StirMark using Octave

Anandha Gopalan and Francis Hoang
Department of Computer Science
Wichita State University

September 29, 2000

Abstract

Digital watermarking has become very important for copyright protection and authentication. Broad claims have been made about the 'robustness' of various digital watermarking method. Unfortunately the criteria as well as the pictures used to demonstrate these claims vary from one system to the other and recent attacks show that the robustness criteria used until now are inadequate. This paper shows that majority of attacks implemented in the Stirmark package can be done in a quick and easy way using Octave. Given a watermarked image, StirMark will apply several transformations with various parameters then produce an output image that can be tested using benchmark to judge the robustness of image watermarking systems.

1 introduction

At the beginning of 1990, there have been a rapid grow of interest in digital watermarking, embedding imperceptible information ino data. The fears that copyright would be eroded by the case with which digital media could be copied led people to study ways of embedding hidden copyright marks and serial numbers in the digital data such as pictures, sound, program code, etc... for the purpose of ownership verification or authentication. Digital watermarking for images is done by embedding the information in images in a robust manner so as to be able to retrieve it later for authentication purposes.

Unfortunately the criteria as well as the pictures used to demonstrate these claims vary from one system to the other and recent attacks show that the robustness criteria used until now are inadequate: additive Gaussian noise, low pass filtering rescaling and cropping have been addressed in most literature but specific distortions such as rotation have been rarely addressed. In some cases the watermark is simply said to be "robust against common signal processing algorithms and geometric distortions when used on some standard images.

Most of the potential attacks detailed in:

Martin Kutter and Fabien A. P. Petitcolas. A Fair Benchmark for Image Watermarking Systems. To be presented at the 11th Annual Symposium on Electronic Imaging, IS&T/SPIE, San Jose, USA, 23-29 January 1999.

http://www.cl.cam.ac.uk/~fapp2/papers/ei99-benchmark/

are actually implemented into this version of StirMark: given a watermarked image, StirMark will apply these transformations with various parameters. Then the output images can be tested with watermark detection or extraction programs.

StirMark is a generic tool for simple robustness testing of image watermarking algorithms and other steganographic techniques. It can be applied to photographic digital images and it will distort the watermark of too simplistic marking techniques such that the embedded watermark or steganographic message cannot any more be detected and decoded from the result image.

2 Implementation of stirmark in Octave

Due to the powerful ability of Octave to perform numerical computation and manipulation of matrices. The Stirmark package was written entirely in Octave. Octave is a high-level interactive language for numerical computations. It was originally intended to teach chemical reactor design [1]. A key functionality of Octave is that it provides a convenient command line interface for solving linear problems numerically, and for performing other numerical experiments. It can also be used as a batch-oriented language and for very efficient image processing. Because of these additional functionalities, it is now being used for other purposes such as solving differential equations, teaching linear algebra, image processing applications, etc. Octave is a free software, but it is not in the public domain. Restrictions on its distribution can be found in the GNU General Public License [2] that comes with Octave. Octave is also available on the Internet from [3], and additional information is available from [4].

Octave has been extended by various contributors, and this has vastly enhanced the abilities of Octave. Contributions by Darrel Hankerson and Greg A. Harris with functions such as getpgm (), dct (), etc. comprise the JPEGtool package of Octave [5]. The function getpgm () can read in a PGM image file and store it in a matrix while the function dct () computes the discrete cosine transform of the matrix. This fast computation is very desirable in most image processing applications. For intance, digital media like images, audio, and video are readily manipulated, reproduced and distributed over information networks.

Octave provides a powerful tool for manipulating matrices. There are a number of functions available for checking to see if the elements of a matrix meet some condition, and for rearranging the elements of a matrix. For example, Octave can rotate the element, extract the upper or lower triangular parts, or sort the columns of a matrix. Since PPM and PGM images are simply matrix of pixels and Octave provides a quick and easy way to manipulate the matrix, Octave proves to be a very easy and efficient language for graphical manipulation.

3 Image formats

StirMark supports images in the PGM, PPM. Any other formats will automatically be converted to PPM.

PGM and PPM files have the following header:

- A “magic number” for identifying the file type.

  “P2” and “P5” for PGM format in ASCII and raw respectively. “P3” and “P6” for PPM format in ASCII and raw respectively.
- Whitespace (blank, TABS, CRs, LFs).
- The width, formatted as ASCII characters in decimal.
- Whitespace.
- The height, again in ASCII decimal.
- Whitespace.
- For PGM file, the max gray value formatted as ASCII characters in decimal and for PPM file, a max color component.

There might be comments in between which begin with #.

PGM is a portable grayscale image while PPM is a portable pixmap image. After the header is a list of pixels of the image.

Here is an example of a PGM file:

```
P2
# a pgm file
10 5
255
```

```
25 34 23 43 65 87 90 1 0 30
23 34 47 43 89 87 90 1 0 30
93 34 52 43 65 87 90 1 9 30
48 34 23 43 65 87 90 1 0 30
25 34 23 43 65 87 90 1 0 30
```

A PPM file looks very similar to a PGM file. The only different is that PPM has three values for each pixel corresponding to the three components: red, green, and blue. This make PPM file an ideal color image format for matrix manipulation in Octave. We can use the reshape function in Octave to reshape the file into three rows where the first row contains all the pixels for the red component, the second row contains all the pixels for the green component, and the third for the blue component. We obtain three matrices by cutting these rows and reshape them to its original size. From then on we can work with these matrices just like we would with a gray scale image. Whatever we did to one component, we carry out the same operation for the others. This make it so easy to manipulate any color image.

Stirmark will look for only four magic number: P2 and P5 (for grayscale images), P3 and P6 (for color images). If these magic numbers are not found, it will call the system function "convert" to convert the image to PPM.

To save the matrix back into an image file, we just need to create a header according to the number of rows and columns of the matrix, and then reshape the matrix into a one column matrix and then write the pixels to the file. To speed up the process we will always write them back as raw text instead of ASCII. If we want to save an image in PPM format, we must reshape and append the three red, green and blue matrices together before writing them to a file.

To display an image, we just write the pixels of the matrix to a temporary file then call the system command "XV" to display the file. After XV displayed the image, we can delete the file.

Octave does not require the user to specify the type of the function parameters, so we take advantage of this option when working with PGM or PPM file format. If we work with PGM image then we just pass the matrix to certain function; if we work with PPM image, we pass a structure contain three matrices to that function. At the beginning of each fuction we will test whether the input parameter is a matrix or a structure of three matrices. This make it easy to write the same function that would process either PGM or PPM image.

4 The functions in stirmark

This Stirmark package is written in Octave with the following functions:

- stirmark(matrix, `<command_line_option>`);

  distort the image by the value specify by the user. The command line options are provided in table 1.

- benchmark(original_matrix, watermark_matrix);

  check the differences between the two images.

- readimg(`<image>`);

  read any pgm or ppm image format (if the input file is not a pgm or ppm file Octave will convert that image file to PPM format) and return a matrix of pixels for that input file.

- writeimg(matrix, `<output name>`);

  write the pixels into a PGM or PPM file.

- imagescan(matrix);

  display the image by writing the pixels of the input matrix into a file and call xv.

- traCrop(matrix, x_coordinate, y_coordinate, size_of_x, size_of_y);

  crop the image beginning at the specify x and y coordinates to the specify size.

- traRemove(matrix, x_coordinate, y_coordinate, #_of_rows, #_of_columns);

  remove certain numbers of rows or columns at specify point.

- traRotate(matrix, angle);

  rotate the image by certain angle.

- traScale(matrix, percent);

  scale the image by a certain percentage (i.e. 50 would reduce the image by half, 200 would doulble the size...)

- Restore(image, rows, columns);

  restore the image to specific rows and column.

- traShearx(matrix, percent);

  shear the image in x direction by a certain percentage.

- traSheary(matrix, percent);

  shear the image in y direction by a certain percentage.

- traFilter(matrix, `<type>`);

  filter the image with Median, Gaussian, Sharpen, and Laplacian method.

-h	help menu
-c	crop image
-med	median filter
-shp	sharpen filter
-gau	gaussian filter
-lap	laplacian filter
-rem	remove rows or columns
-rot <angle>	rotate image
-sca <percent>	scale image
-shx <percent>	shear in x direction
-shy <percent>	shear in y direction

Table 1: Command line options

5 Benchmarking

Even though there are several methods on embedding digital information into images, digital watermarking remains a largely untested field and very few authors have published extensive tests on their system [6]. Benchmark provide an efficient tool to find out which techniques of the research work better than others.

The following equations are used for benchmarking:

Average Difference:

$$AD = \frac{1}{XY}\sum_{x,y}|P_{x,y} - P'_{x,y}|$$

Mean Square Error:

$$MSE = \frac{1}{XY}\sum_{x,y}(P_{x,y} - P'_{x,y})^2$$

Signal-to-noise Ratio:

$$PSNR = XYmax_{x,y}P^2_{x,y}/\frac{1}{XY}\sum_{x,y}(P_{x,y} - P'_{x,y})^2$$

Normalized Cross-correlation:

$$NC = \sum_{x,y}P_{x,y}P'_{x,y}/\sum_{x,y}P^2_{x,y}$$

Correlation Quality:

$$CQ = \sum_{x,y}P_{x,y}P'_{x,y}/\sum_{x,y}P_{x,y}$$

6 Attacks used in Stirmark

Stirmark can perform a series tests of which serve as the basis for the benchmark. For an already watermark image, Stirmark will apply several transformation with various parameters. The result image can then be tested with watermark detection or extraction programs. So the full process can be automated.

Following is a list of attacks that actually implemented into Stirmark

- low-pass filtering: multiply each pixel and its neighbors with certain coefficients
  - Gaussian
    $$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$
  - Simple Sharpening
    $$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$
  - Laplacian
    $$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 1 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$
  - Median Filter

    Each pixel is replaced by the median of the 3 x 3 neighborhood centered at that pixel.

  Low-pass filter can be performed in a quick and easy way in Octave because Octave allows the user to shift the row and column of the matrix. For example, for the Gaussian filter method, to get the center pixels of the entire matrix, we multiplize the entire matrix by 4. To get the left neighbors, shift the matrix to the right by one column and multiplize that matrix by 2. To get the upper left neighbors, shift the original matrix to the right by one column and shift down 1 row. When we add all nine matrices together, this will yield the filter image without going through each pixel because the for loop in Octave take tremendous amount of time. The same can be done for median filter, but after we get all nine matrices, we reshape all of them to 1 column and append all nine of them together. After calling the median fuction, we will get the median filter matrix. We just have to reshape it to the original size.

- scaling:
  the scaling factor enter by the user is the p/q coefficient where:

  The new number of columns equal to the factor divide by 100 and multiplize by the original column. Same way is use to find the new number of rows.

By dividing the original number of rows or columns by the number of new rows or columns, we have the coefficient to determine with specific row or column should fit into the new row or column.

In another word, this coefficient determines which pixels should be repeat for enlarging or which pixels should be throw away for reducing the size of the image. Octave allows the user to work with an index, so this process can be done quickly by scanning through all the columns first and then the rows.

- Cropping:
  Cropping can disturb the watermark of an image since some infringers are just interested in certain part of the copyrighted image. The extreme case of cropping is the "Mosaic" attack which web sites take advantage of image segmentation[8]. Octave provides a very quick way to crop an image. For a matrix of pixels, Octave can copy all the rows and columns starting from a specific coordinate.

- Removing rows or columns:
  Removing rows or columns of pixels from an image can make the watermark undetectable. This process can be done in a similar way as cropping. If the user wants to remove n rows or columns starting from the edge, Octave will copy the pixels starting from coordinate n (or copy from the beginning to n row or column). The user can also remove the middle rows or columns of the matrix. Octave allows us to cut the matrix into smaller matrices and then append them together to remove the specific rows or columns.

- Shearing in x and y direction:
  Shearing the image by a small amount is an effective way to distort the watermark. Shearing in certain direction can be done by multiplize each pixel with the following matrix:

$$SH_x = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  Where a is the angle from the y axis.

$$SH_y = \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  Where a is the angle from the x axis.

  Since Octave allows us to quickly copy rows or columns of the matrix, we only need to find out the new posion for all the pixels in the first column if we are shearing in the X direction Or the new position for all the pixels in the first row if we are shearing in the Y direction. Then start copying the rest of the pixels to their new positions starting from that coordinate.

  In another word, we just need to find the new position of all the pixels in the first row or column depend on which direction the user wants to shear and then use them as the coeficients to push that row or column by that factor.

- Rotating:
  Watermark are very sensitive to rotation. A combination of small angle rotation and cropping do not usually change the commercial value of the image but can make the watermark undectable. The new coordinate of each pixel can be determined by the following matrix:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  Even though rotation is important in geometric distortion, this method points out the weakness of Octave. Octave can not refer to the coordinate of a pixel by two indices. We cannot refer to matrix A(x,y) where x and y are indices with a list of coordinates. We cannot avoid going through at least one for loop and the for loop in Octave causes the running time to be really slow.

  Rotation also produces another problem. There might be holes in the picture because some of the new coordinates might overlap one another when we take the nearest neighbors. We need to go back and replace all the hole with their neighbors. Since we initially set all pixels value to 0, we need to go back and replace all the black pixels with its neighbors.

For more information, please see:
http://www.cl.cam.ac.uk/~fapp2/watermarking/

Still more information is provided in:

Fabien A. P. Petitcolas, Ross J. Anderson and Markus G. Kuhn. Attacks on copyright marking systems. In David Aucsmith, Ed., second workshop on information hiding, in vol. 1525 of Lecture Notes in Computer Science Portland, Oregon, USA, 14–17 April, 1998, pp. 218–238. ISBN 3-540-65386-4.

http://www.cl.cam.ac.uk/~fapp2/papers/ih98-attacks/

7 Conclusion

StirMark is a quick tool for testing the robustness of image watermarking algorithms and other steganographic techniques. It can be applied to photographic digital images and it will distort the watermark with several linear and nonlinear transformations such that the embedded watermark cannot any more be detected or decoded from the result image.

This paper shows that majority of attacks implemented in the Stirmark package can be done in a quick and easy way using Octave. This paper describes several numbers of attacks that are frequently applied by the user. Given a watermarked image, StirMark will apply several transformations with various parameters then produce an output image that can be tested using benchmark to judge the robustness of image watermarking systems. Due to the powerful ability of Octave to perform numerical computation and manipulation of matrices, Octave proves to be quite an efficient language to manipulate digital images. Working with PGM and PPM image under the simple environment of Octave also help the user gain more understanding toward geometrical transformations of computer graphic.

References

[1] http://www.che.wisc.edu/octave/history.html: History of Octave

[2] http://www.gnu.or/copyleft/gpl.html: GNU General Public License

[3] ftp://ftp.che.wisc.edu/pub/octave: Downloading Octave

[4] http://www.che.wisc.edu/octave: Octave Documentation

[5] http://www.dms.auburn.edu/compression: Information Theory and Data Compression

[6] Fabien A. P. Petitcolas and Ross J. Anderson, Evaluation of copyright marking systems. Proceedings of IEEE Multimedia Systems. Vol. 1, pp.574-579. June 1999.

[7] M. Kutter and F. A. P. Petitcolas. A fair benchmark for image watermarking systems. Electronic Imaging '99. Security and Watermarking of Multimedia Contents. Vol. 3657. pp. 226-239. January 1999.

[8] Fabien A. P. Petitcolas, Ross J. Anderson, Markus G. Kuhn. Attacks on copyright marking systems. Second Workshop on Information Hiding. Vol. 1525. PP. 218-238. April 1998.