Fast and robust fragile watermarking enabling real-time self-recovery for UAS

Laurens Le Jeune[0000−0003−0744−4897], Anna Hristoskova[0000−0002−5012−7400], and Farhad Aghili[0000−0002−1830−6003]

Sirris, Brussels, Belgium
firstname.lastname@sirris.be
https://www.sirris.be

Abstract. Unmanned Aircraft Systems (UASs) are increasingly being integrated into the life-cycle management of critical infrastructure, including tasks such as inspection, maintenance, safety, and surveillance. However, legal concerns such as visual privacy must be addressed resulting in the need to explore anonymization techniques. Specifically, in cases where law enforcement requires access to the original data, reversible anonymization necessitates robust, complex and secure solutions, ensuring that only authorized individuals with proper credentials can recover the data.

In this paper, we introduce two novel fragile watermarking approaches with self-recovery, *HiLoSpatial* and *HiResSpatial*, that are tailored towards low-latency execution on UAS platforms, and that enable reconstruction from downstream anonymization steps. Our experiments show that *HiLoSpatial* and *HiResSpatial* match and in most cases surpass related work with regards to security, image integrity, tamper detection and recovery, while significantly outpacing the state of the art with respect to latency. These results highlight the suitability of our approaches for real-time deployment in UASs.

Keywords: Fragile Watermarking · Secure Shuffling · Self-recovery · Visual Privacy.

1 Introduction

UASs see adoption across many domains, as they expedite tasks through automation or execute risky work to improve safety. They are increasingly being incorporated in the life-cycle management of critical infrastructure [11], e.g. damage [15] and safety inspection [24], surveillance tasks [10], and other applications [25]. However, when employed for surveillance or monitoring of critical infrastructure, privacy concerns may arise. Personal data pertaining to visual privacy, such as people or license plates, need to be anonymized in accordance with regulations such as the General Data Protection Regulation (GDPR) [9].

Anynomization methods could take the form of various transformations, such as blurring or morphing [4], or using Machine Learning (ML) techniques [34].

Ideally, such anonymizations should be executed on-device to limit the potential exposure of personal data. However, due to the constrained nature of resources such as compute on UASs the anonymization process should be lightweight to limit any additional overhead. While effective in protecting privacy, these methods introduce challenges when access to original, unaltered data is required in specific situations. For example, in the case of critical infrastructure, the need may arise to reverse the anonymization of a specific footage in view of identifying perpetrators or suspicious vehicles by law enforcement.

1.1 Motivation

The main challenge of designing a privacy-preserving solution for UAS surveillance is being able to anonymize visual data while still retaining the ability to recover the original image details when necessary. Therefore, in this paper, we explore a fragile watermarking technique as a potential solution to recognize anonymization and provide recovery capabilities. Fragile watermarking enables embedding of authentication information within an image prior to the anonymization process. This watermark can later help identify tampering and recover the original image content. However, for such a solution to be feasible for UAS applications (or any resource constrained video-based applications), it must meet several vital requirements. First, the watermark embedding process must demonstrate high throughput, ensuring it is efficient enough to support real-time operations on resource-constrained UAS hardware. Additionally, the watermark should not considerably compromise on the image integrity or interfere with the anonymization process. Moreover, ensuring robust security is essential, as it requires the embedded watermark to be resilient against unauthorized access, thereby preventing leakage of sensitive information or unauthorized alterations.

1.2 Contributions

In this paper, we proposes two novel fragile watermarking solution tailored for on-device UAS execution supporting the following contributions:

- Real-time processing: Maintains a high throughput when embedding, significantly surpassing the state of the art and allowing real-time embedding.
- Robust integrity verification and recovery: Matches and even surpasses related work on fragile watermarking, when it comes to maintaining and recovering the watermark integrity.
- Enhanced security measures: Protects against attacks by applying thorough randomization for both recovery value localization as well as authentication.

Sect. 2 describes the current state-of-the-art solutions that approach the security, throughput and robustness requirements. Our proposed approach is further outlined in Sect. 3, including a detailed discussion of the executed experiments in Sect. 4. After discussing the results in Sect. 5, we conclude the paper in Sect. 6

2 Background

2.1 Reversible visual privacy

Existing techniques to facilitate reversible visual privacy, where sensitive data can be de-identified (or anonymized) and afterwards restored constitute an active research area. One intuitive approach is to use ML for this task. Employing an encoder-decoder structure divides the task in two components, where the encoder does anonymization and the decoder uncovers the original content [18]. Generative Adversarial Networks (GANs) employ a generator that generates alternative representations, and a discriminator that has to learn to distinguish between real and anonymized images [16, 33]. The security of such techniques can be further augmented with encryption [32]. Another approach employs watermarking alongside other cryptographic techniques [31]. However, this requires landmarks or coordinates which indicate the sensitive parts in an image.

Commonly thus, reversible visual privacy techniques either need some detection mechanism to identify the Region of Interest (ROI), or have the detection mechanism built-in when using ML. As such, to facilitate reversibility, computationally expensive techniques are required. In our work, we do not rely on any ROI, but instead watermark the entire image. If the downstream anonymization engine then anonymizes parts of the image, our techniques allow for reconstruction, even if the anonymization technique itself is not reversible. The watermark supports localizing anonymized areas after which recovery information stored in untampered areas enables recovery.

2.2 Fragile watermarking with self-recovery

Watermarking comprises various techniques that can be used to protect digital media. Typically, they belong to one of three categories: robust, semi-fragile or fragile watermarking [14]. Robust watermarking encompasses techniques that embed a watermark inside a cover image that is resilient against filtering, compression or attacks, and are usually used for copyright protection. Fragile watermarks instead embed watermarks that are altered when the cover image is altered. This way, tampering can be detected, and the tampered areas can potentially be restored. Finally, semi-fragile watermarks are robust with regards to benign transformations, but tend to deform when under attack [23].

In this paper, we consider fragile watermarking with self-recovery, meaning that the watermark embeds a representation of the cover image to allow for reconstruction after tampering.

Tampering may represent any malicious change compromising the integrity of the image. For fragile watermarking, this translates to a number of attacks that replace pixel values with different values. Common examples include copy-paste [27, 28, 21, 13, 22, 29] or writing text on top of the image [13, 22, 29]. Most commonly however the cropping attack is considered [27, 8, 28, 21, 12, 13, 22, 29], in which part of an image is removed and replaced by some set of values. This is the tampering we will consider in this paper.

2.3 Related work

In this section we review related work, with a special focus on throughput. In [20], Raj *et al.* investigated a number of watermarking schemes with self-recovery on the same hardware, providing an excellent starting point. Their results consider the impact of watermarking with regards to image integrity (in Peak signal-to-noise ratio (PSNR), as explained in Sect. 3.5) and embedding time, with all experiments executed on the same hardware. Their experiments suggest that Rajput *et al.*'s work [21] significantly outpaces other work with regards to embedding speed. They reach their speed by keeping the embedding process straightforward, i.e., they store four lower-resolution copies of the original image in the four Least Significant Bits (LSBs) of the original image. They then compare these images to detect tampering, and use the median values for recovery. Other fragile watermarking approaches considered by Raj *et al.* [20] all feature embedding processes that run for multiple seconds up to over a minute [2, 27, 3, 19, 28], and as such are too slow to be considered in our investigation.

Further literature review reveals that several authors also consider the embedding time of their algorithm. Tab. 1 presents some of these implementations, to facilitate comparison. Note that these methods are all benchmarked on different platforms, which limits the validity of throughput comparisons based on individually reported results alone.

Bravo-Solorio *et al.* [8] introduce a watermarking mechanism based on tornado codes. After a block-based tamper detection approach based on the hash values of stored reconstruction bits, they use an iterative process to recover tampered areas. Their boast a 100% recovery for tampering of up to 25% of the image, with performance steeply declining afterwards.

Gul *et al.* [12] propose a block-based approach where average values of 4×4 blocks are stored in predetermined partner blocks, and where the recovery values are hashed to facilitate tamper detection. While their approach seems to maintain image integrity, it is very slow.

Hussan *et al.* [13] suggest a similar block-based averaging that uses a chaos mapping to randomize the location of blocks. While they boast good image integrity performance in both embedding and recovery, their latency is also slow.

Sisaudia *et al.* [29] divide the input image in four sub images. They then use 4×4 block averages for recovery and Local Binary Patterns (LBPs) for authentication. They focus on the real-time component of the extraction and recovery however, not the embedding, which is still relatively slow. They also note that they can only resolve tampering for up to 50% of the image, after which their performance quickly deteriorates.

Renklier *et al.* [22] provide another block-based approach that shuffles a portion of the image based on a sudoku-inspired heuristic. They use 5×5 block averages for recovery and hash-based authentication values. Their timing and integrity results seem to be in a similar vein as most other work in Tab. 1. In this Table, the PSNR is after watermarking, not tampering or recovery. Time is given for embedding (T_e), detection (T_d) and recovery (T_r). BPP represents the number of watermark bits that are stored at every pixel. All results are for

512×512 sized images, and the average for many images whenever applicable. Rows in bold have been selected for our experiments.

Table 1. Fragile watermarking approaches in the literature that consider embedding speed. Ranges in square brackets report T_d and T_r combined.

Ref.	Date	PSNR (dB)	BPP	T_e (s)	T_d (s)	T_r (s)	CPU (Intel)
[21]	**2020**	**29.82**	**4**	**2.40**	**2.9**	**0.55**	**i7 8GB**
[8]	2018	37.9	3	3.2	-	27.8	i5 2.67Ghz 4GB
[12]	2020	44.41	2	179.37	[180-198]		i5–7500 3.4GHz 4GB
[13]	2022	39.22	2	16.66	-	-	N3350 1.1GHz 4GB
[29]	**2024**	**44.75**	**2**	**3.16**	**0.55**		**i7 6GB**
[22]	2024	44.22	2	3.57	[4.45-5.07]		i7-10500 2.8GHz 16GB

Besides throughput, we are also interested in maintaining the image integrity as much as possible. Shebab *et al.* [26] use the singular values of a Singular Value Decomposition (SVD) to authenticate blocks, and uses average values to provide reconstructive capabilities. Molina-Garcia *et al.* [17] transform their image to the YCrCb color space and apply Floyd-Steinberg dithering on the luminance component in an effort to downsize the reconstruction image. For authentication, they use Exclusive OR (XOR) computations to hash blocks of the original image. Both methods boast excellent performance in the literature.

From this related work, we include [21, 29] for our experiments, as these seem the most likely candidates for high-speed solutions. Additionally, we consider [26, 17] as a reference for high-quality fragile watermarking. Finally, we also include the work of Bouarroudj *et al.* [6, 5, 7], as their code is publicly available, even though they do not report their speed. They propose a Discrete Wavelet Transform (DWT) approach where they use DWT to both extract recovery bits and do tamper detection, and they boast excellent integrity results.

2.4 Open issues

When considering related work for fragile watermarking, several open issues (i.e., dealing with full pixels to zero tampering, secure shuffling, and latency) need to be resolved to allow for real-time embedded watermarking.

Full pixel zero tampering A core issue occurring in multiple papers [26, 17, 29] is storing authentication bits at the location they authenticate. Consider image $I(x, y)$ with dimensions $m \times n \in \mathbb{N}$ and x and y as coordinates. Function $a(I, x, y)$ generates an authentication digest of I at location (x, y), and $T(I, x, y)$ represents some tampering operation on I at location (x, y). Any time $a(I, x, y) = T(I, x, y)$, the authentication will not detect the tampering. A simple example of this is the special case that a location is zeroed out completely.

$$a(I, x, y) = \frac{\sum_{i=0}^{1} \sum_{j=0}^{1} I(x + i, y + j)}{4} \tag{1}$$

Let a be some averaging operation as demonstrated in Eq. 1. In this case, if an attack would set all pixels to zero, this authentication method would not detect any tampering, as the average of four zero-pixels is also zero.

In this work, we try to avoid this issue by using an authentication function that is independent of a location's value.

Secure shuffling Another issue is that while many papers propose some shuffling approach, those approaches are often not suitable for security applications. For instance, [26, 6] use Arnold's Cat map [1], and [12] uses the logistic map, both of which are chaotic mapping techniques. While these methods may provide adequate shuffling to ensure recovery values are not all stored at their source location, their security is not guaranteed: Arnold's cat is periodic and will return its input value after a number of iterations, while the logistic map has too small of a key space [36]. Other approaches [17, 29] forgo a chaotic approach and simply use fixed patterns, which allow for targeted attacks that tamper with everything except for the authentication bits.

We propose randomization for both our shuffling as well as our authentication to ensure that no locations or values are predictable.

Latency From the related work, it is clear that latency is not a parameter of considerable concern for most fragile watermarking approaches. Only a portion of authors reports latency, and an even smaller portion also explicitly mentions latency as a design constraint [8, 29]. Tab. 1 shows that even low-latency schemes feature multiple-second durations for relatively small 512×512 images, with only two sub-second latencies for tamper recovery. This may be sufficient for single-image use cases, but not for processing larger images in bulk.

In our work, we maintain image integrity and security constraints while reporting millisecond-scale embedding times for comparable images.

3 Proposed fragile watermarking approach

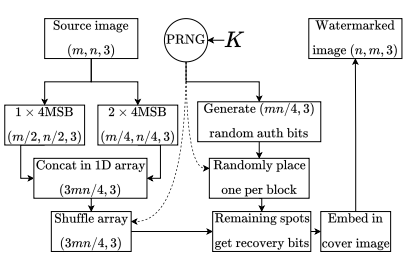
3.1 Scenario

We consider images captures by UAS that to be watermarked and anonymized while still on the UAS to avoid information being leaked downstream. Afterwards, with the appropriate key, hidden information can be recovered.

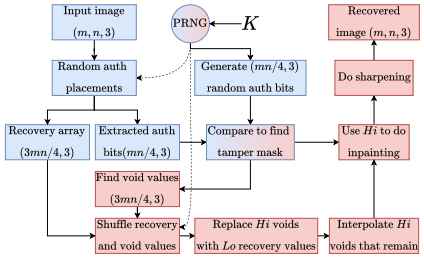
For this scenario, we propose two solutions in this paper, *HiLoSpatial* and *HiResSpatial*, to embed a fragile watermark as well as recovery information. *HiLoSpatial* embeds one 50% size and two 25% size recovery images inside the original image, while *HiResSpatial* embeds one grayscale full-size recovery image. Both methods employ a spatial embedding approach and expect RGB-images, but *HiLoSpatial* also works for grayscale images. Both are designed with low-latency constraints in mind, avoiding complex computations to limit overhead.

3.2 HiLoSpatial

For the *HiLoSpatial* watermarking in Fig. 1a, we consider an $n \times m \times 3$ RGB image. From this image, we extract two downsized versions from reconstruction: *Hi* has higher resolution, with a $\frac{n}{2} \times \frac{m}{2} \times 3$ shape, while *Lo* has a lower resolution, with a shape of $\frac{n}{4} \times \frac{m}{4} \times 3$. We use the *Hi* version once and the *Lo* version twice, and we isolate the 4 Most Significant Bits (MSBs) from these three images. These will form the recovery bits for image reconstruction after tampering.



(a) Watermark embedding process for the *HiLoSpatial* approach.



(b) *HiLoSpatial* tamper detection and recovery flow.

Fig. 1. *HiLoSpatial* embedding, detection and recovery flows. Blue elements represent tamper detection, red elements represent tamper recovery. Dashed lines mean that the PRNG is used to shuffle, not to generate data.

By storing multiple, lower-resolution copies of the cover image, we reduce the number of pixels needing interpolation during the recovery process. After extracting $1 \times Hi$ and $2 \times Lo$, each 4-bit value is first split in two 2-bit values. Their results are reshaped to 1D vectors for each channel, concatenated, and then shuffled using a PRNG. This PRNG allows for predictable shuffling if we use a known key K as seed, but should seem random when the seed is unknown. By shuffling, we ensure that recovery values are not all stored at their origin, as otherwise tampering an area would also corrupt the area's recovery values.
For authentication, we generate $(mn/4) \times 3$ authentication values between 0 and 3, which are all different 2-bit values. To construct the watermark, we divide each channel of the target shape $(m \times n)$ into 2×2 blocks. For example for a channel of a 4×4 image (Fig. 2a), we store one authentication value $a_i, i \in [0, mn/4]$ and three recovery values $r_j, j \in [0, 3mn/4]$ in each 2×2 block. The location of a_i is randomly and independently determined for each block. After assigning all a_i, the r_i values are taken from the shuffled array and stored in the empty locations of the watermark. Finally, the resulting 2-bit watermark, which has the same shape as the cover image, is used to replace the 2 LSBs of the cover.
The *HiLoSpatial* tamper detection and recovery flow is given in Fig. 1b. Using the PRNG with K as seed, the placements of the a_i values, as well as their

expected values, are retrieved. The retrieved values can then be compared to the expected values to detect tampering. However, as explained in Sect. 3.4, this comparison will not necessarily detect all tampering. As such, in order to completely capture tamper areas, we use Algorithm 1 to mark blocks 8-adjacent to tampered blocks as also being tampered. In essence, this algorithm grows all tamper instances with one unit, practically meaning that we mark a 2-pixel band around any detect tampering as tampered.

Algorithm 1 Grow algorithm to improve tamper detection coverage.

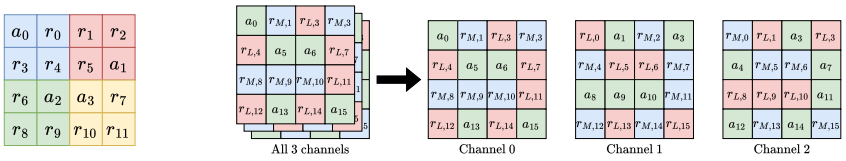
Require: tm with x_{size}, y_{size} ▷ 2D array tamper mask with x_{size}, y_{size} shape.
  $m \leftarrow tm$
  for $x = 0, \ldots, x_{size} - 1$ **do**
    for $y = 0, \ldots, y_{size} - 1$ **do**
      if $m[x, y] = 1$ **then**
        $x_{min} \leftarrow max(0, x - 1)$
        $x_{max} \leftarrow min(x_{size}, x + 1)$
        $y_{min} \leftarrow max(0, y - 1)$
        $y_{max} \leftarrow min(y_{size}, y + 1)$
        $m[x_{min} : x_{max}, y_{min} : y_{max}] \leftarrow 1$
      end if
    end for
  end for
  $tm \leftarrow m$

The resulting tamper mask indicates which 2×2 blocks have been tampered with. In the case of tampering, the appropriate enclosed recovery values must be used to reconstruct the original image. For this, we first determine *void* values, i.e., the recovery values that are no longer valid due to their 2×2 storage block being tampered with. We do this by first marking all tampered pixels in a new array. Then, after undoing the shuffling on the watermark, the array of recovery values can be extracted and reshaped into the *Hi* and *Lo* recovery images. Additionally, performing the same shuffling reversal on the marked tampered pixel array reveals which recovery values have become *void*. We then construct *void* masks to represent this information. Next, valid *Lo* values are used to fill in the *void Hi* values whenever possible, Afterwards, any remaining *voids* in *Hi* are filled by taking the average value of their neighbors. Once no more *voids* are present in *Hi*, it is used to fill in the tampered portions of the input image. Optionally, we also include a sharpening step in which we add the Laplacian of the reconstruction area to that area, as using the lower-resolution *Hi* and *Lo* recovery images can result in a blurry recovery area.

3.3 HiResSpatial

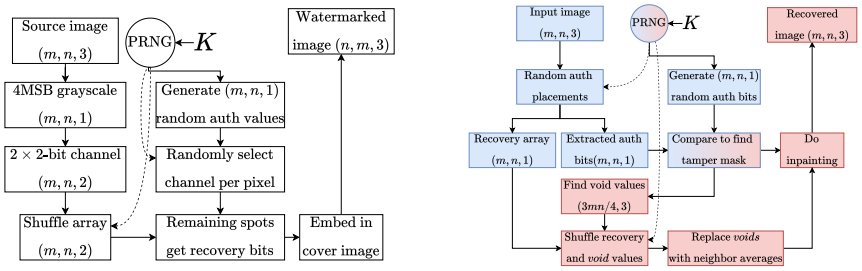
The *HiResSpatial* approach aims to facilitate high-resolution recovery of features in an image, as for example small texts may become illegible after recovery in

HiLoSpatial and identified related work. For example, if a small license plate is anonymized, it is important that all of its characters are legible after recovery. As such, we store the 4 MSBs of a full-resolution grayscale representation of the cover image. The embedding process as shown in Fig. 3a is similar to *HiLoSpatial*, but the approach to distribute authentication values significantly differs. We now generate and store authentication values $a_i, i \in [0, mn[$ for every RGB pixel, and store them in a randomly selected channel. The 4-bit grayscale recovery values are split in their LSB and MSB counterparts, $r_{L,j}$ and $r_{M,j}$ for $j \in [0, mn[$, respectively. The remaining spots are filled channel-wise in the following order: a_i, then $r_{L,j}$, then $r_{M,j}$ as demonstrated in Fig. 2b. The resulting construction is embedded in the cover image.



(a) (b)

Fig. 2. Approach to storing authentication and recovery values in *HiLoSpatial* (a) and *HiResSpatial* (b). While for *HiResSpatial* the per-pixel distribution is randomized, for subsequent channels, the order always amounts to $a_i \to r_{L,i} \to r_{M,i}$.



(a) Watermark embedding process for the *HiResSpatial* approach.

(b) *HiResSpatial* tamper detection and recovery flow.

Fig. 3. *HiResSpatial* embedding, detection and recovery flows. Blue elements represent tamper detection, red elements represent tamper recovery. Dashed lines mean that the PRNG is used to shuffle, not to generate data.

The recovery approach is nearly identical to *HiLoSpatial*. After undoing the data distribution and retrieving the stored information, the authentication values

can be compared to construct a tamper mask. This mask is more fine-grained however, as each pixel has its own authentication value. This means that we can detect tampering on a pixel-level as opposed to a 2×2 block-level, which was the case for *HiLoSpatial*. If tampering is detected, the tamper mask can once again be used to identify *void* recovery values by undoing the shuffle operation on marked pixels. As we only store one copy of the cover however, the only way to fill in these *voids* is by averaging over the neighbors. Once all *voids* have been filled in, the tampered areas can be restored. We do not do any sharping here, as the reconstruction image is based on a full-size version of the original image.

3.4 Discussion of the security aspects

We secure both *HiLoSpatial* and *HiResSpatial* by employing randomization with a PRNG and a seed K, meaning the randomization can only be reproduced if you have access to K. Using the output of the PRNG, authentication bits are generated and watermark bits are shuffled, such that an attacker cannot predict the values or locations to perform targeted attacks. If an attacker knows the location of a_i, he can avoid it by only tampering the neighboring recovery values. If the locations of specific recovery values are known, an attacker can extract the recovery image, or also tamper with those values to compromise recovery.

For the authentication, we use 2-bit authentication values $a_i \in \{0, 1, 2, 3\}$ that are stored in the 2 least significant bits (LSBs). If a pixel storing a_i is tampered with and receives a random value a_i', the probability that $a_i' \neq a_i$ is:

$$P(a_i' \neq a_i) = \frac{3}{4} = 0.75.$$

In the case of *HiLoSpatial* specifically, for each block of 2×2 pixels, only one pixel stores an authentication value a_i. Let E be the event that a randomly chosen pixel in the block is tampered with, and D be the event that the tampering is detected. The probability of detecting a change when only one pixel is randomly altered is:

$$P(D) = P(E) \cdot P(a_i' \neq a_i),$$

where $P(E) = \frac{1}{4}$ (since only one out of four pixels in the block stores a_i), and $P(a_i' \neq a_i) = \frac{3}{4}$. Thus,

$$P(D) = \frac{1}{4} \cdot \frac{3}{4} = \frac{3}{16} = 0.1875,$$

or approximately 19%.

However, as the main purpose of these techniques is to recover anonymized areas, we do not need to detect every tampered pixel. Instead, if we detect some instances of tampering in an area, we assume that their direct neighbors are also tampered with. This assumption is tested in the experiments in the next section to verify whether we can accurately detect tampering. Note that *HiResSpatial* allows for a more fine-grained tamper detection, as each pixel is protected by an authentication value, with a 0.75 probability to detect a pixel being randomized.

3.5 Evaluation metrics

To evaluate image integrity, the ground truth cover image I and the changed image $\hat{I}$ with shapes $m \times n$ are typically compared. The Mean Square Error (MSE) as given in Eq. 2 averages the square difference between corresponding pixels in I and $\hat{I}$.

$$MSE\left(I, \hat{I}\right) = \frac{1}{n \cdot m} \sum_{x=1}^{m} \sum_{y=1}^{n} \left(I(x,y) - \hat{I}(x,y)\right) \tag{2}$$

The PSNR in Eq. 3 considers the ratio between the maximum value of a pixel L (255 for 8-bit images) and the measured error, expressed in decibel (dB).

$$PSNR\left(I, \hat{I}\right) = 10 \log \left(\frac{L^2}{MSE\left(I, \hat{I}\right)} \right) \tag{3}$$

The Structural Similarity Index Measure (SSIM) aims to measure the similarity between two images [30], and is given in Eq. 4. It measures similarity by comparing luminance (μ_x and μ_y), contrast (σ_x and σ_y) and structure (σ_{xy}) of corresponding windows x and y in images I and $\hat{I}$. The constants $c_1 = (0.01L)^2$ and $c_2 = (0.03L)^2$ stabilize the equation in the case of small denominators. In our experiments, we use 11×11 windows, where $x = y = 11$.

$$SSIM\,(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{4}$$

To evaluate tamper detection performance, we use the detection rate or recall R, as given in Eq. 5. It measures the number of true positives (tp), which are pixels with correctly detected tampering, with respect to the tp and the number of tampered pixels that go unnoticed (false negatives fn).

$$R = \frac{tp}{tp + fn} \tag{5}$$

4 Experiments

We conduct several experiments to investigate the viability of our approach. First, we investigate how well our authentication approach can detect tampering. Secondly, we explore how well the overall performance of both approaches is. For each experiment, we use five RGB 512×512 images that are commonly used in watermarking research: *Airplane, Baboon, Car, Peppers, Sailboat*, in Fig. 4. Additionally, we consider one larger image from the VisDrone drone image dataset [35], along with a 1024×1024 crop (in Fig. 5), to explore how well the tested algorithms scale if the number of pixels increases.

We implement *HiLoSpatial* and *HiResSpatial*, as well as the comparative work, in Python 3.12 on a laptop with an Intel i7-1365U processor and 16GB Random Access Memory (RAM). For comparison, we implement [26, 21, 17, 29] based on their respective paper descriptions and we optimize the resulting code to

(a) Airplane (b) Baboon (c) Car (d) Peppers (e) Sailboat

Fig. 4. 512×512 test images.

(a) (b) (c) (d) (e)

Fig. 5. Left: Base image (a) and two 50% cropping attacks: Replacing the cropped areas with randomized values (b) or zeroes (c). Right: 1920×1080 baseline large image (d), with 1024×1024 crop (e).

the best of our ability. For [6], we use the available code [1], which we first modify to allow non-square images as well as introduce some speed optimizations. Our code will be made publicly available[2].

We use cropping as a tamper mechanism to evaluate watermarking. This replaces a portion of the attacked image by other values, e.g. random values or zeroes (in Fig. 5). Unless otherwise specified, in our experiments we always crop a specific percentage of the overall area, centered in the middle of the image.

4.1 Tamper detection coverage

To investigate the coverage of our tamper detection approach, we watermark a zero-valued 512×512 image on which we perform cropping attacks. For various crop edge sizes, we replace the pixels at a random area in the image with randomized values. We then perform tamper detection, both with and without using Algorithm 1, and compute the recall. We also consider the variant of *HiLoSpatial* for 1-channel images, as this should verify the percentages mentioned in Sect. 3.4. Each instance is run for a 100,000 iterations. The results are given in Tab. 2, and indicate that our approach can detect 99% of all tampering for edge size 32 with *HiLoSpatial* and edge size 2 with *HiResSpatial*.

[1] `https://github.com/Riadh-Bouarroudj/Fragile-image-watermarking-with-recovery`
[2] `https://github.com/sirris-fragile-watermarking/CIMSS2025`

Table 2. Detection recall with (grow) or without (base) using the grow function, for *HiLoSpatial* (HiLo) and *HiResSpatial* (HiRes). c is the number of channels in an image.

			Edge size						
Name	c	Variant	1	2	4	16	32	128	256
HiLo	1	base	0.1873	0.4226	0.5738	0.7039	0.7267	0.7442	0.7471
		grow	0.1873	0.5224	0.8066	0.9578	0.9776	0.9916	0.9939
HiLo	3	base	0.2455	0.5548	0.7539	0.9239	0.9539	0.9767	0.9805
		grow	0.2455	0.6678	0.9013	0.9817	0.9914	0.9980	0.9990
HiRes	3	base	0.7527	0.7493	0.7502	0.7500	0.7500	0.7500	0.7500
		grow	0.7527	0.9961	0.9997	1.0000	1.0000	1.0000	1.0000

4.2 Watermarking cost

Furthermore, we investigate the cost of watermarking by investigating two aspects. Firstly, watermarking decreases the integrity of an image, as a portion of the image is used to store authentication and reconstruction information, rather than its original information. Secondly, watermarking introduces new processing steps resulting in additional computational overhead.

We measure the integrity cost by comparing the watermarked images with the original and measuring the MSE, PSNR and SSIM, for each of the 512×512 test images. We compute these results for every test image separately, then average the results. Additionally, we evaluate the processing overhead by measuring the latency of the watermark embedding step T_e and the tamper detection step T_d for both the 512×512 and larger test images. These measurement are done by respectively repeating the embedding and tamper detection steps 100 times and computing the average. We use the singular drone images in Fig. 5e and 5d for each of the larger dimensions to limit the growing processing time and because embedding and tamper detection speed should not depend on the content of the source image. Tab. 3 gives the results, with the best values in bold. Clearly, *HiLoSpatial* and *HiResSpatial* both considerably surpass related work when considering latency, with the sole exception of [21] (who have a very high MSE), while maintaining a very small impact on image integrity.

Table 3. Watermark embedding impact with regards to average image integrity and computation time.

	Shape: 512×512					1024×1024		1080×1920	
Approach	MSE	PSNR	SSIM	T_e (s)	T_d (s)	T_e (s)	T_d (s)	T_e (s)	T_d (s)
HiLo	2.47	44.20	**0.99**	**0.02**	0.11	0.08	0.40	**0.15**	0.74
HiRes	2.46	44.23	**0.99**	0.03	0.21	0.11	0.76	0.22	1.44
[26]	2.50	44.15	**0.99**	2.91	2.58	8.22	7.37	15.50	13.82
[21]	37.75	32.37	0.89	**0.02**	**0.05**	**0.07**	**0.14**	**0.15**	**0.28**
[17]	9.84	38.21	0.97	2.35	0.67	6.82	1.93	12.76	3.72
[6]	**1.79**	**45.62**	**0.99**	2.34	1.44	7.47	4.52	13.36	8.37
[29]	2.54	44.08	**0.99**	1.17	11.42	4.38	242.46	7.60	890.44

Note that T_d may increase slightly in the case of tampering. For *HiLoSpatial* and *HiResSpatial*, the grow function iterates over every pixel to check whether it has been tampered with. In the case of tampering, an additional step to label its neighbors is done. To provide a baseline, we consider images without tampering.

4.3 Tamper recovery

After measuring what the cost of watermarking is, we can measure how well the watermarking mechanism is suited for both tamper detection and recovery. We test this by doing tamper detection and recovery experiments for all 512×512 test images and averaging results. To emulate tampering, we range a cropping attack from 10% to 75%, replacing the cropped area with random values or zeroes. Tab. 4 gives the tamper detection performance, while Tab. 5 gives the recovery results. Our approaches consistently match or surpass the best related work, while most other approaches only work reliably for low amounts of randomized tampering and quickly fail in the case of zero tampering[26, 21, 17, 29].

Table 4. Recall of the tamper detection after cropping areas of the image.

	Center crop random				Center crop zero			
Approach	10%	25%	50%	75%	10%	25%	50%	75%
HiLo	1.0000	1.0000	0.9987	0.9989	1.0000	1.0000	0.9987	0.9989
Gray	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
[26]	1.0000	1.0000	1.0000	1.0000	0.0124	0.0000	0.0110	0.0135
[21]	0.8182	0.8148	0.8270	0.8437	0.9609	0.8338	0.3184	0.0204
[17]	0.9995	0.9998	0.9996	0.9997	0.0124	0.0000	0.0110	0.0135
[6]	0.9997	1.0000	1.0000	1.0000	0.9997	1.0000	0.9999	1.0000
[29]	1.0000	1.0000	1.0000	1.0000	0.9993	1.0000	0.6698	0.3099

Table 5. Tamper correction with respect to the original image without watermarking for increasing amounts of tampering.

	Center crop random											
	10%			25%			50%			75%		
Approach	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
Gray	151.86	27.45	0.96	365.23	23.56	0.92	647.58	21.00	**0.85**	993.41	18.94	**0.72**
HiLo	23.86	34.44	0.97	60.56	30.41	0.93	**200.78**	**25.42**	0.82	680.22	19.89	0.59
[26]	23.04	34.61	0.97	72.08	29.64	0.93	364.08	22.63	0.79	5838.06	10.61	0.28
[21]	223.32	24.65	0.81	682.60	19.80	0.66	2221.34	14.67	0.28	4679.68	11.45	0.09
[17]	36.85	32.49	0.95	76.00	29.39	0.91	245.92	24.38	0.76	762.88	19.38	0.49
[6]	44.40	31.79	0.96	94.27	28.56	0.91	227.82	24.86	0.80	**410.47**	**22.21**	0.65
[29]	**16.58**	**35.98**	**0.98**	**35.73**	**32.67**	**0.96**	2937.63	13.54	0.73	8743.79	8.82	0.35
	Center crop zero											
	10%			25%			50%			75%		
Approach	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM
Gray	151.86	27.45	0.96	365.21	23.56	0.92	647.58	21.00	**0.85**	993.32	18.94	**0.72**
HiLo	**23.86**	**34.44**	**0.97**	**60.57**	**30.41**	**0.93**	**201.99**	**25.40**	0.82	686.52	19.85	0.58
[26]	2485.01	14.27	0.88	6253.86	10.24	0.72	11851.11	7.49	0.44	17673.73	5.81	0.18
[21]	221.24	24.75	0.82	1869.77	15.48	0.56	10929.14	7.86	0.14	20796.92	5.11	0.02
[17]	2489.59	14.26	0.86	6259.35	10.23	0.70	11878.35	7.48	0.43	17638.16	5.81	0.18
[6]	45.26	31.70	0.95	94.33	28.56	0.91	228.37	24.85	0.80	**410.27**	**22.21**	0.65
[29]	2471.16	14.29	0.88	6253.88	10.24	0.72	11918.27	7.47	0.45	17752.41	5.79	0.18

5 Discussion

5.1 Tamper detection

Tab. 2 and Tab. 4 present the tamper detection performance of *HiLoSpatial* and *HiResSpatial* and a comparison with the related work. Tab. 2 reveals that

while our authentication approach is not well suited to detect pixel-scale disturbances, it quickly scales to a very high detection rate. Considering 3-channel images, 99% of all 32×32-sized tampering is already detected using *HiLoSpatial*, while *HiResSpatial* detects all tampering starting from 16×16 attack windows. Clearly using the grow function in Alg. 1 significantly increases tamper detection performance. For *HiLoSpatial* the added value decreases as the tamper area increases, but as *HiResSpatial* is stuck at a maximum 75% detection rate for random tampering, it remains very relevant. The specific case of 1-pixel tampering for 1-channel *HiLoSpatial* also results in a tamper detection of 18.73%, which confirms the accuracy of our estimates in Sect. 3.4. The comparison in Tab. 4 clearly indicates our results match and in several cases surpass the state of the art. *HiLoSpatial* has 99.9% tamper detection rate which is sufficient. Rajput *et al.* [21] has in many cases 82% and the other approaches all work well for the random cropping case. However, for zero cropping, in which attacked pixels are replaced with zeroes, most tamper detection mechanism break down. This is due to the fact that [26, 17, 21, 29] all suffer from the problem described in Sect. 2.4. As the authentication function $a(0) = 0$, the tampering is hardly detected. For [21], the authentication is based on the median of four pixel values with randomly determined locations. As the amount of tampering increases the probability that the majority of these pixels is tampered with increases. That explains that while the approach works for limited tampering, it completely breaks down when tampering increases.

For the specific case of anonymization in a UAS surveillance flow, our approach is very well suited, as it excels at detecting areas of tampering. Since anonymization will always target an area, e.g. a license plate or face, which encompasses many pixels, our approaches will not suffer from the reduced performance they face when detecting tampering on individual, isolated pixels.

Overall, our proposed techniques succeed in accurately and reliably detecting tampering for cropping-based attacks, even in cases where related work fails.

5.2 Tamper recovery

In order to evaluate the tamper recovery, we consider random and zero cropping attacks in Tab. 5, which we additionally visualize in Fig. 7 and 8. As shown in Sect. 5.1, all related work except for Bouarroudj *et al.* [6] fails to do reliable tamper detection for zero cropping, which means no recovery can be done. Consequently, we will mostly consider the best-case scenario of random cropping for comparisons. Clearly, [29] greatly outperforms all other approaches for sub-50% tampering. From 50%, which is a little earlier than described in the corresponding paper, their performance drastically deteriorates. In all other cases, our *HiLoSpatial* achieves the best MSE and PSNR. Only for 75% tampering its SSIM is surpassed by [6] and *HiResSpatial*. This is because for this tampering amount, there only remains a small portion of valid recovery pixels.

In that case, [6] and *HiResSpatial* are slightly better at representing the overall structure, contrast and luminance. Overall [6] has performance comparable to *HiLoSpatial*. As expected, *HiResSpatial* achieves worse results for MSE and

(a) Original (b) Anonymized (c) HiLo (d) HiRes

Fig. 6. Recovery of details using *HiLoSpatial* and *HiResSpatial*.

PSNR, because it does not recover color information. It however remains competitive with regards to SSIM and even has the best SSIM for $\geq 50\%$ tampering. This indicates that while the color information is lost, *HiResSpatial* excels at retaining structural information, meaning that the reconstructed image will be the sharpest. Or, in other words, structural details will be much clearer, which helps when trying to read a small license plate or make out specific facial features in UAS footage. This principle is demonstrated for a 256×256 crop of the VisDrone image in Fig. 6, where *HiResSpatial* is able to completely restore the structural information of the license plate.

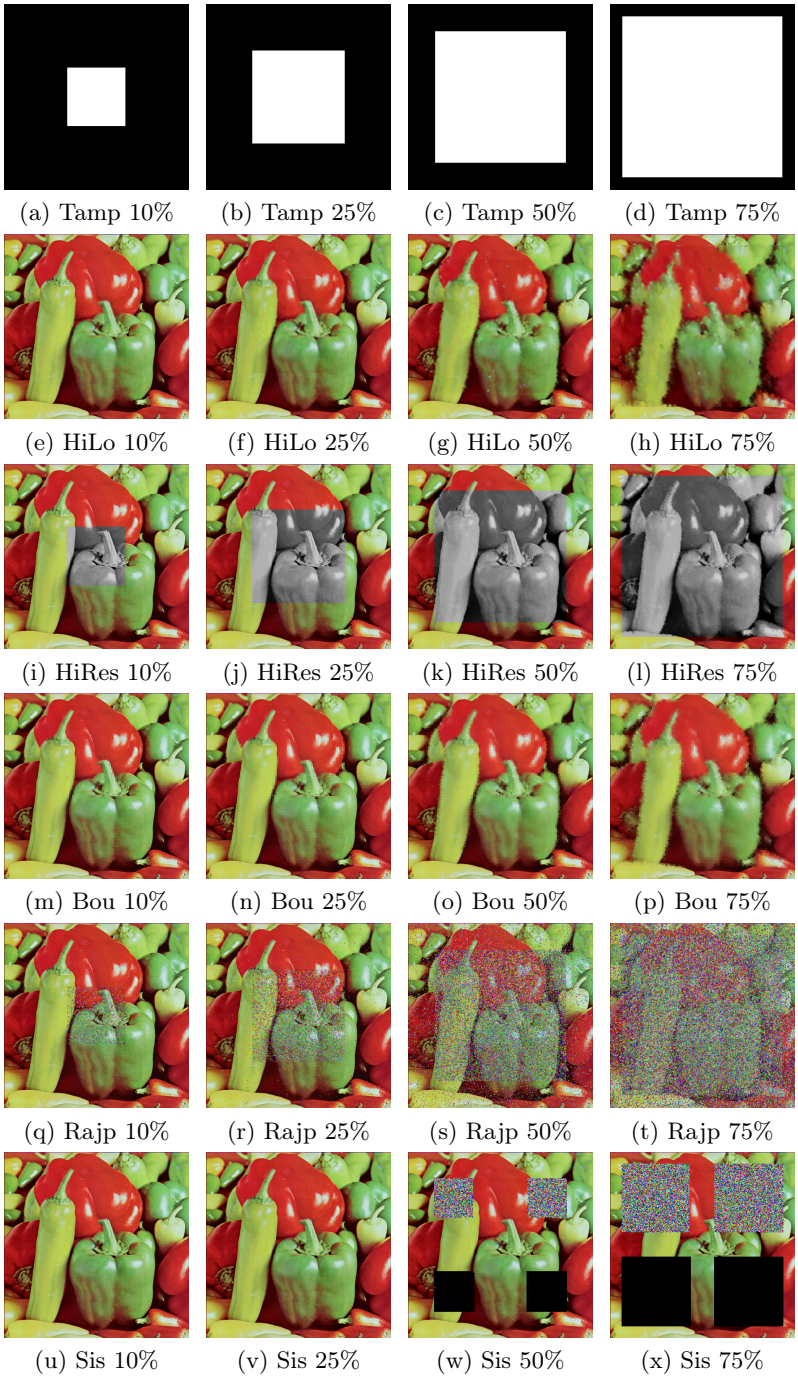
5.3 Latency performance

Tab. 3 gives both the impact of embedding on 512×512 images as well as the latency of watermark embedding and tamper detection. It is clear that our proposed techniques are only slightly surpassed by [6] with regards to maintaining image integrity while watermarking. Solely [21] has a higher impact on image integrity, due to the approach needing the 4 LSBs to store recovery bits. When considering the latency, this median-based technique is also the only technique that has lower latencies. This can mostly be attributed to the very simple process, i.e., the technique only has to randomly shuffle and store 4 lower-resolution copies. As the image size increases, the latencies do as well. It is clear that our approaches significantly outpace related work, with the exception of [21]. However, [21] has to compensate for its speed by its considerable loss in image integrity, after both embedding and recovery. Overall, our techniques sport among the best integrity retention ones while significantly outpacing nearly all related work.

5.4 Security

As discussed in Sect. 2.4, nearly all related work is unsuitable for security applications due to insecure or lack of shuffling. We mitigate this by applying randomization in three instances:

– The location of recovery values to ensure that tampering an area does not corrupt all corresponding recovery values, and prevent attackers from guessing where specific recovery values are stored.



(a) Tamp 10% (b) Tamp 25% (c) Tamp 50% (d) Tamp 75%

(e) HiLo 10% (f) HiLo 25% (g) HiLo 50% (h) HiLo 75%

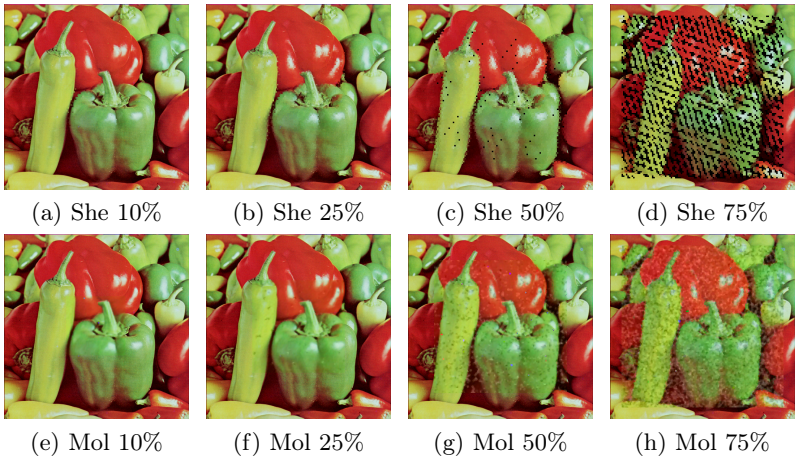
(i) HiRes 10% (j) HiRes 25% (k) HiRes 50% (l) HiRes 75%

(m) Bou 10% (n) Bou 25% (o) Bou 50% (p) Bou 75%

(q) Rajp 10% (r) Rajp 25% (s) Rajp 50% (t) Rajp 75%

(u) Sis 10% (v) Sis 25% (w) Sis 50% (x) Sis 75%

Fig. 7. Recovery results on *Peppers* with varying degrees of tampering. The tamper masks are given in the top row (Tamp). Results for *HiLoSpatial* (HiLo), *HiResSpatial* (HiRes), Bouarroudj *et al.* (Bou), Rajput *et al.* (Rajp) and Sisaudia *et al.* (Sis).



(a) She 10% (b) She 25% (c) She 50% (d) She 75%

(e) Mol 10% (f) Mol 25% (g) Mol 50% (h) Mol 75%

Fig. 8. Continutation of recovery results on *Peppers* for Sisaudia *et al.* (Sis), Shebab *et al.* (She) and Molina-Garcia *et al.* (Mol).

- Authentication values to prevent targeted attacks that avoid changing authentication values to go unnoticed.
- The location of authentication values to prevent attackers from specifically avoiding the LSBs of authentication pixels.

As shown in the experiments, our solutions are robust to attacks. There is however one caveat. Currently, our randomization is based on Numpy's implementation. While fast, their random number generation is not cryptographically secure. To further ensure thorough security, the solution should apply Cryptographically Secure Pseudorandom Number Generators (CSPRNGs) instead.

6 Conclusion and future work

In this paper we present *HiLoSpatial* and *HiResSpatial*, two fragile watermarking techniques that allow respectively full-colour and grayscale recovery for 3-channel images after tampering. These match and in many cases surpass related work with regards to tamper detection performance and image integrity retention after watermarking and recovery. Moreover, they significantly outpace related work when considering embedding and detection latency. This makes them very suitable for deployment in real-time on-device image processing for UAS applications supporting the security of critical infrastructure while protecting personal data. Interesting future research directions include investigating how to apply these techniques to video streams and how to optimally implement them on-device. Additionally, we need to investigate what lightweight CSPRNGs are suitable for this application, in an effort to maximize throughput while maintaining security.

Acknowledgments. This work is conducted in the context of the SINTRA project (`https://sintra-ai.odoo.com/`), a project labeled by ITEA4 under project no. 22006,

with funding support from Innoviris Brussels and VLAIO Flanders in Belgium. The project features cooperation with over 30 partners from industry and academia and 4 countries, i.e. Belgium, Finland, The Netherlands and Türkiye.

Disclosure of Interests. The authors have no competing interests.

References

1. Arnold, V., Avez, A.: Problèmes ergodiques de la mécanique classique. Monographies internationales de mathématiques modernes, Gauthier-Villars (1967)
2. Benrhouma, O., Hermassi, H., Belghith, S.: Security analysis and improvement of an active watermarking system for image tampering detection using a self-recovery scheme. Multimed. Tools Appl. **76**(20), 21133–21156 (Oct 2017)
3. Bolourian Haghighi, B., Taherinia, A.H., Harati, A.: Trlh: Fragile and blind dual watermarking for image tamper detection and self-recovery based on lifting wavelet transform and halftoning technique. Journal of Visual Communication and Image Representation **50**, 49–64 (2018)
4. Bonetto, M., Korshunov, P., Ramponi, G., Ebrahimi, T.: Privacy in mini-drone based video surveillance. In: 2015 11th IEEE FG. vol. 04, pp. 1–6 (2015)
5. Bouarroudj, R., Souami, F., Belalla, F.Z.: Reversible fragile watermarking for medical image authentication in the frequency domain. In: 2023 2nd IC2EM. vol. 1, pp. 1–6 (2023)
6. Bouarroudj, R., Souami, F., Bellala, F.Z.: Fragile watermarking for medical image authentication based on dct technique. In: 2023 5th PAIS). pp. 1–6 (2023)
7. Bouarroudj, R., Souami, F., Zohra Bellala, F., Zerrouki, N.: A reversible fragile watermarking technique using fourier transform and fibonacci q-matrix for medical image authentication. Biomedical Signal Processing and Control **92**, 1–11 (2024)
8. Bravo-Solorio, S., Calderon, F., Li, C.T., Nandi, A.K.: Fast fragile watermark embedding and iterative mechanism with high self-restoration performance. Digital Signal Processing **73**, 83–92 (2018)
9. Council of European Union: Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation) (2016), `https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng`
10. Gohari, A., Ahmad, A.B., Rahim, R.B.A., Supa'at, A.S.M., Abd Razak, S., Gismalla, M.S.M.: Involvement of surveillance drones in smart cities: A systematic review. IEEE Access **10**, 56611–56628 (2022)
11. Greenwood, W.W., Lynch, J.P., Zekkos, D.: Applications of uavs in civil infrastructure. Journal of Infrastructure Systems **25**(2), 04019002 (2019)
12. Gul, E., Ozturk, S.: A novel triple recovery information embedding approach for self-embedded digital image watermarking. Multimed. Tools Appl. **79**(41), 31239–31264 (Nov 2020)
13. Hussan, M., Parah, S.A., Jan, A., Qureshi, G.J.: Self-embedding framework for tamper detection and restoration of color images. Multimed. Tools Appl. **81**(13), 18563–18594 (May 2022)
14. Lee, S.J., Jung, S.H.: A survey of watermarking techniques applied to multimedia. In: ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No.01TH8570). vol. 1, pp. 272–277 (2001)

15. Mandirola, M., Casarotti, C., Peloso, S., Lanese, I., Brunesi, E., Senaldi, I.: Use of uas for damage inspection and assessment of bridge infrastructures. International Journal of Disaster Risk Reduction **72**, 102824 (2022)
16. Maximov, M., Elezi, I., Leal-Taixe, L.: Ciagan: Conditional identity anonymization generative adversarial networks. In: Proc. of the IEEE/CVF CVPR (June 2020)
17. Molina-Garcia, J., Garcia-Salgado, B.P., Ponomaryov, V., Reyes-Reyes, R., Sadovnychiy, S., Cruz-Ramos, C.: An effective fragile watermarking scheme for color image tampering detection and self-recovery. Signal Processing: Image Communication **81**, 115725 (2020)
18. Proença, H.: The uu-net: Reversible face de-identification for visual surveillance video footage. IEEE TCSVT **32**(2), 496–509 (2022)
19. Qin, C., Ji, P., Chang, C.C., Dong, J., Sun, X.: Non-uniform watermark sharing based on optimal iterative btc for image tampering recovery. IEEE MultiMedia **25**(3), 36–48 (2018)
20. Raj, N.R.N., Shreelekshmi, R.: A survey on fragile watermarking based image authentication schemes. Multimed. Tools Appl. **80**(13), 19307–19333 (May 2021)
21. Rajput, V., Ansari, I.A.: Image tamper detection and self-recovery using multiple median watermarking. Multimed. Tools Appl. **79**(47), 35519–35535 (Dec 2020)
22. Renklier, A., Öztürk, S.: Image authentication and recovery: Sudoku puzzle and md5 hash algorithm based self-embedding fragile image watermarking method. Multimed. Tools Appl. **83**(5), 13929–13951 (Feb 2024)
23. Rey, C., Dugelay, J.L.: A survey of watermarking algorithms for image authentication. EURASIP Journ. on Adv. in Sign. Proc. **2002**(6), 613–621 (Jun 2002)
24. Santos de Melo, R.R., Costa, D.B., Álvares, J.S., Irizarry, J.: Applicability of unmanned aerial system (uas) for safety inspection on construction sites. Safety Science **98**, 174–185 (2017)
25. Shakhatreh, H., Sawalmeh, A.H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N.S., Khreishah, A., Guizani, M.: Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. IEEE Access **7**, 48572–48634 (2019)
26. Shehab, A., Elhoseny, M., Muhammad, K., Sangaiah, A.K., Yang, P., Huang, H., Hou, G.: Secure and robust fragile watermarking scheme for medical images. IEEE Access **6**, 10269–10278 (2018)
27. Singh, D., Singh, S.K.: Dct based efficient fragile watermarking scheme for image authentication and restoration. Multimed. Tools Appl. **76**(1), 953–977 (Jan 2017)
28. Singh, D., Singh, S.K.: Block truncation coding based effective watermarking scheme for image authentication with recovery capability. Multimed. Tools Appl. **78**(4), 4197–4215 (Feb 2019)
29. Sisaudia, V., Vishwakarma, V.P.: Approximate regeneration of image using fragile watermarking for tamper detection and recovery in real time. Multimed. Tools Appl. **83**(25), 66299–66318 (Jul 2024)
30. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE TIP **13**(4), 600–612 (April 2004)
31. Yamaç, M., Ahishali, M., Passalis, N., Raitoharju, J., Sankur, B., Gabbouj, M.: Multi-level reversible data anonymization via compressive sensing and data hiding. IEEE TIFS **16**, 1014–1028 (2021)
32. Ye, M., Shen, W., Zhang, J., Yang, Y., Du, B.: Securereid: Privacy-preserving anonymization for person re-identification. IEEE TIFS **19**, 2840–2853 (2024)
33. Zhai, L., Guo, Q., Xie, X., Ma, L., Wang, Y.E., Liu, Y.: A3gan: Attribute-aware anonymization networks for face de-identification. In: Proceedings of the 30th ACM MM. p. 5303–5313. MM '22, ACM, New York, NY, USA (2022)

34. Zhang, J., Ye, M., Yang, Y.: Learnable privacy-preserving anonymization for pedestrian images. In: Proceedings of the 30th ACM MM. p. 7300–7308. MM '22, ACM, New York, NY, USA (2022)
35. Zhu, P., Wen, L., Du, D., Bian, X., Fan, H., Hu, Q., Ling, H.: Detection and tracking meet drones challenge. IEEE TPAMI **44**(11), 7380–7399 (2021)
36. Zia, U., McCartney, M., Scotney, B., Martinez, J., AbuTair, M., Memon, J., Sajjad, A.: Survey on image encryption techniques using chaotic maps in spatial, transform and spatiotemporal domains. Intl. Journ. of Inf. Sec. **21**(4), 917–935 (Aug 2022)