Diffusion-Based Image Compression in Steganography

Markus Mainberger[1], Christian Schmaltz[1], Matthias Berg[2], Joachim Weickert[1], Michael Backes[2]

[1] Mathematical Image Analysis Group,
Faculty of Mathematics and Computer Science, Campus E1.7
Saarland University, 66041 Saarbrücken, Germany
{mainberger,schmaltz,weickert}@mia.uni-saarland.de

[2] Information Security and Cryptography Group,
Faculty of Mathematics and Computer Science, Campus E1.1
Saarland University, 66041 Saarbrücken, Germany
{berg,backes}@cs.uni-saarland.de

Abstract. We demonstrate that one can adapt recent diffusion-based image compression techniques such that they become ideally suited for steganographic applications. Thus, the goal is to embed secret images within arbitrary cover images. We hide only a small number of characteristic points of the secret in the cover image, while the remainder is reconstructed with edge-enhancing anisotropic diffusion inpainting. Even when using significantly less than 1% of all pixels as characteristic points, sophisticated shapes of the secret can be clearly identified. Selecting more characteristic points results in improved image quality. In contrast to most existing approaches, this even allows to embed large colour images into small grayscale images. Moreover, our approach is well-suited for uncensoring applications. Our evaluation and a web demonstrator confirm these claims and show advantages over JPEG and JPEG 2000.

1 Introduction

In a time that is characterized by devices that allow anyone at anytime to take and share high resolution digital images, the need for image compression codecs with high compression rates is more important than ever. A promising recent class of image compression codecs relies on inpainting techniques based on partial differential equations (PDEs); see e.g. [1]. In contrast to conventional methods such as JPEG, PDE-based image compression does not require transformations to the frequency domain: It simply stores a small fraction of characteristic pixels. In the decoding step, the missing pixels are reconstructed by the inpainting effect of the PDE. PDE-based approaches can outperform JPEG and even its sophisticated successor JPEG 2000 [2,3]. This oberservation holds all the more for perceptual quality metrics as shown in [4]. However, they have not been adapted to specific application scenarios outside classical image compression so far.

In this paper, we consider one of those scenarios, namely *steganography*. Steganography is the art of hiding the *presence* of an embedded message (called *secret*) within a seemingly harmless message (called *cover*). It can be seen as the complement of *cryptography*, whose goal is to hide the *content* of a message. Media data such as images currently constitute the most widely used cover types for practical steganographic systems, mainly because they contain much redundant data. However, most existing approaches only embed single words or a short text within a cover image. We show that, by exploiting the potential of PDE-based image compression codecs, our approach can hide complete images.

Apart from steganography, embedding messages within other messages is also useful in applications such as digital content protection and copyright management: A content provider can distribute redacted media data already containing the missing information. Afterwards the missing information can be selectively disclosed to paying customers by simply supplying the password.

Example algorithms to embed data in images are *Jsteg* [5], *F5* [6], and *YASS* [7]. *Jsteg* replaces least significant bits with those to embed, which is vulnerable to histogram attacks [8]. *F5* implements matrix encoding to improve the embedding efficiency and a permutation to spread the changes over the whole steganogram. *YASS* employs error correcting codes to be robust against *JPEG* compression. Often, steganographic methods also focus on hiding information inside specific file formats such as JPEG [7,9] or JPEG 2000 [10,11].

Our Contribution. In this article, we show how PDE-based image compression must be adapted such that it meets the requirements occurring in steganography. We call the resulting novel method *Steganograhy with Diffusion Inpainting (SDI)*. First we select a very small number of characteristic points of the given secret image. Thereby, we carefully combine insights from several recent papers to obtain a fast compression codec which yields high quality results and small file sizes at the same time. Then, we encode the obtained bitstream by employing a state of the art entropy coder. Before embedding this data, we encrypt it to avoid detectable patterns that might be a result of the compression.

Note that, for steganographic purposes, it is desirable to keep the number of characteristic points – and thus the amount of data that must be hidden – as small as possible. Since experiments have demonstrated that diffusion inpainting is well-suited to extremely high compression rates [2], it fits perfectly to steganography. Even with less than 1% of all pixels as characteristic points, the shapes of the secret image can be clearly identified.

Interestingly, our SDI approach even allows to black out a region of an image and embed its former content in the remainder of the image. Using the correct password, the censored part can be uncovered at any time. Diffusion-based inpainting is particularly well-suited for this application, as the boundary of the censored region provides additional information which facilitates the reconstruction process. A web demonstrator for hiding and censoring is available at `http://stego.mia.uni-saarland.de`.

Related Work. To the best of our knowledge, inpainting methods have not been used for steganographic purposes before. In [12] techniques are developed

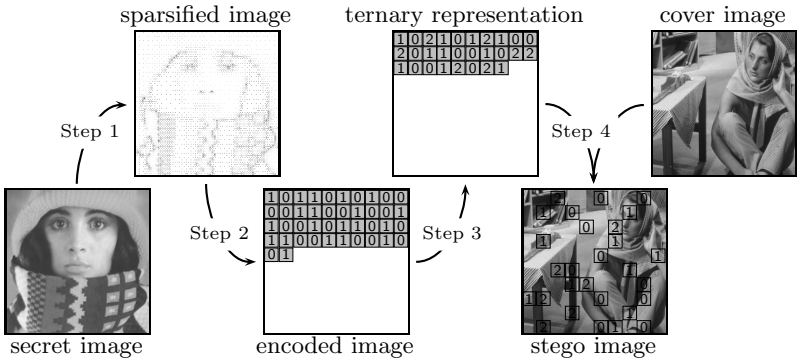

Fig. 1. Schematic overview of the main steps of our steganography method: Steps 1 and 2 illustrate the encoding of the PDE-based image compression codec. Steps 3 and 4 depict the steganographic preparation and embedding phases.

to prevent an unexpected user from eliminating objects in videos by inpainting. This is achieved by segmenting the objects, followed by a steganographic embedding. However, this has nothing to do with our approach, which tailors diffusion inpainting for compact data representation in steganography.

Paper Structure. Section 2 illustrates our diffusion-based compression algorithm. Based on this algorithm, we explain how to embed secret images within arbitrary cover images in Section 3. In Section 4, we describe our performance measurements before concluding the paper in Section 5.

2 PDE-based Image Compression

2.1 Basic Concept

While anisotropic diffusion processes have been popular for denoising images for many years, their use in image compression is relatively new and a topic of ongoing research. Compared to conventional compression methods such as *JPEG* or *JPEG 2000*, diffusion-based image compression offers a number of advantages: It has a very intuitive physical interpretation, it is well-suited for extremely high compression rates, where it can outperform *JPEG* and even *JPEG 2000* [2,3], and it can be used in a generic way for various types of data – including 2-D images, 3-D data sets, surface data and videos.

For simplicity, we consider a greyscale image, i.e. a function $f : \Omega \to \mathbb{R}$, where $\Omega \subset \mathbb{R}^2$ denotes a rectangular image domain. The extension to colour images does not create specific problems. The idea behind diffusion-based image compression is to store only the grey-values of a small subset $K \subset \Omega$ [1]. Using the pixels in K as Dirichlet boundary data, we reconstruct f in the unspecified domain $\Omega \setminus K$ as steady state ($t \to \infty$) of the partial differential equation (PDE)

$$\partial_t u = \operatorname{div}(\boldsymbol{D}(\boldsymbol{\nabla} u_\sigma)\, \boldsymbol{\nabla} u) \;, \tag{1}$$

where $\nabla = (\partial_x, \partial_y)^\top$ is the spatial nabla operator, and div the corresponding divergence operator. This PDE is known as *edge-enhancing anisotropic diffusion (EED)* [13]. The diffusion process is steered by the positive definite 2×2 *diffusion matrix* $\boldsymbol{D}$. It depends on the gradient of the Gaussian-smoothed version u_σ of the image u, where σ is the standard deviation of the Gaussian. The direction of ∇u_σ is along the steepest ascent, i.e. perpendicular to image edges, and its magnitude $|\nabla u_\sigma|$ measures the strength of the edge (contrast). The two eigenvectors of $\boldsymbol{D}$ are orthogonal resp. parallel to ∇u_σ, and their eigenvalues are given by $\mu_1 = 1$ and $\mu_2 = \frac{1}{\sqrt{1+|\nabla u_\sigma|^2/\lambda^2}}$. Thus, along image edges, $\mu_1 = 1$ ensures that full diffusion is performed, while the decreasing behaviour of μ_2 w.r.t. $|\nabla u_\sigma|$ reduces diffusion across edges with high contrast. The parameter $\lambda > 0$ allows to steer this contrast dependence.

2.2 Step 1: Finding Characteristic Pixels

Since steganographic methods are often applied in situations with limited resources, the compression algorithm has to be fast. Compared to methods such as [2], which are tuned to yield optimal *quality*, we thus give greater priority to the *running time*. However, real-time capable methods are often designed for more restricted application areas, e.g. to cartoon-like images [3]. Since our codec aims at combining interactive running times with suitability for a large class of images, we have to modify ideas from [2] and [3] and combine them with results from [14]. Moreover, our compression codec should work particularly well for high compression rates such that we are also able to hide a large colour image within a much smaller greyscale image. This also allows not to exploit the maximal available space such that the secret information is only stored in a fraction of all pixels, in order to prevent an observer from detecting the secret information.

To fulfil these requirements, we first extract a set of characteristic pixels which will be used for inpainting in the recovering step (Figure 1, Step 1), followed by saving these points in a compact representation, see next section (Figure 1, Step 2). We rely on an adaptive subdivision method similar to the one chosen in [2], but refrain from time-consuming optimisation steps. Instead, we exploit the theoretical results obtained in [14] to speed up the method.

As in [2], our subdivision method starts with the rectangle defined by the whole image domain, or the censored image part, respectively. The rectangle is recursively split into smaller rectangles until a predefined stopping criterion is fulfilled. The final inpainting mask K, i.e., the pixel locations used in the reconstruction, is given by the four corner points and the centre of each rectangle. This allows to efficiently encode pixel positions; see Section 2.3. Moreover, the algorithm can adaptively store more pixels in significant structures by using a finer subdivision instead of investing many points in unimportant image areas.

In [2], splitting a rectangle is stopped as soon as the interpolation error within the rectangle is smaller than a threshold. Thus, a rectangle is split whenever decoding would lead to an inadequate reconstruction in this area. The drawback of this natural stopping criterion is that its evaluation requires to inpaint image parts while compressing. Thus, this method is too slow for our purpose.

Therefore, we exploit the analytical result obtained in [14] for homogeneous diffusion inpainting to speed up the subdivision. In this paper, it is proven that the pixel positions should be chosen depending on $|\Delta f|$, the magnitude of the Laplacian of the original image f. Even though we prefer EED over homogeneous diffusion due to its superior inpainting quality, experiments indicate that this choice still yields good results. Our new subdivision scheme exploits this idea and works as follows: During the subdivision process, we assign each rectangle the sum over the Laplace magnitudes $|\Delta f_\sigma|_i$ of all pixels i within the rectangle. We denote these sums, which can be computed efficiently using the idea of integral images [15], by $D(R)$. Starting with the rectangle defined by the whole image domain, the rectangle that corresponds to the largest sum is subdivided in the middle of the x or y direction, whichever is larger. This rectangle can be found efficiently by utilising a priority queue, where the rectangle R has the priority $-D(R)$. After each subdivision step, the two new rectangles are included into the queue.

Obviously, the number of subdivisions s is related to the size of the encoded image. Therefore, we can adapt s to the desired compression ratio. We perform a binary search for s to approximate this compression rate.

2.3 Step 2: Compact Representation

To store the inpainting mask K obtained in the last section, it is sufficient to represent the splitting decisions with a binary tree structure. We encode this tree in the image header: First the depth range of the tree is stored, followed by single bits indicating the splitting decisions. The file header also contains a few bytes for properties such as image size and number of channels.

The grey or colour values of the selected points are quantised regularly into q possible values. In contrast to [2], where a time-consuming optimisation of q is advocated to achieve best quality, we use the constant value $q = 32$. To further speed up the algorithm, we employ the *LPAQ* [16] coder to compress the brightness data instead of the slower *PAQ* [16] coder proposed in [2]. Finally, the binary strings containing the compressed pixel values and the file header are concatenated to obtain a bit stream representation of the secret.

To restore the image, the saved grey values are placed at the locations indicated by the stored splitting decisions, and the remainder of the image is inpainted using EED. For uncensoring, we can exploit additional information in the reconstruction step, namely the fact that (a part of) the boundary of the censored image is known. This has two consequences: First of all, characteristic points lying on known boundaries do not have to be saved, which reduces the size of the compressed file. Secondly, the known boundary pixels can be used as known points in the inpainting step, which improves the reconstruction quality.

3 Steganographic Preparation and Embedding

In this section we present the steganographic part our *SDI* algorithm, depicted as Step 3 and 4 in Figure 1. It allows to hide a bitstream of a restricted size, as

obtained at the end of the previous section, in a set of cover image pixels. An observer cannot detect whether the resulting image contains a hidden secret or not. Moreover, the secret bitstream can only be extracted with a password p.

3.1 Step 3: Steganographic Preparation

As a first steganographic step we need to add the size of the bitstream (in bytes) to its beginning to be able to separate secret data from cover data when extracting the bitstream again. Then we encrypt this concatenation using the *Advanced Encryption Standard* (*AES*) in *Cipher-block Chaining* (*CBC*) mode. This step further ensures the confidentiality of the secret data. The key used for this encryption is derived from password p by using the hash function SHA-256 as a random oracle [17]. Following the approach proposed in [18], we split the encrypted binary stream into blocks of 11 bits. Each block is transformed to 7 ternary bits (note that $2^{11} < 3^7$). This conversion is necessary to use *mod-3 matching* later on, which embeds a ternary bit in a cover pixel.

3.2 Step 4: Embedding

Next, we need to select the cover image pixels in which we want to hide the secret data. The simplest way would be to start at the beginning of the cover's data and to sequentially embed the ternary bits into the pixels of the cover. However, this approach simplifies the detection of the secret image since all modifications of the cover occur at its beginning, except the ternary stream is large enough to exhaust the cover's capacity. Therefore, we use a pseudo-random permutation ρ, inferable by anybody who knows the password p, to spread the ternary bits homogeneously over the cover's data. This ensures that each pixel of the cover image carries secret information with the same probability. Our embedding of secret data into an image relies on the assumption that the distribution of the least significant bits for each byte representing a cover pixel essentially corresponds to random noise. Hence, changing the pixel values by ± 1 does not change an image in a manner that is noticeable by the user. However, *LSB replacement*, i.e. overwriting the *least significant bits* with those that should be embedded, is vulnerable to statistical detection methods, as even values are never decreased and odd values are never increased [8]. *LSB matching* [19] instead randomly decides whether to increase or decrease a value, which bypasses these statistical tests. We use mod-3 matching which embeds a ternary bit t by modifying a pixel value v, such that $v \bmod 3 = t$. More precisely, if $v - t \bmod 3 = 1$, we set $v := v - 1$; if $v - t \bmod 3 = 2$, we set $v := v + 1$; otherwise we leave v unchanged. In the special cases where v already is 0 or 255, we skip over v as modifying completely saturated values might be noticeable in some scenarios. Furthermore, an embedding of t in a value $v \in \{1, 254\}$ might result in shrinkage with a completely saturated byte $\in \{0, 255\}$; see [6]. The extraction cannot distinguish such saturated stegobytes from skipped saturated image bytes. Therefore, we re-embed the respective ternary bit in the subsequent cover byte, until the result is not a saturated byte.

Neglecting those two restrictions, we can approximately embed one ternary bit per cover image pixel. That means if the cover image's data consist of n pixels, we can embed up to $t = 11 \cdot \lfloor \frac{n}{7} \rfloor$ bits, which amounts to up to $b = \lfloor \frac{t}{128} \rfloor$ *AES* blocks. Assuming 4 bytes of length information at the beginning of the stream, we can embed secret data of size up to $\frac{128 \cdot b}{8} - 4$ bytes in the cover image.

To extract the secret image from the cover, we simply invert the operations explained above: Given the password p, we can infer the pseudo-random permutation ρ. This allows us to extract the bits of the ternary representation in the correct order by calculating mod-3 for each value and skipping over completely saturated values as explained above. To determine how many pixels need to be processed, we decrypt the first 128 extracted bits with the key derived from password p. Then, by using the obtained length information we extract the remaining bits of the secret bitstream and decrypt it. Finally we apply the decompression algorithm from Section 2.3 to reconstruct the secret image.

3.3 Uncensoring Images

SDI as described above can immediately be applied to hide an image into another one. However, there are various modifications and extensions that allow the usage of SDI in other interesting fields of application.

For instance, SDI can be used for uncensoring applications where censored parts of an image are still contained in the same image and accessible by a password. However, it should not be detectable that there is any other information available than the visible one. In this scenario, only some minor adjustments are necessary. First of all, we store the location of the part of the image that is to be censored using a few additional bits in the header of the compressed image. Moreover, as explained in Section 2.3, we can exploit the fact that the boundary of the censored region is known. Additionally, the file header of our approach is much smaller than those of JPEG and JPEG 2000, which is an additional advantage over these approaches. This makes our diffusion-based image compression algorithm particularly well-suited for such uncensoring scenarios.

Even in cases where a significant part of the image is censored and only a small surrounding frame of the censored region is available for embedding, we still achieve good reconstruction results. We encourage the reader to verify this claim with our web demonstrator (see http://stego.mia.uni-saarland.de).

4 Experiments

For a quantitative evaluation of SDI, we consider the *mean squared error (MSE)*, which is defined as

$$\mathrm{MSE} := \frac{1}{M \cdot N} \sum_{m=1}^{M} \sum_{i=1}^{N} (f_{m,i} - u_{m,i})^2 \,, \tag{2}$$

where N is the number of image pixels, M is the number of channels, and $(f_{m,i})_{i=1..N}$ and $(u_{m,i})_{i=1..N}$ are the pixel values of the original secret image in channel m, pixel i and its reconstructed version, respectively.

Fig. 2. Colour test images: **(a) Left:** Cover image *cake*, 256×256 pixels. **(b) Right:** Secret image *knife*, 512×512 pixels.

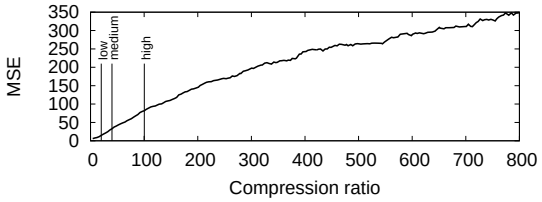

Fig. 3. Compression ratio versus mean squared error when our compression codec is applied to the test image *knife* (see Figure 2). The green rectangles mark the automatically computed ratios corresponding to the quality settings "high", "medium", and "low" of our web-based prototype.

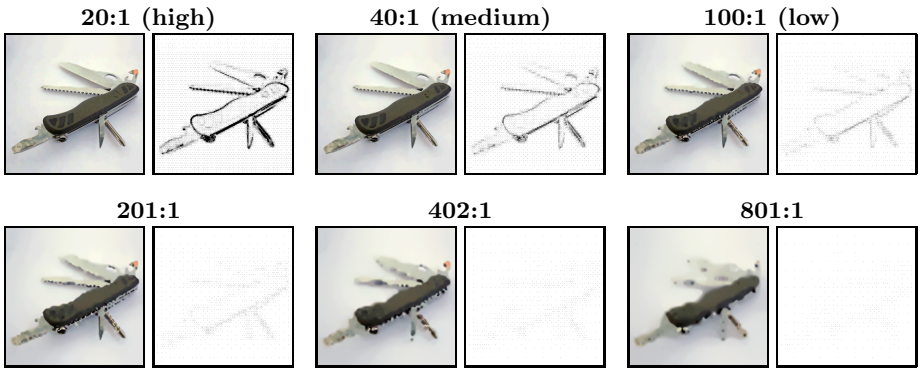

Fig. 4. Reconstructed images and corresponding inpainting masks for increasing compression ratios of the test image *knife* (see Figure 2). The first row depicts the results obtained with different quality settings offered by our web-based prototype.

Fig. 5. The information removed from the original image was hidden in the censored image. The magnifications on the top of the right side illustrate the reconstruction quality with different amounts of data. The images in the bottom row compare the performance of our codec with JPEG and JPEG 2000.

In Figure 2, we apply our method to two test images. Note that the secret image is four times larger than the cover image. Figures 3 and 4 reflect the respective results of the recovered secret depending on the number of stored secret pixels. As expected, the higher the compression ratio, the lower the quality. However, it is typically only necessary to recognise what was originally depicted. With SDI this is possible: Even for compression ratios as high as 800 : 1, we can clearly identify the secret to be a knife. This means we could hide this secret in the given cover roughly 40 times.

For the colour images from Figure 2, the embedding and extracting time on a standard PC lies in the order of seconds. If the speed of the hiding process is negligible, it is also possible to replace the suggested compression codec by the more sophisticated one from [2], which yields results of even higher quality at the same compression ratio.

The benefit of our codec over JPEG and JPEG 2000 can be seen in Figure 5: Storing the censored image part with JPEG or JPEG 2000 yields a much worse quality, even though our compression algorithm additionally stored the position of the censored image part, while the saved JPEG and JPEG 2000 image parts were manually pasted onto the right position.

5 Conclusions and Future Work

We have established a novel application field for recent PDE-based image compression techniques by adapting them to the needs of steganographic problems. In order to compress the image data in short time, high quality, and with small file sizes, it was necessary to combine a number of concepts and insights. This was integrated with carefully engineered steganographic preparation and embedding processes to end up with SDI, a novel technique for hiding secret image data in a cover image in a secure manner. Moreover, we have shown the usefulness of the SDI approach for uncensoring applications.

The success of these applications benefits heavily from the fact that diffusion-based image compression offers several properties that makes it preferable over classical codecs such as JPEG or JPEG 2000. This includes for instance its excellent performance for very high compression ratios, and its ability to exploit known boundary data in uncensoring applications. It appears that PDE-based compression and steganography constitute an ideal, hitherto unexplored match.

We are convinced that our work is not the end of the road: Due to the generic concept of diffusion-based inpainting, the SDI approach can be extended to a variety of different secret and cover formats, including 2-D images, 3-D data sets, surface data and videos. Even hiding small videos within a single image seems within range given our experiments. This is part of our ongoing research.

References

1. Galić, I., Weickert, J., Welk, M., Bruhn, A., Belyaev, A., Seidel, H.P.: Image compression with anisotropic diffusion. Journal of Mathematical Imaging and Vision **31** (2008) 255–269

2. Schmaltz, C., Weickert, J., Bruhn, A.: Beating the quality of JPEG 2000 with anisotropic diffusion. In Denzler, J., Notni, G., Süße, H., eds.: Pattern Recognition. Volume 5748 of Lecture Notes in Computer Science., Berlin, Springer (2009) 452–461
3. Mainberger, M., Bruhn, A., Weickert, J., Forchhammer, S.: Edge-based image compression of cartoon-like images with homogeneous diffusion. Pattern Recognition **44** (2011) 1859–1873
4. Galić, I., Zovko-Cihlar, B., Snježana, R.D.: Computer image quality selection between JPEG, JPEG 2000 and PDE compression. In: Proc. 19th International Conference on Systems, Signals and Image Processing, IEEE Computer Society Press (2012) 437–441
5. Upham, D.: JSteg. http://zooid.org/~paul/crypto/jsteg/ (1997)
6. Westfeld, A.: F5-a steganographic algorithm. In Moskowitz, I.S., ed.: Information Hiding. Volume 2137 of Lecture Notes in Computer Science., Berlin, Springer (2001) 289–302
7. Solanki, K., Sarkar, A., Manjunath, B.S.: Yass: Yet another steganographic scheme that resists blind steganalysis. In Furon, T., Cayre, F., Doërr, G., Bas, P., eds.: Information Hiding. Volume 4567 of Lecture Notes in Computer Science., Berlin, Springer (2007) 16–31
8. Westfeld, A., Pfitzmann, A.: Attacks on steganographic systems. In Pfitzmann, A., ed.: Information Hiding. Volume 1768 of Lecture Notes in Computer Science., Berlin, Springer (1999) 61–76
9. Jókay, M., Moravcík, T.: Image-based JPEG steganography. Tatra Mountains Mathematical Publications (2010) 65–74
10. Su, P.C., Ku, C.C.J.: Steganography in JPEG2000 compressed images. IEEE Transactions on Consumer Electronics **49** (2003) 824 – 832
11. Zhang, I., Wand, H., Wu, R.: A high-capacity steganography scheme for JPEG2000 baseline system. IEEE Transactions on Image Processing **18** (2009) 1797–1803
12. Chou, Y.C., Chang, C.C.: Restoring objects for digital inpainting. In Pan, J.S., Huang, H.C., Jain, L.C., eds.: Information Hiding and Applications. Volume 227 of Studies in Computational Intelligence. Springer, Berlin (2009) 47–61
13. Weickert, J.: Theoretical foundations of anisotropic diffusion in image processing. Computing Supplement **11** (1996) 221–236
14. Belhachmi, Z., Bucur, D., Burgeth, B., Weickert, J.: How to choose interpolation data in images. SIAM Journal on Applied Mathematics **70** (2009) 333–352
15. Crow, F.C.: Summed area tables for texture mapping. Computer Graphics **18** (1984) 207–212
16. Mahoney, M.: Data compression programs. http://mattmahoney.net/dc/ (2011) Last visited July 22, 2011.
17. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V., eds.: Proc. First ACM Conference on Computer and Communications Security. (1993) 62–73
18. Zhang, X., Wang, S.: Efficient steganographic embedding by exploiting modification direction. IEEE Communications Letters **10** (2006) 781–783
19. Sharp, T.: An implementation of key-based digital signal steganography. In Moskowitz, I.S., ed.: Information Hiding. Volume 2137 of Lecture Notes in Computer Science., Berlin, Springer (2001) 13–26