GEOGRAPHIC DATA AND STEGANOGRAPHY
Using Google Earth and KML Files for High-Capacity Steganography

Malte Diehl
Department of Computer Science, Oldenburg University, Germany
malte.diehl@mail.uni-oldenburg.de

Keywords: Steganography, Security, Google Earth, KML, Geodata.

Abstract: Steganography is the art of hiding the existence of information, whereas cryptography only aims at hiding the content of a message. Most steganographic algorithms try to embed data into images, audio or video files that provide reasonable capacities. However, such systems are often vulnerable to simple statistical attacks.
In this paper, in order to provide an appropriate alternative to the currently used algorithms, we examine the information hiding properties of vector data that is used by many geographic information systems in great quantities. Unlike watermarking, we focus on maximising embedding capacities rather than on robustness, while still providing security against statistical attacks.
Our implementation that uses the KML format known from Google Earth and other map services can replace more than 20 % of the original data with hidden messages, provided that a lot of numerical geodata is present in the KML file. Thus, our algorithm can hide about twice as much as current algorithms for images. Yet, virtually no distortions are inflicted to the cover data.

1 STEGANOGRAPHY AND GEOGRAPHIC DATA

Steganography is the art of hiding a message within an unsuspicious cover such that the message's existence will not be detected by an adversary. Only the legitimate receiver shall be able to discover and extract the hidden message. In contrast, cryptographic algorithms merely try to hide the contents of a secret message. However, both steganography and cryptography use secret keys that solely the sender and the receiver possess[1]. Especially in the past two decades, many steganographic concepts and algorithms have been suggested for embedding data into images, video and audio files because these media are usually thought to provide enough redundancy to securely hide reasonable amounts of data.

Steganography is not to be confused with watermarking: Steganography's main purpose is secret communication, while watermarking is used for copyright protection by inserting a few key-dependent bits into an object so that a watermark detector can identify that object as property of some person or enterprise. While the main focus in steganography is secrecy, a watermark is not required to be totally invisible; it is more important that the watermark cannot be removed without rendering the watermarked object unusable.

In the recent years, geographic information systems (GIS) have been enjoying an increasing popularity. Besides professional applications that are, e. g., needed for spatial planning, internet or network-based services for route planners or topographic maps are attractive to many casual users.

Many methods have been proposed to insert robust watermarks into polygons, shapes and other spatial objects which GISes rely upon. However, by now, using such geodata for steganographic communications with notable embedding capacities has not received much attention, although spatial objects are likely to provide enough noise to securely embed

[1]There also exist some so-called pure stegosystems for which no keys are needed. Unfortunately, every adversary can then extract hidden messages as soon as the used algorithm is known, so that these systems are not really advantageous (cf. (Katzenbeisser and Petitcolas, 2000)).

messages of practically usable lengths.

Therefore, the objective of this paper is to examine how we can apply steganography to geodata and achieve both reasonable security against detection and high embedding capacities.

2 GOOGLE EARTH

The most prominent GIS example is perhaps Google Earth that millions of people have installed on their computers. It offers satellite images of the entire world, enriched with additional information, e. g., about highways or the locations of shops and restaurants. In many parts of the world, the resolution of the satellite data is so high that even objects of less than one metre in diameter are recognisable.

A major feature of Google Earth is that users can create their own geospatial data and share it with other people. User-defined objects can be as simple as placemarks that point at a certain location or complicated like, e. g., polygons with hundreds of vertices. Single objects can be combined to complex maps or animations, such as to illustrate a town's developement in annual steps. The underlying geodata to describe these objects is stored in the XML-based Keyhole Markup Language (KML) format. Many websites exist where users can upload their own KML files and download others' work to view it in their installation of Google Earth or related services.

Seen from a steganographic perspective, software like Google Earth can be turned into an inexhaustible repository of cover objects, and the above-mentioned websites represent places for steganogram exchange, both without raising any suspicion because of their popularity. In our experimentation and implementation of geodata steganography, we always used Google Earth due to its popularity and the many possibilities for users to create own data.

3 UNDETECTABLE EMBEDDING IN GEOGRAPHIC DATA

Many papers and articles are available that deal with the watermarking of geospatial data such as that used by Google Earth. In many approaches, the numerical values in the objects are transformed into another domain, e. g., in (Doncel et al., 2005) or (Bazin et al., 2007). Such transformations aim at finding some features or correlations within the data that resist common attacks such as rotating, scaling or cropping the object. However, this is likely to give away much of the available redundancy by using only a small part of the numerical data for the transformations. Thus it seems preferable to work directly with the numerical values that define the spatial object.

```
<Placemark>
  <name>Las Vegas, USA</name>
  <visibility>0</visibility>
  <LookAt>
    <longitude>-115.17288734724</longitude>
    <latitude>36.100841674561</latitude>
    <altitude>0</altitude>
    <range>320.7608545894</range>
    <tilt>0</tilt>
    <heading>131.1630019178</heading>
  </LookAt>
  <styleUrl>#default_myplaces_style
  </styleUrl>
  <Point>
    <coordinates>-115.1728874567,
    36.100841621771,0</coordinates>
  </Point>
</Placemark>
```

Figure 1: KML example: Representation of a placemark for Las Vegas including coordinates and camera view.

As with images, any alteration of geodata introduces some distortion. Following the idea of model-based steganography (MBS) that was introduced by Sallee (Sallee, 2004), we think of a numerical value as a vector of digits v that consists of a significant part v_s and an insignificant part v_i. The task that must be done before embedding is to identify v_s and v_i. Let the variables that are instantiated to v_s and v_i be V_s and, respectively, V_i. We assume V_i to follow a certain probability distribution $P_{V_i|V_s=v_s}$ that depends on the choice of v_s. Secure embedding algorithms will then replace any v_i with v_i' that holds some steganographic data. The choice of v_i' must also be in accordance to $P_{V_i|V_s=v_s}$ so that no statistical deviations are caused. Although this model – due to more complex statistical dependencies – seems too weak for the image domain in which it was first applied and broken soon afterwards (Böhme and Westfeld, 2004), it seems suitable for numerical data that contains a higher degree of randomness.

The data we want to use for embedding is completely contained in the KML files that store the geospatial objects in Google Earth. When we try to identify v_i in the digit vectors that are available in these KML files, this means that we have to find the part of the data that remains unsuspicious when altered. But what is unsuspicious? Whether a stego object looks suspicious in Google Earth, mainly depends on its layout in relation to the underlying satellite images. We can think of two attacks, assuming

Data type	Explanation
coordinates	Triples of latitude, longitude and altitude values to define a point.
LonLatBox	Top, bottom, left and right side of a bounding box and its rotation.
LonLatAltBox	Just as **LonLatBox**, but includes the altitude of the bounding box.
LookAt/Camera	Camera view on an object with 3D values for position and angles.
Orientation	Rotation of a 3D model along the *x*-, *y*- and *z*-axes.
Location	Location of a 3D model on the map.
ViewVolume	Defines how much of the current scene is visible.

Table 1: User-definable data elements with floating-point vectors as geographic references in KML 2.2.

that the adversary has, as usual, no access to the original cover:

- A human, who can relate the pixels in satellite images with KML data due to his contextual knowledge, observes objects that have been assigned a certain semantics by their creator in the metadata (e. g., a placemark for a certain building). If layout and semantics do not fit (e. g., the mark is placed far away from the building's location in the satellite image), the observer will raise alert.
- An algorithm could try to automatically map the semantics of a user-defined object to those contained in a database of reference objects. If matches are found and the layout of the observed object differs too much from the references, alert may be triggered, too.

Consequently, the challenge is to embed such that neither of the above adversaries ever triggers alert. Besides, we must also ensure that the stego objects are valid and do not violate the above-mentioned statistical constraints.

4 STEGANOGRAPHICALLY USABLE DATA IN KML FILES

The newest version of KML is 2.2 (Open Geospatial Consortium Inc., 2007). The objects that users can define are, among others, placemarks, lines, paths, overlays, 3D models, and polygons. Each of them can be equipped with a camera view, i. e., a set of parameters that determine how the user initially sees an object. A numerical value, or digit vector, consists of up to 16 digits to describe coordinates or angles and is contained in special KML elements within the object it describes. Note that, since modifiying digits in the integer part seems critical to us, we only regard floating point numbers for embedding, and of them only those where a large number of decimal places is actually frequently used. We give a list of floating point KML elements that we consider suitable for embedding in Tab. 1.

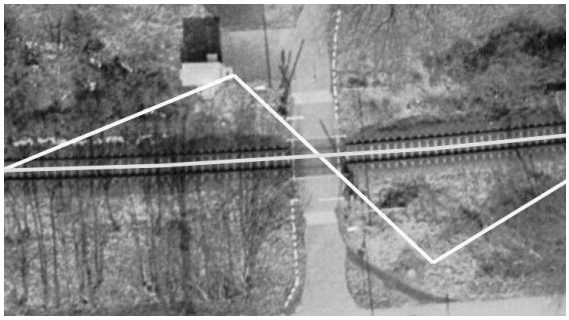

Figure 2: Example for strange appearance of objects with embedded message: The almost straight path marks a railway track in the original version, while the zigzagged stego version of the path fails to mark the track.

The size of the insignificant vector part in these elements varies depending on the strength of the correlation between digit vector and represented object, if such a correlation at all exists. E. g., coordinates of a polygon that confines a lake are all related to the lake's shore, and mostly even to a certain section of that shore. If, after embedding, these coordinates jitter arbitrarily within 100 metres around the shore, this is unacceptable because the original meaning of the shape is destroyed to a large extent. On the other hand, if a camera position from which that lake polygon is viewed jitters within the same range, this usually is completely unsuspicious as long as the polygon remains visible because the correlation between the lake and the camera position is less tight. Figure 2 provides an example of a path that was altered too much so that it is not correlated to the represented object anymore and looks quite strange.

Unfortunately, there is no formal way to precisely identify the insignificant part of a digit vector in a certain KML element. As a consequence, defining the number of digits per vector that can be replaced with stego data is somewhat ad hoc. However, the very same is also true for most steganographic algorithms

that work with the least significant bits of image pixels or DCT coefficients. We experimented with different degrees of alterations and found that the embedding capacities in Tab. 2 seem not to induce inappropriate changes and may thus be secure against detection. In that table, the number of digits replaced by steganographic data is shown for the attributes used in the KML elements of Tab. 1. The n exchanged digits are always the n rightmost in the vector before the last digit. The last decimal place is never changed (cf. Sect. 6.3).

Floating-point attrs.	**Capacity in digits**
north, *east*, *south*, *west*, *longitude*, *latitude*, *rotation*	#(Decimal places) -7
altitude, *heading*, *range*, *minAltitude*, *maxAltitude* *tilt*, *roll*, *topFov*, *leftFov* *bottomFov*, *rightFov*, *near*	#(Decimal places) -1

Table 2: Embedding capacities of digit vectors in KML files, where #(x) denotes the count of x.

For instance, if all but seven decimal places for the *north* attribute are exchanged, and the given digit vector is $v = (d_0, \ldots, d_{13})$ with d_2 to d_{13} as decimal places, we can replace the five digits d_8 to d_{12} with steganographic data. If a *north* vector contains the maximum of 16 digits, we can replace at most eight digits because there is always at least one integer digit and seven decimal places remain untouched.

For coordinate values which are measured in degrees of longitude or latitude, not using the first six decimal places means that the changes caused by embedding amount at most to 10^{-6} degrees. According to the definition of a degree of longitude and latitude (cf. (Cox, 1999)), this equals about 11 centimetres. In contrast, embedding starting at, say, the fifth decimal place would be clearly insecure with a maximum deviation of over 11 metres (Fig. 2 was created with that maximum deviation).

With all other values that either describe angles or distances in metres, such as *tilt* or *altitude*, we use all but the last decimal place. This choice is justified as follows: Angles are used to position the camera view that is coupled to an object. However, this relationship is quite loose. For example, it does not make any difference to a user whether the tilt towards an object is 85.562 or 85.982 degrees, which, in terms of longitude and latitude, would be critical for the positioning of the object itself. The same is true for the altitude in which an object is positioned. If the user decides not to have an object clamped to the ground automatically and chooses an unusual altitude like 2501.45 metres, changes to the decimal places will be insuspicious as well. However, changing digits in the integer part might easily result in violation of the intentions the user had when setting up the original view, so we recoiled from that.

5 IMPLEMENTATION AND PERFORMANCE

We implemented a stegosystem that subsequently replaces numerical geodata in one or more KML files with steganographic data as described in the previous section. The user types in an arbitrary message or selects a hidden-message file[2]. The message is Huffman-encoded with a given probability distribution that is optimised for English texts. In a second step, the message is encrypted with AES into a ciphertext of $n \cdot 16$ bytes before it is turned into an integer value.

To embed the integer that represents the message, the selected cover files are parsed and searched for vectors suitable for embedding in a predefined order. Now the algorithm splits up the message into short strings that fit into the insignificant parts of the digit vectors and overwrites the original data. As the receiver must distinguish the embedded data from other random numbers, the first five replaced digits contain the encrypted hidden-message length. For hidden communication, sender and receiver need to share a secret key and the stego file that is output by our algorithm.

The most important performance criterion for a stegosystem is how much data can be embedded without compromising security. We empirically tested the embedding capacity for KML files with different numbers of paths, where each vertex in a path is represented by a coordinate triple of latitude, longitude and altitude values, and randomly selected excerpts from *Wikipedia* articles in English. Because the altitude values are 0 by default, they did not provide any capacity in our test. Table 3 shows the results.

The embedding capacity, i. e., the ratio between file and message size, varies between 0.20 and 0.27, depending on the degree of compression that the Huffman encoding can achieve for the given text. The AES encryption is also relevant for capacity to some extent because the ciphertexts are $n \cdot 16$ bytes in length, thus causing jumps in capacity requirements whenever n increases. Some constant amount of tex-

[2]Currently, as this is a proof-of-concept implementation, only 8-bit character sets are supported.

Coord. triples	File size	Message size	Ratio
100	4,892 bytes	990 bytes	0.202
200	8,723 bytes	2,089 bytes	0.239
300	12,531 bytes	3,007 bytes	0.240
400	16,278 bytes	4,085 bytes	0.251
500	20,137 bytes	5,084 bytes	0.252
600	23,969 bytes	6,297 bytes	0.263
700	27,886 bytes	7,034 bytes	0.252
800	31,731 bytes	7,879 bytes	0.279
900	35,607 bytes	8,534 bytes	0.248
1000	39,549 bytes	9,723 bytes	0.246

Table 3: Experimental capacities for KML files with path objects consisting of between 100 and 1000 coordinate triples of latitude, longitude and altitude values. The message size refers to the size of the original plain text before the embedding process starts.

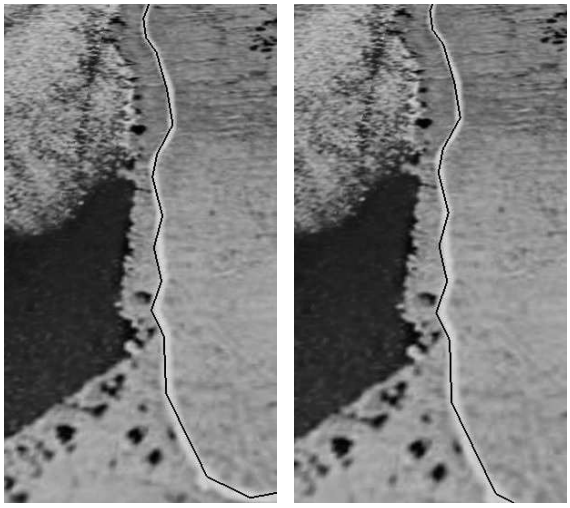

Figure 3: Embedding into a path that marks a trail right of the shore of a lake (cover left, steganogram right).

tual metadata that precedes the numerical data also lowers the embedding capacity especially for small numbers of vectors. Anyhow, with the values in Tab. 3, our system clearly outperforms the embedding capacities of current steganographic algorithms to embed into JPEG images, such as *OutGuess* (Provos, 2001), *F5* (Westfeld, 2001), *MBS* (Sallee, 2004), or *Perturbed Quantization* (Fridrich et al., 2004). Furthermore, Fridrich et. al. (Fridrich et al., 2007) estimate the amount of data that can be securely embedded into JPEG images to be less than 5 % of the cover.

To give an example of the capability of our algorithm, Fig. 3 contains the original version of a KML path (black line) that marks a trail on the right side of a lake and the steganogram with 576 bytes embedded. If one looks carefully, one sees that the camera position has slightly changed and that some pixels of the path differ. However, the camera position is still reasonable and the KML path never leaves the trail or jitters around it. So, without knowledge of the original cover, an attacker would hardly be able to detect the steganogram as the images are nearly identical.

6 SECURITY ISSUES

Having identified the insignificant parts v_i of the digit vectors in KML files, we have to estimate the security of our embedding method. This means that we must find out, according to our statements in Sect. 3, if our steganographic replacements v_i' obey the same probability distribution as the original instances from V_i, given the significant data v_s. Since the sequences of digits that our algorithm outputs have a high degree of randomness, we examined whether the data found in KML files that were created by Google Earth was equally random. We encountered some flaws and biases in the data, produced by our system and by Google Earth, which could lead to reliable detection of steganograms. We will tackle them in the following subsections. The effect of any measure taken to counter such security problems is already included in the performance figures presented in Tab. 3.

6.1 Digit distribution

First, we examined the distribution of the digits in the insignificant parts of the vectors. We assumed both the original and our stego data to be uniformly distributed, i. e., all digits appear with about the same probabilty. To verify this assumption, we generated cover files with a total of more than 100,000 digits that were suitable for embedding. We further used a second, equivalent set of KML files to embed messages that consumed between 50 and 100 % of the available capacity. For both sets, we computed the digits' probability distribution. The results can be seen in Fig. 4.

Obviously, there are only small deviations (below 1.5 %) from the expected frequency for each digit, so we have no reason to believe that either cover or steganogram data is not uniformly distributed. We repeated this test separately for the first, second and third digits in the insignificant vector parts with the same result. Therefore, the overall distribution of the digits in a file does not seem to provide an anchor for attacks.

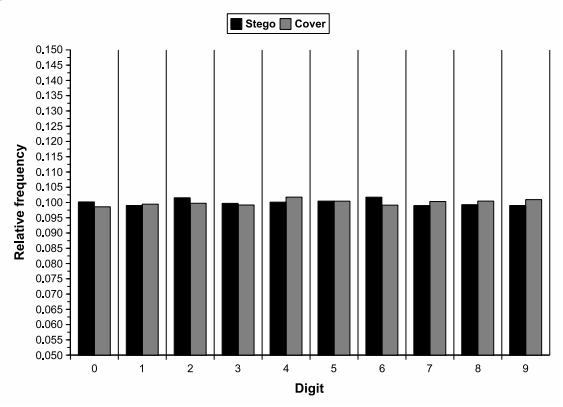

Figure 4: Probability distribution of the digits in cover and stego file.

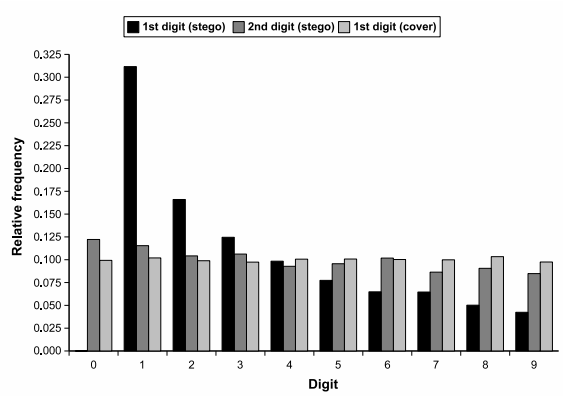

Figure 5: Probability distribution of the first and second digits in stego data and the first digits in cover data.

6.2 Benford's Law

Benford's law (Benford, 1938) demands that the first digit of a multi-digit number be not uniformly, but logarithmically distributed. For the digits from 1 to 9, it implies that the first digit is 1 with about 30 % probability, and 2 with only about 18 %. Finally, 9 occurs in the first place in somewhat less than 5 % of the samples. Of course, 0's are never first digits.

Figure 5 shows the empirically measured probability distributions for both the first and the second digit of the stego data produced by our program as well as for the first digit in the KML cover files before embedding. We found that the first digits we output also obey Benford's law, while the first digits of the vectors in which the first digits of our messages are embedded are uniformly distributed. The second digits in our output are not uniformly distributed either, though the bias is smaller. To fix this potential insecurity, we apply a key-dependent random permutation to the entire number that is produced by our system. Thus, the bias in the distribution of the leading digits is spread over the whole hidden message and vanishes.

6.3 Biased values in Google Earth

While we were testing our algorithm, we experienced a strange bias in Google Earth: If one types in an exact coordinate value, this is often not stored as typed but in- or decremented by some marginal quantum. For example, when typing 50° as latitude in a dialog box, this value is likely to become 50.00000000001°. Since we assume that users will tend to hand over exact coordinates to Google Earth, such values very close to the next integer will occur with increased probability due to the bias. In order not to distort their distribution, our algorithm does not change them.

In case that two or more values are equal in a KML file, there is a high probability that these values relate to each other. For instance, these values could belong to layered overlays. Therefore, we only replace one in a couple of such values and set the others to the same value such that the relationship is maintained.

Finally, we restrained from embedding in the least significant digit of any value. The reason is that with classical LSB embedding we risk to change the probability distribution for the digit vectors' lengths throughout a file because trailing 0's are cut by Google Earth. This would also cause additional problems for the extraction of the embedded data.

6.4 Validity of the generated values

Data generated by our system would be easily discoverable if it was invalid, i. e., if it contained digit vectors v that Google Earth cannot generate. Since we had no access to Google Earth's source code, we had no chance to directly look for impossible values. Thus, we had to test empirically whether all values are possible. In Google Earth, with the camera zoomed in to the ground as close as possible, we created a large set of placemarks with digit vectors $v = (d_1, \ldots, d_n)$, where d_1 is the most significant digit and d_n the least significant digit in v. We then tried to create a vector $v' = (d_1, \ldots, d_n \pm 1)$. This was done by zooming out in equidistant steps while keeping the position and observing any changes to the vectors.

We found that with increasing camera height the least significant vector digits began jittering around the original values. In many cases, a vector v' as above was found after some zooming so that we have evidence that all theoretically possible values can actually be generated by Google Earth. However, we

could not check whether such v' can only be found under certain side conditions. Anyhow, an attacker will face the same problems when he tries to distinguish original data and such generated by our program.

7 CONCLUSIONS

The steganographic algorithm we developed for embedding in KML files can achieve significantly higher capacities than those that are possible for image-based steganography. It is just as easy to use as other current methods, and providing suitable cover files is not more difficult than taking a picture. Although we cannot proof it formally, we are confident that the very small level of distortion that our algorithm causes to the original data and the additional security mechanisms of Sect. 6 are enough to prevent the attacks described in Sect. 3.

As far as the suspicion that the mere fact of using KML files for steganography might raise is concerned, we believe that it will not be larger than with exchanging images: Besides Google's promotion of the KML format, many websites have already specialised on distributing KML files. Therefore, no-one will become a suspect just by frequently transmitting KML files through a network.

However, one might argue that it is already possible to do steganography with Google Earth or other GISes by adapting existing watermarking schemes for geospatial objects such as polygons. But these schemes have in common that their embedding capacities are rather insignificant.

An overview of embedding capacities for some recent watermarking schemes based on 3d triangular meshes can be found in (Wang et al., 2007). Although the polygons we use are no meshes and only 2d, the numbers stated there allow for some comparison. The five schemes discussed in that paper offer capacities between less than one bit and approximately one byte per edge in a mesh, and the high-capacity scheme is already somewhat fragile.

In contrast (cf. Tab. 3), we can embed about ten bytes per coordinate triple which corresponds to one vertex in a polygon or mesh. Consequently, we can conclude that our scheme is by far more suitable for steganographic communications than those that were initially designed for watermarking.

ACKNOWLEDGEMENTS

This work has been supported by the German Research Foundation (DFG), grant GRK 1076/1. Furthermore, thanks to the anonymous SECRYPT reviewers and to our colleagues at Oldenburg University that provided us with useful comments.

REFERENCES

Bazin, C., Le Bars, J.-M., and Madelaine, J. (2007). A Blind, Fast and Robust Method for Geographical Data Watermarking. In *Proceedings on the 2nd ACM symposium on Information, Computer and Communications Security*, pages 265–272, Singapore.

Benford, F. (1938). The Law of Anomalous Numbers. In *Proceedings of the American Philosophical Society*, number 78, pages 551–572.

Böhme, R. and Westfeld, A. (2004). Breaking Cauchy Model-Based JPEG Steganography with First Order Statistics. In Samarati, P., Ryan, P., Gollmann, D., and Molva, R., editors, *Computer Security ESORICS 2004. 9th European Symposium on Research in Computer Security*, volume 3193 of *LNCS*, pages 125–140.

Cox, A., editor (1999). *Allen's Astrophysical Quantities*. Springer-Verlag, 4th edition.

Doncel, V. R., Nikolaidis, N., and Pitas, I. (2005). Watermarking Polygonal Lines Using an Optimal Detector on the Fourier Descriptors Domain. In *Proceedings of 2005 EURASIP European Signal Processing Conference*, Antalya, Turkey.

Fridrich, J., Goljan, M., and Soukal, D. (2004). Perturbed Quantization Steganography with Wet Paper Codes. In *Proc. ACM Multimedia Security Workshop*, pages 4–15, Magdeburg, Germany.

Fridrich, J., Pevny, T., and Kodovsky, J. (2007). Statistically Undetectable JPEG Steganography: Dead Ends, Challenges, and Opportunities. In *Proceedings of the ACM MM&S Workshop*, pages 3–14, Dallas, USA.

Katzenbeisser, S. and Petitcolas, F., editors (2000). *Information Hiding. Techniques for Steganography and Digital Watermarking*. Artech House.

Open Geospatial Consortium Inc. (2007). KML 2.2 An OGC Best Practice. Technical report.

Provos, N. (2001). Defending Against Statistical Steganalysis. In *Proceedings 10th USENIX Security Symposium*, Washington, USA.

Sallee, P. (2004). Model-based Steganography. In *Proceedings of International Workshop on Digital Watermarking*, volume 2939, pages 154–167.

Wang, K., Lavou, G., Denis, F., and Baskurt, A. (2007). Hierarchical Blind Watermarking of 3D Triangular Meshes. In *Proceedings of the IEEE International Conference on Multimedia & Expo*, pages 1235–1238, Beijing, China.

Westfeld, A. (2001). F5 – A Steganographic Algorithm: High Capacity Despite Better Steganalysis. In Moskowitz, I. S., editor, *Information Hiding. 4th International Workshop, IH'01*, pages 289–302.