$ , let  $\mathbf{N}_\xi$  represent the set of nodes corresponding to the pixel pair tree of  $\xi$ . Let  $n \in \mathbf{N}_\xi$  refer to a node, i.e a pixel pair in the pixel-pair tree of  $\xi$ . Further let  $s_n$  represent any path of the tree, containing the node  $n$  and let  $d_n$  represent the depth of node  $n$ . Hence  $s_n[d_n] = n$ . Now we prove the first part of the theorem. We start with the LHS and from algorithm 1, it is evident that for any node  $n \in \mathbf{N}_\xi$  with depth  $d_n = k$ , the number of pixel pairs contributed by the pair  $\xi$ , to the pixel pair  $n$  is  $C(\xi) \prod_{m=0}^k p_{s_n[m]}$ . This is clear from the updation step in algorithm 1. We then break down the sum  $\sum_{\xi \in \mathbb{D}} C_k(\xi) b_\xi$ , into contributions from every node  $n \in \mathbf{N}_\xi$  with depth  $d_n = k$ , from every pixel pair in  $\xi \in \mathbb{D}$ . In other words we can write:

$$\sum_{\xi \in \mathbb{D}} C_k(\xi) b_\xi = \sum_{\xi \in \mathbb{D}} \sum_{n \in \mathbf{N}_\xi: d_n=k} C_0(\xi) \prod_{m=0}^k p_{s_n[m]} b_{s_n[k]} \quad (42)$$

Now it is obvious from the definition that:  $\sum_{n \in \mathbf{N}_\xi: d_n=k} \prod_{m=0}^k p_{s_n[m]} b_{s_n[k]} = \sum_{s \in \mathbf{S}_\xi} \prod_{m=0}^k p_{s[m]} b_{s[k]}$ . Hence using this in equation (42) we can prove the first part of the theorem.

We now consider the second part. Using the result from earlier section we can write:

$$\begin{aligned} \sum_{k=0}^{P-1} \sum_{\xi \in \mathbb{D}} C_k(\xi) b_\xi &= \sum_{\xi \in \mathbb{D}} \sum_{k=0}^{P-1} \sum_{n \in \mathbf{N}_\xi: d_n=k} C_0(\xi) \prod_{m=0}^k p_{s_n[m]} b_{s_n[m]} \\ &= \sum_{\xi \in \mathbb{D}} \sum_{n \in \mathbf{N}_\xi} C_0(\xi) \prod_{k=0}^{d_n} p_{s_n[k]} b_{s_n[d_n]} \end{aligned}$$

In the above expression, observe that we are summing over every node in the pixel pair tree as  $k$  goes from 0 to  $P - 1$ . Hence we re-write the expression, by explicitly summing over every node in the pixel pair tree. We finally replace the variable  $m$  by  $k$ , to obtain the above expression. In order to prove the second part of the theorem, it is sufficient to show that for every pixel pair  $\xi$ :

$$\sum_{n \in \mathbf{N}_\xi} C_0(\xi) \cdot \prod_{k=0}^{d_n} p_{s_n[k]} b_n = \sum_{s \in \mathbf{S}_\xi} C_0(\xi) \prod_{k=0}^{P-1} p_{s[k]} \sum_{k=0}^{P-1} b_{s[k]}$$

Note that we used the fact that  $b_{s_n[d_n]} = b_n$ . We then start with the R.H.S as:

$$\begin{aligned} \text{R.H.S} &= \sum_{s \in \mathbf{S}_\xi} C_0(\xi) \prod_{k=0}^{P-1} p_{s[k]} \sum_{k=0}^{P-1} b_{s[k]} \\ &= \sum_{n \in \mathbf{N}_\xi} \sum_{s \in \mathbf{S}_\xi} C_0(\xi) \prod_{k=0}^{P-1} p_{s[k]} \sum_{k=0}^{P-1} b_{s[k]} I(s[k] = n) \\ &= \sum_{n \in \mathbf{N}_\xi} \sum_{s \in \mathbf{S}_\xi} C_0(\xi) \prod_{k=0}^{P-1} p_{s[k]} b_n I(s[k] = n) \end{aligned}$$

In the above equation, we consider only those paths which pass through the node  $n$ . Let  $\mathbf{S}_n$  be a subset of paths which pass through node  $n$ . Hence we can write:

$$\begin{aligned}
\text{R.H.S} &= \sum_{n \in \mathbf{N}_\xi} \sum_{s \in \mathbf{S}_n} C_0(\xi) \prod_{k=0}^{d_n} p_{s[k]} \prod_{k=d_n+1}^P p_{s[k]} b_n \\
&= \sum_{n \in \mathbf{N}_\xi} C_0(\xi) \prod_{k=0}^{d_n} p_{s_n[k]} b_n \sum_{s \in \mathbf{S}_n} \prod_{k=d_n+1}^P p_{s[k]} \\
&= \sum_{n \in \mathbf{N}_\xi} C_0(\xi) \prod_{k=0}^{d_n} p_{s_n[k]} b_n \\
&= \text{L.H.S}
\end{aligned}$$

Observe that the nodes in the paths  $s \in \mathbf{S}_n$  with depth greater than  $d_n$ , form a complete subtree and hence we have:  $\sum_{s \in \mathbf{S}_n} \prod_{k=d_n+1}^{P-1} p_{s[k]} = 1$ . Further also observe that the nodes with a depth  $\leq d_n$  occur in every path in  $s \in \mathbf{S}_n$ . Hence we replace these nodes by  $s_n[k]$ , since  $s_n$  is a path in  $\mathbf{S}_n$ . Hence proved. Note that in this proof though we have proved the equivalence of the tree based and co-occurrence based algorithms for the estimates of  $\omega(\mathcal{B})$  and  $\omega(\mathcal{B}_k)$ , the same proof can be used to prove the equivalence of the estimates of  $\eta(\mathcal{B})$  and  $\eta(\mathcal{B}_k)$ .  $\square$

**Lemma 1.** For any function  $f : \mathbb{S} \rightarrow [0.5, 1]$ ,  $\min_{\mathbf{s} \in \mathbb{S}} H_0(f(\mathbf{s})) \equiv H_0(\max_{\mathbf{s} \in \mathbb{S}} f(\mathbf{s}))$ . Similarly for any function  $g : \mathbb{S} \rightarrow [0, 0.5]$ ,  $\min_{\mathbf{s} \in \mathbb{S}} H_0(g(\mathbf{s})) \equiv H_0(\min_{\mathbf{s} \in \mathbb{S}} g(\mathbf{s}))$ .

*Proof.* First note that  $H_0(z)$  is a decreasing function for  $0.5 \leq z \leq 1$ . Hence  $H_0(z_1) \leq H_0(z_2)$  implies  $z_1 \geq z_2$ . Using this fact, we now prove this lemma by contradiction. Assume that  $\mathbf{s}_1 = \arg \min_{\mathbf{s} \in \mathbb{S}} H_0(f(\mathbf{s}))$  and  $\mathbf{s}_2 = \arg \max_{\mathbf{s} \in \mathbb{S}} f(\mathbf{s})$ . Further assume that  $f(\mathbf{s}_1) \neq f(\mathbf{s}_2)$ . Then clearly by definition  $f(\mathbf{s}_1) \leq f(\mathbf{s}_2)$  and  $H_0(f(\mathbf{s}_1)) \leq H_0(f(\mathbf{s}_2))$ . This is a contradiction since  $H_0(z)$  is a decreasing function. Thus  $f(\mathbf{s}_1) \equiv f(\mathbf{s}_2) \Rightarrow \min_{\mathbf{s} \in \mathbb{S}} H_0(f(\mathbf{s})) \equiv H_0(\max_{\mathbf{s} \in \mathbb{S}} f(\mathbf{s}))$ . Similarly we can prove the second part.  $\square$

**Lemma 2.** For any function  $h : \mathbb{S} \rightarrow [0, 1]$ ,  $\min_{\mathbf{s} \in \mathbb{S}} H_0(h(\mathbf{s})) \geq \min(H_0(\max_{\mathbf{s} \in \mathbb{S}} h(\mathbf{s})), H_0(\min_{\mathbf{s} \in \mathbb{S}} h(\mathbf{s})))$ .

*Proof.* This lemma directly follows from Lemma 1. If  $h(\mathbf{s}) \leq 0.5$ . then using lemma 1, we have:  $\min_{\mathbf{s} \in \mathbb{S}} H_0(h(\mathbf{s})) \equiv H_0(\min_{\mathbf{s} \in \mathbb{S}} h(\mathbf{s}))$ . Similarly if  $h(\mathbf{s}) \geq 0.5$ ,  $\min_{\mathbf{s} \in \mathbb{S}} H_0(h(\mathbf{s})) \equiv H_0(\max_{\mathbf{s} \in \mathbb{S}} h(\mathbf{s}))$ . Hence  $\min_{\mathbf{s} \in \mathbb{S}} H_0(h(\mathbf{s}))$  is definitely greater than the minimum of these two. Hence proved.  $\square$