Nested Object Watermarking

Claus Vielhauer[a,b], Maik Schott[a], Jana Dittmann[a]
[a] Dept. of Comp. Science, Univ. of Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany;
[b] Dept. of Informatics and Media, University of Applied Sciences in Brandenburg, Magdeburger Straße 50, 14770 Brandenburg an der Havel, Germany
Email: {claus.vielhauer|maik.schott|jana.dittmann}@iti.cs.uni-magdeburg.de, claus.vielhauer@fh-brandenburg.de

ABSTRACT

Annotation watermarking is an application of watermarking where information about a cover or a part thereof are embedded in the cover itself to link both directly together. In earlier work we introduced Nested Object Annotation Watermarking as a special case, where the semantic, shape and hierarchical relations between the depicted nested objects are embedded in the area of each object only. As these regions can be anywhere and may be composed of any shape there is very limited a-priori knowledge for synchronization, which results in a higher complexity and ultimately in a higher error-proneness. In general an exhaustive search strategy for proper block to reconstruct the shape suffers from the intrinsic combinatorial explosion of this process. Therefore in earlier work, at first we focused on rectangular embedding schemes with a block luminance algorithm, a steganographic WetPaperCode algorithm, a rectangular and finally a polygonal Dual-Domain DFT algorithm.

In this paper we review and compare these algorithms. For the latter one we also show the influence of several parameters, present our idea of a method to reduce the combinatorial explosion by collating the paths in the search tree, and show that our new synchronization approach surpasses our former rectangular method in terms of correct retrievals, despite its higher complexity.

Keywords: Watermarking, Image Processing, Protocol, Synchronization, Exhaustive Search

1. INTRODUCTION

Annotation watermarking – also called caption watermarking or illustration watermarking – specifies an application of watermarking for embedding additional informations about the cover – the non-watermarked original – or parts thereof directly into the cover. Although additional metadata may be embedded using metadata functionality of some file formats, they get easily removed by a file format conversion, e.g. the Image Description block of TIFF after a conversion to JPEG. Therefore in annotation watermarking the additional informations are intrinsically linked to the content they describe and should remain this link for non-malicious processing steps like such conversions and compression. Early annotation watermarking approaches for images include MediaBridge from DigiMarc [1], a robust method to withstand digital-to-analog and analog-to-digital conversion, like printing and xeroxing; or a non-robust method with interactive segmentation with hidden object-based annotations [2]. Illustration watermarking was introduced in [3] for computer generated images and in [4] for natural images to describe annotation watermarking, where the user interactively adds keywords for certain objects of the image. Annotation watermarking should be robust against cropping and only embed the metadata of objects they describe in the image area of these objects to overcome a "semantic gap". In common watermarking schemes the metadata is embedded in the entire picture. After cropping the embedded metadata can not be retrieved for the remaining objects although their metadata is only related to them not the entire image and therefore a "semantic gap" is created. Illustration watermarking was extended in [5] with Nested Object Annotation Watermarking (NOAWM), where the hierarchical semantic relationships between the annotated objects are embedded too.

Alongside with the arbitrary shape support introduced in [7], NOAWM allows the semantic annotation of objects in images and their relationships and further narrows the "semantic gap" by preserving almost their annotated shape. Such semantic annotations may be useful in the field of digital long-term preservation, which aims to preserve digital objects for at least 100 years and coming generations as these are considered to be part of our cultural heritage, particular considering many digital objects were "born digital". As over such a time span the semantics change it may be wise to embed them in the image, whereas their correct interpretation can be preserved in conjunction with the WordNet

ontology [10] we use retrieve the semantics of words and their relationships. In the EU FP7 SHAMAN project [12] for digital long-term preservation there is even an objective to find methods for the automatic segmentation and (semantic) annotation of image parts using machine learning, which can be combined with NOAWM.

Regarding the arbitrary shapes and comparing them to rectangular shapes where we could use our a-priori knowledge about the shape of the watermarks for proper synchronization, we have no such knowledge for arbitrary shapes, except the simple fact that they arbitrary. Therefore in synchronization we must additionally reconstruct the shape, which increases complexity of the retrieval and therefore the retrieval errors. In [7] our synchronization serially composed the shape by using a heuristic to find one next block, because a less error-prone exhaustive search approach seemed very resource consuming due to its intrinsic combinatorial explosion. In this paper we also describe an exhaustive search approach avoiding this combinatorial explosion by collating paths in the search tree.

For a better understanding and to summarize our previous work we describe the embedding and retrieval of 4 algorithms: Block-Luminance [4][5], WetPaperCode [9], rectangular Dual Domain DFT (DDD) [11] and polygonal DDD [7] in section 2. In section 3 we describe our experimental setup used for the evaluation in section 4, where we present our results of robustness against JPEG compression and cropping and compare them with our previous approaches. Section 5 concludes this paper with a summary and a discussion about future work.

2. EMBEDDING AND RETRIEVAL

This section describes the Hierarchical Graph Concept (HGC) in sub-section 2.1 and for all 4 algorithms their mode of operation w.r.t. embedding and retrieval, beginning with the Block-Luminance algorithm in sub-section 2.2, followed by the WetPaperCode algorithm in sub-section 2.3 and finally both DDD algorithms, more detailed, in sub-section 2.4.

2.1 Hierarchical Graph Concept

Each object is annotated by its semantic, shape and its relationship to other objects, i.e. if it is a meronym (subpart) of another object. The semantic can only be chosen by a user by entering a word expressing the desired semantic. One problem that arises is that one semantic may by expressed by different words, so called synonyms. Thus to avoid redundancy and reducing the input space synonyms should be identified and mapped to single semantic item. With this mapping another problem arises, that words with different semantics are written the same, so called homonyms. To avoid accidentally selected wrong semantics caused by selecting the wrong homonym a glossary is therefore needed describing each word and its semantic behind. These issues can be addressed by using a thesaurus, which links words by their semantic relationships and describes them. For this we use the WordNet ontology of the Cognitive Science Laboratory of the University of Princeton, which includes 155.000 English words. To embed the semantic of an object its ID in the WordNet ontology is used as message.

To embed the relationships which build a hierarchy each hierarchy node is relatively or absolutely numbered depending on the watermarking approach. For the Block-Luminance and the WetPaperCode algorithm the hierarchy nodes are relatively numbered for one hierarchy node o from the top hierarchy node down to the hierarchy node o, including each hierarchy node on the path, each number of hierarchy nodes of the path is concatenated and appended as part of the message.

For the DDD algorithm the relationship is embedded independently of the actual message for an intrinsic representation of the relationships. Each hierarchy node is absolutely numbered and assigned to a certain fixed frequency in the used subband. The relationship between a child hierarchy node and its parent hierarchy node is represented by setting the magnitude of the parent hierarchy node's frequency to the double (hs=2) of the highest frequency in the subband, and the magnitude for the child hierarchy node to the $\sqrt{2}$ of the former highest magnitude, as illustrated in Figure 1.

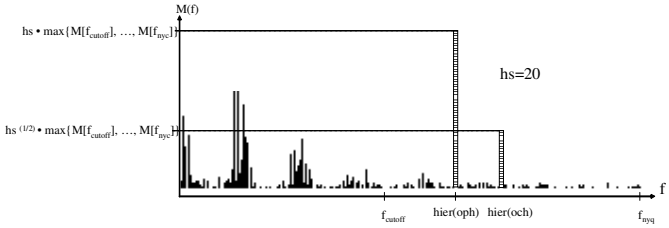

Figure 1: Embedding of a hierarchical relationship, exaggerated with a hierarchy embedding strength of *hs*=20

2.2 Block-Luminance

The Block-Luminance algorithm [4][5] is based on a method from Fridrich [6]. It is a block-based method, operating on the luminance values of the blocks. The main idea is for each block *B* where a bit should be embedded to add a pattern *P* to the luminance values of this block to embed a *bit*=1 or to subtract it from the luminance values to embed a *bit*=0. Both have the dimensions *blw*×*blh*=8×8. For each block a new pattern is chosen by initializing it with pseudo-randomly chosen binary values (either -1 or +1). For security purposes the used PRNG is initialized with a user-defined key. This pattern has a high frequency and therefore may be negatively affected by JPEG compression. Therefore first a cellular automaton is applied and each value is replaced by the values which most often occurs in its 8 neighbors or it is left as is if half of its neighbors are -1 and the other half +1 values. At next the pattern is smoothed by multiplying it with 3 and then changing each value to the integer mean of its 8 neighbors To allow the usage of an embedding strength each value is finally multiplied by an embedding strength *bls*.

For synchronization purposes a special fixed synchronization pattern is used, looking like a 2×2 chess board. The actual synchronization sequence consists of 15 such patterns with the synchronization bit string 4001_{16} as seen in Figure 2.

Figure 2: Appearance of the synchronization and the actual watermark patterns for a high embedding strength (*s*=10)

To retrieve what bit was embedded at a block at first the pattern for this block must be exactly calculated in the same manner as during the embedding. To detect if *bit* =1 was embedded, the sum of the luminances of the block values were a positive pattern values were added (L^+) should by higher than the luminance values were pattern values were subtracted (L^-), with L^+ and L^- defined as:

$$L^+ = \sum_{x=0}^{blw-1}\sum_{y=0}^{blh-1}\begin{cases} B_{xy} & \text{if } P_{xy} > 0 \\ 0 & \text{otherwise} \end{cases} \text{ and } L^- = \sum_{x=0}^{blw-1}\sum_{y=0}^{blh-1}\begin{cases} B_{xy} & \text{if } P_{xy} < 0 \\ 0 & \text{otherwise} \end{cases}$$

The opposite holds to detect if *bit*=0 was embedded, i.e. if $L^- > L^+$ then *bit*=0.

2.3 WetPaperCode

The WetPaperCode implementation is based on [9], where a detailed description can be found. WetPaperCodes are a steganographic approach where the message is embedded in the bits of the least significant bits of some pixels in the blue channel. In short, the code is computed from the solution of the linear equation system $H \cdot v = m - D \cdot b$ using the Gauss's algorithm in our case. Thereby *b* denotes a binary column vector, defining a set of indices $C \in \{0, 1, \ldots, n-1\}$, $|C| = k$ of the pixels that can be modified to embed a message, *m* is the message of *q* bits length to be embedded, and *v* is an unknown $k \times 1$ binary vector. Matrix *D* denotes a pseudo-random binary matrix of dimensions $q \times n$ generated by a shared secret key, whereas *H* is a binary $q \times k$ matrix consisting of those columns of *D* that correspond to the indices in set *C*. With the equivalence of $v = b' - b$, the embedding process generates the code by modifying each of the *C* positions in *b*, b_j, $j \in C$, so that the modified binary column vector *b*' satisfies $D \cdot b' = m$. Using the same shared matrix *D*, the decoder can

retrieve the message *m* in an analog manner. If a solution is not found for a given *D*, this matrix must be calculated again, based on another key *k' (k' ≠ k)* and a solution of the linear system of equations is computed again, based on the permutation of vector *v*.

2.4 DDD

The DDD algorithm [11][7] operates in the frequency domain, or more precisely, embeds the message in the phase domain and the hierarchy in the magnitude domain, thus *D*ual *D*omain *D*FT algorithm. The algorithm has two variants: the original rectangular and the polygonal. For the rectangular DDD algorithm, as well as the other two watermark algorithms, the actual embedding was performed into the area of the bounding box of the annotated shape. As the selection of blocks out of a rectangle is trivial it is not described in this paper. Rectangles represent only very coarsely the actual shape and thus are a "semantic gap". Originally, they were a compromise because the embedding of a polygonal shape described by the coordinates of its vertices would need a very high capacity and thus only usable for large objects. As the DDD algorithm supports the embedding of more than one bit per block, one bit is used as a so called "presence bit" describing that the block is part of a certain watermark. This can be used to allow arbitrary watermark shapes, not just polygonal ones. Another advantage of this approach for polygonal shaped watermarks is a better handling of overlapping watermarks. With the rectangular algorithms one watermark would always overwrite the mutual used blocks of another watermark. Thus children generally being smaller than their parents were always embedded last. With arbitrary shaped watermarks mutual used blocks can be removed from one of the overlapping watermarks thus punching holes. Because of this importance, block selection strategy is presented at first, than the embedding and retrieval processes and at least our new synchronization process using an exhaustive search but without the combinatorial explosion by collating paths.

Block selection
Let *po* be the polygonal shape of an object *o*. The whole image is partitioned into blocks. For each block the vertices are counted which are inside *po*. A block is included into the set of usable blocks for *o*, if the count exceeds at least a threshold $bet \in \{1,...,4\}$. The effect of different threshold values can be seen in Figure 3. *bet*=0 returns an outer block set and *bet*=4 an inner block set. We have arbitrary chosen the inner block set with a threshold *bet*=3.

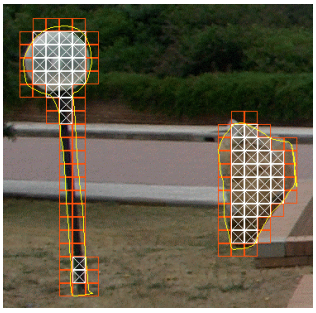

Figure 3: Objects with their polygonal shape and usable blocks for the 1 (w/o crosses) and the 3-edge criterion (with crosses)

The second step is the solution of conflicts because of mutual overlapping areas (MOA) of objects. Each block is classified, whereas all members of class are claimed by the same objects. For classes that represent more than 1 object and where all blocks are fully surrounded by other blocks that are to be used for watermarking, the member blocks are interleaved between the objects, e.g. for class $cmoa=\{b_1,...,b_m\}$ of *m* blocks from a MOA with *n* objects $Omoa=\{o_1, ..., o_n\}$ *b1* is assigned to o_1, b_2 to o_2, b_n to o_n, b_{n+1} to o_1 and so on. MOAs can be handled in two ways: the blocks in *cmoa* can be entirely assigned to exactly one object from *Omoa* or to several of them. If the blocks are not fully surrounded all blocks of this class are allotted to the smallest object. Due to the interleaving in the retrieval the shape would be reconstructed distorted as the shape has its more accurate representation with the initial sets than with the final sets where some blocks are assigned to other objects. This distortion can be countered by deinterleaving.

Embedding
The embedding is done for *bx*×*by* blocks in a subband of the spectrum defined by the Nyquist frequency $f_{nyq}=0.5(bx \cdot by)$ as the upper cut-off frequency and a lower cutoff frequency f_{cutoff}. Thus it operates in the high frequency part of the spectrum as high frequent noise is less perceptible.

The hierarchy is embedded in the magnitude part as described in section 2.1. The message is embedded in the phase part. The usable subband of frequencies between f_{cutoff} and f_{nyq} is further evenly divided into *bc* subbands, each holding one bit. The first subband holds the so called "presence bit" which is always set to 1 for synchronization purposes, whereas the other bands are used for the actual payload. The actual payload is encoded with a Reed Solomon ECC. For each watermark a PRNG is initialized with a user-defined key and the watermark index. The bit embedding for each frequency of one band is done as follows:

$$\Theta(f) = \begin{cases} -0.5\pi + rnd & \text{if } bit = 0 \\ +0.5\pi + rnd & \text{if } bit = 1 \end{cases}$$

whereby *rnd* is real-numbered output from the PRNG between [-π; +-π]. Therefore each phase is set to an pseudo-random value and the embedded bit can only be retrieved if a) each phase and therefore each block is read in the same order, which allows a detection if a wrong block was chosen during synchronization, b) if the PRNG is initialized with the same user-defined key, a simple security or authentication mechanism, and c) with the watermark index, which causes an unique PRNG sequence for each watermark and therefore allows a differentiation between them.

Retrieval

As the hierarchical relationship was embedded in every block in embedding process, in the retrieval process the magnitudes in the subband of each block of the watermark must read. For each block the read magnitudes are normalized to values between *0* and *1*. For each frequency *f* of the subband over all blocks the mean μ_f and standard deviation σ_f are calculated. If μ_f is not zero the real-numbered possible hierarchy level *posslvl* is the negative of the $\sqrt[2]{}$*hs-th* logarithm of μ_f and the nearest hierarchy level *nearlevel* is the rounded value of *posslvl*. Each *f* stands for a certain numbered hierarchy item. The distinction between a parent hierarchy item and the current child hierarchy item can be made by the hierarchy level *posslvl*, where a value of 1 denotes the child object and a hierarchy level of 2 the parent object. To detect corruptions a quality check is done by calculation a ratio $ratio_f$ between *posslvl* and *nearlevel* by subtracting both and scaling them back to linear scale. *nearlevel* denotes an actual level, if μ_f as well as σ_f are in the range is between $ratio_f - \sigma_f > 1 - \theta$ and $ratio_f + \sigma_f < 1 + \theta$. In the implementation θ is set to 0.25 to have a exclusion area as large as the range used for detection.

As the phases of the blocks are affected by noise and manipulations statistical means are necessary to detect if a 0 or 1 was embedded. For this over all phases of the subband the mean μ and standard deviation σ^2 are calculated using the magnitudes as weights, because the higher a magnitude the less it should be affected by noise. Assuming the noise satisfies a normal distribution with our calculated parameters $N(\mu, \sigma^2)$, we can calculate the probability P_0 that an embedded 0 equals the probability of a given random phase *X*, satisfying both distribution parameters, to fall in the range of dispersion at -0.5π with a radius of 0.5π, and P_1 analogously with replacing -0.5π *by* $+0.5\pi$. The probabilities of an embedded 0 or 1 are then determined in relation to each other to get the "soft bit" $sb = (-P_0 + P_1 + 1) / 2$. For the final "hard bit" *bit* a threshold value $sbt \geq 0.5$ is introduced as system parameter, controlling the width of the exclusion area between the areas for an embedded 0 and 1.

$$bit = \begin{cases} 0 & if\ sb < 1 - sbt \\ 1 & if\ sb > sbt \\ Error & otherwise \end{cases}$$

The value of *sbt* depends on the bit type. As the presence bit is not subject to the RS EEC we need a larger exclusion area to reduce false-positives and therefore set to *sbt=0.9*. If the bits are subject to the RS EEC we can set the exclusion area to the minimum value and let the RS EEC handle the errors.

Synchronization

The synchronization part consists of three steps: 1) all blocks' spectrums are read, b) the starting point of each watermark is searched, and 3) if a starting point was found all further blocks are exhaustively searched and the shape is reconstructed.

At embedding the PRNG is initialized for each watermark with a user-defined key and the watermark serial number to create a unique embedding sequence for each watermark. Therefore to find the starting point an exhaustive search must be used for each watermark separately, by initializing the PRNG with the user-defined key and trying each possible watermark serial number, retrieving the presence bit of each block and test if this bit is set. If none such block was found,

this may either mean that the user-defined key is wrong, a watermark with the used serial number was never embedded beforehand, or the first block got corrupted. In the first two cases the watermark is never be found, but unknown to the system, but in the latter case the watermark can be found by just search for the second block, and so on. For performance reasons not all serial numbers are tried, only *Δsync_wm* from the last found number on but at least at least *sync_wm_n*, and if the first block was not found only *sync_wmblocks_n* are searched.

If a starting block and therefore candidate for a watermark was found, the actual reconstruction of the watermark block sequence begins using our initially mentioned exhaustive search scheme with reduced combinatorial explosion.

Due to the exhaustive search there will be different paths, which one can follow. Each path is represented by a context comprised of the found watermarks blocks, its position in the image, PRNG state, count of successive dummy blocks and if the path reached a dead-end.

Each next block is searched on the current and next block row of the current block only. For each found block its position and the "soft bit" value of the presence bit is logged. If no next block was found, the direct right neighbor, or if the current block is at the end of the line, the direct neighbor below is treated as dummy block with a retrieved probability is set to 0.0. If the last block was found, no more neighbors can be found so that only dummy blocks will be generated from this point on. Therefore the state of a context is set to finished if the last block in the image was reached or the count of successive dummy blocks equals the count of blocks per row.

For each next block, a new path is assumed and therefore the current context will be copied and updated according to the respective next block. All contexts are searched who were already at the position of the current context, i.e. all contexts that have reached the same position and therefore will develop identically so that all but one can be discarded. To detect the context that should remain as the only one a performance criterion is used. A context has a higher performance if its cumulative "soft bits" are higher and there are fewer gaps in the found blocks. To detect such collating pathes/contexts as early as possible a width-first search is applied. The process ends when the last context gets invalidated or finished.

In the next section the experimental setup is presented.

3. EXPERIMENTAL SETUP

For our experiments we used the same Test Set B as introduced in [11] consisting of 15 images with at least 6 Mpixels size. For the cropping tests from these 15 images 52 cropped images were made. These images contain up to 120 annotations. During the creation it was taken care, that there are cropped images with missing parents, children and siblings to detect the performance of the hierarchy error detection and correction. Missing objects are either completely outside the cropped image or only partially as illustrated in Figure 4. In the last case they cannot be retrieved if the upper-left is outside as the embedding begins at this position. Only for cropping a second Test Set C is used with 30 images. From these cropped images were created each with 10 annotations in the mean.

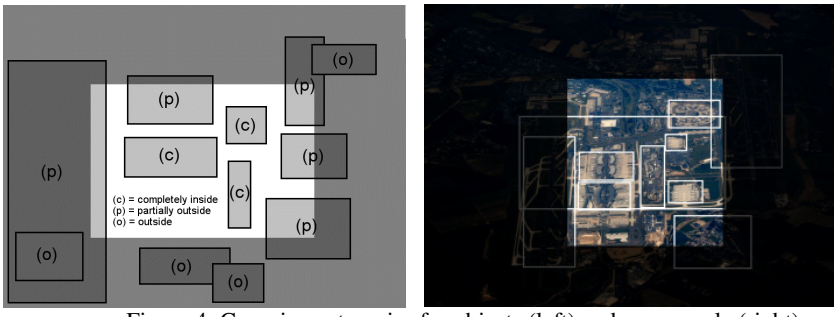

Figure 4: Cropping categories for objects (left) and an example (right)

For the robustness tests the images are compressed with JPEG at different quality levels. "None" denotes no compression at all and J<x> denotes a JPEG compression with a quality level of x. In the result tables the best results are highlighted with a light gray background.

Note that because of the block-based nature of the algorithm and its operation in frequency-domain, no robustness against geometric operations is expected. Therefore such evaluation was omitted. However our synchronization scheme

aims for robustness against JPEG compression cropping, provided that all watermarked blocks are preserved in the cropping, two operations commonly used on images.

At first in section 4.1 all for algorithms are evaluated in terms of transparency, capacity, capacity and shape infidelity. Shape infidelity was introduced in [8] and is defined as the ratio between the area of the annotated shape and the area of the retrieved shape. In the next section 4.2 the robustness of the polygonal DDD algorithm against cropping is evaluated and in the last section 4.3 the effects of the several internal parameters are measured.

4. EVALUATION RESULTS

4.1 Comparison between the watermark algorithms

At first the polygonal DDD algorithm is compared against the algorithms of the former project. Table 1 shows the results for transparency, measured with the Peak Signal-to-Noise Ratio (PSNR), the capacity and the shape infidelity of a polygonal shape.

		Block-Luminance	**WetPaperCode**	**Rectangular DDD**	**Polygonal DDD**
Average PSNR [dB]		49.55	90.37	47.89	47.22 (49.04)
Capacity [bits]		54,816.2	3,589,895.5	53,506.1	30,415.2
Shape infidelity		0.63	0.57	0.59	0.04
Robust-ness	**No JPEG compression**	99.33	98.59	98.59	98.59
	JPEG100	99.33	0.00	98.59	99.33
	JPEG90	99.33	0.00	20.42	37.33
	JPEG75	98.67	0.00	0.00	0.00

Table 1: Comparison of transparency, capacity, robustness and shape infidelity between the watermark algorithms

Although the polygonal DDD algorithm should have a higher transparency than the rectangular approach, because fewer blocks are used, this is actually not the case. The reason for this is that the polygonal embeds 2 bits more for each watermark due to its extended protocol. There may be much fewer presence bits set, because of the reduced block set in comparison to the rectangular approach, but the 2 additional data bits are embedded in the data sub-bands that – with the default settings – are located in lower frequencies than the presence bit's band and lower frequencies are known to be more perceptible than high frequencies. Therefore, additionally the average PSNR of the algorithm where the said two bits were not embedded as in [8] is additionally given in brackets. For this case we have the expected better transparency compared to the rectangular DDD. Of course the steganographic WetPaperCode algorithm has much higher transparency.

A drawback of the polygonal approach is the significant smaller shapes compared to bounding boxes, which leads to the lowest capacity.

The shape infidelity shows that the area of reconstructed shape of the annotations of polygonal DDD differs only 4% from the original annotated. The rectangular algorithms on the hand differ greatly with 57-63%.

The Block-Luminance algorithm has the highest robustness with only little degradation even for JPEG compression with at quality level 75. As expected the steganographic WetPaperCode algorithm has no robustness against JPEG compression even in the highest quality mode. Regarding the DDD algorithms, although non-rectangular annotations are much more complex, because of the non-existence of the a priori information about the shape of a watermark, the synchronization algorithm could be such way improved to find the same count of watermarks as the rectangular version. For JPEG quality level 90 this new synchronization algorithm also increases the robustness by 84% from 20.42% to 37.33%. For this JPEG compression in [7] we only gained a watermark detection rate of 2.07% for the polygonal DDD. The slightly higher percentage for the polygonal DDD with JPEG100 than without compression may just be a lucky occurrence where the JPEG compression subdued exactly the false block responsible for the one watermark not to be read correctly.

At next the cropping robustness for the polygonal DDD approach is evaluated.

4.2 Cropping

From the 120 annotations of the Test Set B only 118 were actually embedded, whereas the other 2 were too small. From these 118 annotations only 4 (3.3%) could not be found, but also 9 objects could be detected as missing although they

were not embedded. The last case can occur for missing parents as children always have links to their parents. For Test Set C all cropped images contained 156 embedded annotations in total – 4 could not be embedded in the regions of the cropped images, because they were too small. From these 5 (4.47%) could not be found and 39 could be detected as missing.

The cropping tests were not made with different embedding strengths, as the embedding strength has no effect on the robust against cropping as seen in [7]. Table 2 shows the results for cropping.

	Images	Total annotations	Missing annotations	
			Not detected	Detected
Test Set B	15	118	4 (3.3%)	9 (7.6%)
Test Set C	30	156	5 (4.47%)	39 (25%)
Total	45	274	9 (3.3%)	48 (17.5%)

Table 2: Cropping test results for both Test Set B and C

Therefore we can state that the algorithm is relatively robust against cropping with an error of 3.3%.

Next the several parameters of the polygonal DDD are evaluated.

4.3 Parameter evaluation

In the several tables showing the results, these are set in comparison to the default settings used up to now. These were intuitively chosen with block size = 16×16, embedding strength = 10, sync (presence bit) in last band, cut-off frequency percentage = 75%, sub-band count = 5, no cover pre-conditioning, Reed Solomon error correction percentage = 20%.

The first parameter, the *block size* (see Table 3), controls the size of the blocks used for embedding. As the block size directly influences the count of blocks, the higher block size the lower block count and the lower the capacity.

Parameter	Com-pression	Bit errors [%]		WM errors [%]	Hier. errors [%]	PSNR [dB]	Area mis-match [%]	Block errors [%]
		Raw	Cooked					
Block size = 8×8	None	0.00	0.00	-	-	40.95	1.31	-
	JPEG100	0.00	0.00	-	-	40.78	1.31	-
	JPEG90	0.26	0.00	-	-	38.06	1.31	-
	JPEG75	28.26	15.59	-	-	37.42	3.91	-
	JPEG50	64.80	21.32	-	-	37.45	21.32	-
Default Block size = 16×16	None	0.00	0.00	1.41	0.74	47.2	3.91	0.38
	JPEG100	0.00	0.00	0.67	0.74	46.7	3.95	0.37
	JPEG90	0.00	0.00	62.67	64.26	41.0	96.70	0.37
	JPEG75	10.30	10.38	100.00	100.00	39.7	-	-
	JPEG50	31.30	30.43	100.00	100.00	37.8	-	-
Block size= 32×32	None	0.00	0.00	0.00	2.50	50.3	7.33	1.28
	JPEG100	0.00	0.00	0.00	2.50	49.2	7.33	1.28
	JPEG90	0.04	0.00	100.00	100.00	41.7	-	-
	JPEG75	9.69	9.68	100.00	100.00	40.0	-	-
	JPEG50	25.38	25.78	100.00	100.00	37.8	-	-

Table 3: Evaluation of the impact of the block size

The cooked bit error is in some times higher than the raw bit error rate, because there are measured on a different data basis. For example using a RS(15, 9) code and assuming 15 raw bit errors in the 9 data symbols. As for this RS code 4 bits/symbol are used we have a raw bit error rate of 25%. After the error correction which cannot be successful we have these same 15 bit errors but only the 9 data symbols and thus a cooked bit error rate of 41.7%. An equal phenomenon is that although the cooked bit error rate is zero the watermark error rate is larger than zero. This happens because the search of the watermarks during the synchronization is only based on the correction detection of presence bit. For the default settings we have an almost ideal robustness up to JPEG100. Although the cooked bit error count is zero for JPEG90 68.21% of the watermarks were not found. For block size 8×8 some tests could not be performed, because the low block size leads to a to high count of blocks and such a large amount of data that an out of memory occurred. Also such a small block size in combination with the default cut-off frequency percentage of 75%, is more or less unusable as only 9 hierarchy items can be embedded, because only 9 frequencies are usable with this setting and each hierarchy items is assigned to a frequency. The transparency is also much less with a 5dB smaller PSNR. A block size of 32×32 is the only case where all embedded watermarks could be retrieved for no JPEG compression and a compress at the highest quality level. On the other hand for JPEG90 no watermarks could be found, whereas in the default mode 62.67% of the watermarks were not retrievable. The cause maybe that for 32×32 with its 4 times larger block than the default, 8.13 watermarks/image and 470.13 bits in total were embedded, but for the default settings 10.2 watermarks/image with 605.60 bits. Therefore it is possible that all the missing watermarks in the default mode were actually not embedded. The result for JPEG90 can be interpreted such that 32×32 is not robust against JPEG compression as a 32×32 consists of a set

of JPEG blocks and only one from these needs to be heavily altered by the JPEG compression to render the whole 32×32 block unreadable.

The *embedding strength* · 10 (see Table 4) is the minimum value a magnitude for a frequency used for embedding must have. Although the actual bits are embedded in the phase part, the higher the magnitude the more robust against changes the frequency (including its phase part) is. If a magnitude has a value below the embedding strength · 10 it is set to this value.

Parameter	Com-pression	Bit errors [%]		WM errors [%]	Hier. errors [%]	PSNR [dB]	Area mis-match [%]	Block errors [%]
		Raw	Cooked					
Strength= 5	None	0.00	0.00	1.41	0.74	49.1	3.86	0.43
	JPEG100	0.00	0.00	1.41	0.74	48.2	3.87	0.43
	JPEG90	1.82	0.32	98.42	97.44	41.4	-	-
	JPEG75	24.28	25.98	100.00	100.00	39.8	-	-
	JPEG50	37.17	36.53	100.00	100.00	37.8	-	-
Default Strength = 10	None	0.00	0.00	1.41	0.74	47.2	3.91	0.38
	JPEG100	0.00	0.00	0.67	0.74	46.7	3.95	0.37
	JPEG90	0.00	0.00	62.67	64.26	41.0	96.70	0.37
	JPEG75	10.30	10.38	100.00	100.00	39.7	-	-
	JPEG50	31.30	30.43	100.00	100.00	37.8	-	-
Strength= 50	None	0.00	0.00	1.04	0.74	35.0	3.94	0.37
	JPEG100	0.00	0.00	1.04	0.74	34.9	3.95	0.37
	JPEG90	0.00	0.00	2.07	2.15	33.9	4.06	0.37
	JPEG75	0.00	0.00	24.37	12.83	33.3	22.82	0.39
	JPEG50	0.03	0.00	79.19	93.47	34.5	-	-
Strength= 100	None	0.00	0.00	1.78	0.74	29.1	3.98	0.37
	JPEG100	0.00	0.00	1.78	0.74	29.9	3.92	0.38
	JPEG90	0.00	0.00	3.42	1.41	28.7	4.15	0.42
	JPEG75	0.00	0.00	6.16	2.81	28.3	4.45	0.40
	JPEG50	0.00	0.00	26.73	18.62	28.1	23.25	0.30

Table 4: Evaluation of the impact of the embedding strength

For an embedding strength = 5 we have almost no robustness against JPEG90, but the transparency is slightly higher (2dB). For an embedding strength = 50 we have a robustness against JPEG90 but at the high cost of a PSNR 12.2 dB smaller. For an embedding strength = 100 we have also a pretty good robustness against JPEG75 and a small against JPEG50, but with an additional PSNR decrease of 6.0dB, i.e. 18.2dB less than with the default embedding strength.

The *position of the synchronization band* (see Table 5) hosting the so-called "presence bit". The position of this presence bit's band can be either the first band in the spectrum – meaning nearer to the DC coefficient – or the last one – meaning nearer to the Nyquist frequency.

Parameter	Com-pression	Bit errors [%]		WM errors [%]	Hier. errors [%]	PSNR [dB]	Area mis-match [%]	Block errors [%]
		Raw	Cooked					
Default Sync in last band	None	0.00	0.00	1.41	0.74	47.2	3.91	0.38
	JPEG100	0.00	0.00	0.67	0.74	46.7	3.95	0.37
	JPEG90	0.00	0.00	62.67	64.26	41.0	96.70	0.37
	JPEG75	10.30	10.38	100.00	100.00	39.7	-	-
	JPEG50	31.30	30.43	100.00	100.00	37.8	-	-
Sync in first band	None	0.00	0.00	2.07	0.74	46.7	3.91	0.45
	JPEG100	0.00	0.00	2.81	0.74	46.2	3.93	0.39
	JPEG90	0.01	0.00	68.21	66.66	40.8	95.90	0.32
	JPEG75	11.53	13.18	100.00	100.00	39.6	-	-
	JPEG50	31.77	31.18	100.00	100.00	37.8	-	-

Table 5: Evaluation of the impact of the presence bit band position

Placing the presence bit in the lower frequencies instead of the default higher frequencies has only minimal effect on the robustness and transparency. In some way this is surprising, considering that the pattern from the presence bit is more low frequency and this was supposed be more visible.

The *cut-off frequency percentage* (see Table 6) defines the cut-off frequency in percent regarding to the Nyquist frequency and thus the total embedding band width. All frequency bands will be distributed with the same band width between the cut-off and the Nyquist frequency. The lower the cut-off frequency (percentage) the more frequencies can be used and changed, thus the higher the robustness but also the lower the transparency.

Parameter	Com-pression	Bit errors [%]		WM errors [%]	Hier. errors [%]	PSNR [dB]	Area mis-match [%]	Block errors [%]
		Raw	Cooked					
Cut-off frequency %= 25	None	0.00	0.00	0.67	0.74	39.5	3.99	0.34
	JPEG100	0.00	0.00	0.67	0.74	39.4	3.99	0.34
	JPEG90	0.00	0.00	36.79	33.61	37.0	38.07	2.80
	JPEG75	0.08	0.00	100.00	100.00	36.9	-	-
	JPEG50	3.27	0.73	100.00	100.00	36.4	-	-
Cut-off frequency %= 50	None	0.00	0.00	0.67	0.74	44.0	3.94	0.34
	JPEG100	0.00	0.00	1.06	0.74	43.7	4.01	0.34
	JPEG90	0.00	0.00	49.72	53.29	39.7	60.41	1.91
	JPEG75	1.14	0.00	100.00	100.00	39.0	-	-
	JPEG50	14.22	0.00	100.00	100.00	37.6	-	-
Default Cut-off frequency %=75	None	0.00	0.00	1.41	0.74	47.2	3.91	0.38
	JPEG100	0.00	0.00	0.67	0.74	46.7	3.95	0.37
	JPEG90	0.00	0.00	62.67	64.26	41.0	96.70	0.37
	JPEG75	10.30	10.38	100.00	100.00	39.7	-	-
	JPEG50	31.30	30.43	100.00	100.00	37.8	-	-

Table 6: Evaluation of the impact of the cut-off frequency (total embedding band width)

A cut-off frequency percentage of 25%, i.e. 75% of the spectrum is used for embedding, leads to a doubled robustness up to JPEG90 but lowers the PSNR by 7.7dB. A cut-off frequency percentage of 50% only lowers the PSNR by 3.2dB but instead of ⅓, ½ of the watermarks are correctly found for JPEG90. A drawback is that with these ½ correct watermarks the shape fidelity is only 60.41%, instead of 96.70% for the default cut-off frequency percentage and JPEG90.

The *sub-band count* (see Table 7) defines the count of frequency bands used between the cut-off frequency and the Nyquist frequency. One band is reserved for the presence bit, while the every other can contain one bit of actual payload. This means the higher the sub-band count the higher the capacity. Additionally, as one Reed Solomon symbol is embedded per block, sub-band count - 1 is also the size in bits of such a Reed Solomon symbol.

Parameter	Com-pression	Bit errors [%]		WM errors [%]	Hier. errors [%]	PSNR [dB]	Area mis-match [%]	Block errors [%]
		Raw	Cooked					
Sub-band count = 3	None	0.00	0.00	0.74	9.09	45.8	2.83	0.31
	JPEG100	0.00	0.00	0.74	9.09	45.3	2.83	0.31
	JPEG90	0.01	0.00	67.40	49.99	40.4	74.81	1.27
	JPEG75	6.36	2.91	100.00	100.0	39.5	-	-
	JPEG50	27.49	22.35	100.00	100.00	37.8	-	-
Default Sub-band count = 5	None	0.00	0.00	1.41	0.74	47.2	3.91	0.38
	JPEG100	0.00	0.00	0.67	0.74	46.7	3.95	0.37
	JPEG90	0.00	0.00	62.67	64.26	41.0	96.70	0.37
	JPEG75	10.30	10.38	100.00	100.00	39.7	-	-
	JPEG50	31.30	30.43	100.00	100.00	37.8	-	-

Table 7: Evaluation of the impact of the sub-band count

Setting the sub-band count to 3 should make the bits more robust as more frequencies are used. This value also sets the Reed Solomon symbol size to 2 and the Reed Solomon word length to 3. Surprisingly this has a negative effect on the hierarchy information with an error count of more than 10 times using the default value, but also a slight positive effect on the shape infidelity that is cut to half for no JPEG compression. The cause for this could not be found.

The *Reed Solomon error correction percentage* (see Table 8) defines how many erroneous symbols can be corrected at most. For one erroneous symbol to be corrected two parity symbols are needed. As a Reed Solomon has a constant length, more parity symbols means less data symbols. Therefore, the higher this percentage, the higher the robustness, but the lower the capacity.

Parameter	Com-pression	Bit errors [%]		WM errors [%]	Hier. errors [%]	PSNR [dB]	Area mis-match [%]	Block errors [%]
		Raw	Cooked					
Default RS error correction = 20%	None	0.00	0.00	1.41	0.74	47.2	3.91	0.38
	JPEG100	0.00	0.00	0.67	0.74	46.7	3.95	0.37
	JPEG90	0.00	0.00	62.67	64.26	41.0	96.70	0.37
	JPEG75	10.30	10.38	100.00	100.00	39.7	-	-
	JPEG50	31.30	30.43	100.00	100.00	37.8	-	-
RS error correction = 50%	None	0.00	0.00	0.95	10.32	46.7	2.47	0.27
	JPEG100	0.00	0.00	0.95	10.32	46.1	2.49	0.27
	JPEG90	0.03	0.00	74.37	75.05	40.8	-	0.28
	JPEG75	11.01	3.75	100.00	100.00	39.6	-	-
	JPEG50	31.11	33.16	100.00	100.00	37.8	-	-

Table 8: Evaluation of the impact of the Reed Solomon error correction symbol count

Setting the Reed Solomon parity symbols to maximum, i.e. only 1 symbol is left for the data all other are used as parity. But surprisingly the hierarchy error count also increases. A possible cause is that with only 1 symbol left for the actual message, a new Reed Symbol word is needed for every data symbol. This higher usage of blocks can also be seen in the lower PSNR. This capacity requirement cannot be met by all watermarks. Actually only 7 watermarks were embedded in average per image (in total 105), compared to 10.2 watermarks/image (in total 153). As there are 33% watermarks less, it may be possible that probably the watermarks with a robust embedded hierarchy had.

Finally, it can be seen that the embedding strength and the cut-off frequency percentage have the largest positive influence on the robustness, but also a negative influence on transparency. If we set a PSNR of 40 dB for the uncompressed marked images as a lower threshold, and we want to distribute the robustness increase by using both parameters, we can use an embedding strength = 20 and a cut-off frequency = 50% (see Table 9).

Parameter	Com-pression	Bit errors [%]		WM errors [%]	Hier. errors [%]	PSNR [dB]	Area mismatch [%]	Block errors [%]
		Raw	Cooked					
Default	None	0.00	0.00	1.41	0.74	47.2	3.91	0.38
	JPEG100	0.00	0.00	0.67	0.74	46.7	3.95	0.37
	JPEG90	0.00	0.00	62.67	64.26	41.0	96.70	0.37
	JPEG75	10.30	10.38	100.00	100.00	39.7	-	-
	JPEG50	31.30	30.43	100.00	100.00	37.8	-	-
Strength = 20, Cut-off frequency %= 50	None	0.00	0.00	1.04	0.74	40.2	4.01	0.34
	JPEG100	0.00	0.00	1.04	0.74	40.0	4.01	0.34
	JPEG90	0.00	0.00	26.22	18.93	37.6	28.51	0.27
	JPEG75	0.00	0.00	96.65	98.00	37.6	-	-
	JPEG50	1.30	0.00	100.00	100.00	37.3	-	-

Table 9: Evaluation of the impact of combining the best performers embedding strength and cut-off frequency

Using this combination we can increase our robustness against JPEG90 in terms of correct found watermarks by 2.3 times and terms of a correct found hierarchy by 3.3 times.

5. CONCLUSIONS AND FUTURE WORK

We have shortly described the general embedding and retrieval processes of four different watermarking algorithms: the spatial domain, block-based Block-Luminance algorithm, the steganographic LSB-based WetPaperCode algorithm, as well as the frequency based DDD algorithm in its original rectangular and current polygonal variant. All these were compared against each other, w.r.t. to capacity, transpareny, robustness and shape infidelity. It was shown that the most robust algorithm was the Block-Luminance algorithm and the most transparent algorithm as well as the one with highest capacity is the WetPaperCode, which also has no robustness against JPEG compression. The polygonal DDD algorithm preserves the original shape with only 4% error, whereas the rectangular approaches had at least 57% error. The robustness of the polygonal DDD algorithm is on par with its former rectangular variant, although much less a priori information is known. This can be attributed to the exhaustive search synchronization process with the novel search tree paths collation which reduced the combinatorial explosion and thus made its use practically feasible. Another part of this

paper was the evaluation of the different internal parameters of the polygonal DDD algorithm to get estimates for the impact of each parameter, whose values were initially only chosen by intuition. It was shown that the embedding strength and the cut-off frequency (total embedding band width) have the greatest impact on the robustness. Whereas the position of the synchronization band and the count of maximum correctable errors for the Reed Solomon ECC have only very little impact. As the count of correctable errors has only this little impact, this may be an indication that the Reed Solomon ECC has reached its limit. Therefore future work may be the use of either a more powerful ECC like Turbo Codes or combining the Reed Solomon ECC with a Viterbi decoder. Both being options that operate very near to the Shannon capacity. Another future work could be the research of a invertible variant of the polygonal DDD algorithm.

ACKNOWLEDGEMENTS

This work has been sponsored by the Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant number FA8655-07-1-3013. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

The work in this paper has been supported in part by the European Commission through the FP7 ICT Programme under Contract FP7-ICT-216736 SHAMAN. The information in this document is provided as is, and no guarantee or warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

REFERENCES

[1] B. Perry, B. MacIntosh, D. Cushman: “Digimarc MediaBridge: the birth of a consumer product from concept to commercial application”, Proc. SPIE Vol. 4675, p. 118-123, Security and Watermarking of Multimedia Contents IV, 2002

[2] Y. Rytsar, S. Voloshynovskiy, F. Ehrler and Thierry Pun: “Interactive segmentation with hidden object based annotations: towards smart media”, Proceedings of SPIE Electronic Imaging, Storage and Retrieval Methods and Applications for Multimedia, San Jose, USA, 2004

[3] H. Sonnet, T. Isenberg, J. Dittmann, T. Strothotte: “Illustration Watermarks for Vector Graphics”, Proc. of IEEE Pacific Conference on Computer Graphics and Applications, pp. 73-82, 2003

[4] T. Vogel and J. Dittmann: “Illustration Watermarking: An Object Based Approach for Digital Images”, Proceedings of SPIE 2005, Jan. 20 - 27, San Jose, California, USA, 2005

[5] C. Vielhauer and M. Schott: “Image Annotation Watermarking: Nested Object Embedding using Hypergraph Model”, Proceeding of the 8th ACM Workshop on Multimedia and Security, pp. 182-189, Geneva, Switzerland, 2006

[6] J. Fridrich: “Methods for data hiding”, Center for Intelligent Systems & Department of Systems Science and Industrial Engineering, SUNY Binghamton, 1997

[7] C. Vielhauer, M. Schott: “Nested Object Watermarking: From the Rectangular Constraint to Polygonal and Private Annotations”, Proceedings of the ACM Multimedia and Security Workshop 2007, pp. 187–193, Dallas, Texas, USA

[8] C. Vielhauer, M. Schott, C. Krätzer, J. Dittmann: “Nested Object Watermarking: Transparency and Capacity Evaluation”,: Proceedings of SPIE, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X, January 26th–31st, 2008, San Jose, CA, USA

[9] J. Fridrich, M. Goljan, P. Lisonek, and D. Soukal, Writing on Wet Paper, IEEE Trans. on Sig. Proc., Special Issue on Media Security, Eds. T. Kalker and P. Moulin, vol. 53, pp. 3923-3935, 2005

[10] C. Fellbaum: “WordNet: An Electronic Lexical Database (Language, Speech and Communication)”, MIT Press, ISBN 0-262-06197-X, 1998

[11] C. Vielhauer, J. Dittmann: “Nested Object Watermarking: Comparison of Block-Luminance and Blue Channel LSB Wet Paper Code Image Watermarking”, Proceedings of SPIE Electronic Imaging - Security, Steganography and Water-marking of Multimedia Contents IX, Vol. 6505, pp. 65050L, 2007

[12] SHAMAN website, http://www.shaman-ip.eu