Digital Watermarking for Video
A Forensic Solution to Movie Piracy

Jacob S. DeBoer
School of Film and Animation and Center for Imaging Science, Rochester Institute of Technology
May 2014

***Abstract*—A method is proposed for embedding data on a digital motion picture to be shown in a theater, such as date and time of exhibition, and a theater identifier. Embedded data appears on the video in the form of noise. The goal of this project is to keep the noise level less than what it takes to ruin the viewing experience to the human visual system, while the embedded data can still be extracted from a secondhand recording of a movie in a theater and can lead to determining the locations and times that are most prone to theater piracy.**

This paper also includes the results of a psychophysical experiment and a practical simulation of piracy performed at Rochester Institute of Technology.

***Index Terms*—Motion Pictures, Watermarking**

I. Introduction

As the Internet continues to expand, it becomes easier and much more commonplace for pirated videos to be shared. This is, in my opinion, one of the greatest challenges the film industry faces today, as the U.S. Congressional International Anti-Piracy Caucus estimates that the industry loses about $25 billion every year to piracy. It is also estimated that 90% of pirated videos originate from someone secretly recording while sitting in a theater.

Digital watermarking is a process by which data is embedded into a video in such a way that it is largely invisible to the human eye. This is not file metadata, but rather another signal well hidden in the image. This is important to remember, since it is a secondhand recording of the video on screen that we will ultimately be dealing with. Detecting information such as the copyright owner, the name of the theater, and the date and time that the video was played can all be important information in learning the circumstances under which a video was stolen.

While it may be difficult to use watermarking to find the criminal himself, learning the circumstances under which the video was stolen can be valuable so that we can find the most targeted theaters and the most common times that piracy happens and enforce better standards. While there is no way to prevent people from taking out their phones in a theater and recording video, we can certainly use this information to make it harder to do so. In some cases, especially with a video shared online, rather than sold on the streets, there are further measures that can help us catch the thief after a video has been identified as a stolen one.

The embedding and extracting algorithms proposed in this paper are loosely based on a paper by Chris Honsinger and Majid Rabbani of the Eastman Kodak Company in Rochester, NY. Honsinger and Rabbani proposed a method of data embedding on an image based on a convolution of a message image with a random phase carrier. This will be further expanded to apply to video.

Additionally, it is of the utmost importance that the ability to extract data from a video remains effective regardless of position, rotation, scale, perspective, color balance, and any other manipulation that a secondhand recording of a video in a theater can apply. Ways to accomplish these invariances will be discussed.

Last, but certainly not least, the watermarking must not make any artifacts such that the embedded content is unpleasant for a viewer to watch. Included in this research are the results of a psychophysical experiment in which I have found the just noticeable difference (JND) – the minimum amount of watermarking that a human being can detect while watching the content.

II. Embedding Data

A. What to Embed

If there is one thing we can say for certain, it is this: the more data we embed in a video, the more noticeable the watermark is, and the harder it is to extract the correct results. Arguably the most important step is the first: determining the best way to encode the least amount of data necessary, since the more data we try to fit in to an image, the harder it is to hide this data, and the harder it will be to extract.

Manuscript received May 21, 2014. This work was completed in partial requirement of the B.S. Motion Picture Science degree at Rochester Institute of Technology's School of Film and Animation.

Jacob S. DeBoer is with Rochester Institute of Technology, 1 Lomb Memorial Dr., Rochester, NY 14623 USA (RIT Phone: 585-475-2411 e-mail: jsd4568@rit.edu).

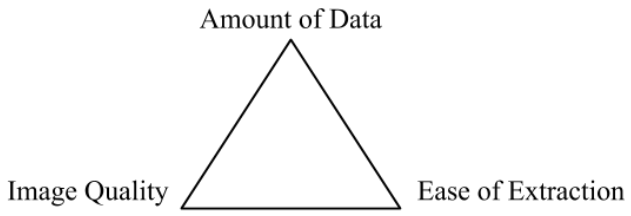

Fig. 1. In general, all watermarking algorithms can be thought of as a trilemma between embedding a greater amount of data, ease and chance of successful extraction, and preservation of the original image quality. The more heavily we invest in one, the more difficult the other two become.

Depending on the exact application of this process, the exact information one may wish to embed can vary. For the purposes of finding information about the circumstances surrounding movie piracy, I find it appropriate to embed:

1) The date and time that the movie was played, given in a certain amount of time after a fixed date. Given that the average length of a Hollywood movie today is 110 minutes, the precision of an hour seems adequate enough to determine the correct showing of the movie in which the piracy occurred while not taking up a large amount of encoding space.
2) The 5-digit US ZIP code of the area in which the movie was played.
3) A 3-digit ID number for the theater, to identify theaters within a ZIP code area.

The 3 pieces of data are to be arranged in a string as follows: the time in hours, followed by the ZIP code, followed by the 3 digit ID number. The reason for placing the numbers in this order is that the final integer made from these numbers will be converted to binary and must be as small as possible. Suppose that the ZIP code or ID were first. A leading digit such as 9 would make the encoded value unpredictably high. Instead, with the number of hours first in the string, it is certain that we will not need 49 bits (a number which will soon come in to play) to encode the final number until September 16, 2634.

As an example:

1) If the movie is shown at 12:00PM on December 25, 2013 and our fixed reference date is July 1, 1992 (the author's birthday), then the encoded number is 188341, because that is the number of hours that have passed between the two dates. Of course, the classic Unix epoch of January 1, 1970 can certainly be used, but a later reference date is preferable to take up less bits.
2) If the movie is shown in Rochester, NY (Henrietta), the ZIP code would be 14623.
3) Suppose the movie was shown in a theater in Rochester with the ID number 394 associated with it.

The final string would be

18834114623394

which, as an unsigned integer, requires 45 bits to represent.

B. *The Message Image*

In order for data to be watermarked via the Honsinger and Rabbani convolution method, it must exist in the form of an image. The image to be embedded and later extracted is called the "message image". The image on which the watermark is placed (in this case, a frame from a motion picture) shall be called the "base image".

The message image proposed for this application begins as a 17x17 image where each pixel is 0.5 gray. The pixel at row 2, column 16 is made white, to signify the correct rotation, which will prove useful during extraction.

The number created in the previous step is now converted to binary (49 bits, with leading zeroes if necessary) and placed on the message image as shown:

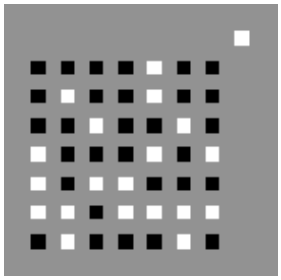

Fig. 2. A sample message image representing the number 18834114623394. Images such as these are what we aim to embed and extract. Multiple convolved images like these will be tiled and placed on the base image. A design such as this will ensure that the image is rotated properly and that parts of two adjacent messages are not mistaken for a single message.

From left (column 2) to right (column 14), top (row 4) to bottom (row 16), a bit is represented as either black for a zero or white for a one.

I will say again that embedding the data from the string as I have suggested, in which the hours after the reference date are placed first, will take up at most 49 bits until the year 2634.

C. *Embedding Data Into a Single Frame*

The process of embedding a watermark proposed by Honsinger and Rabanni is actually quite straightforward. In the following equation that they have proposed, let $I(x,y)$ be the base image, $I'(x,y)$ be the resultant watermarked image, $M(x,y)$ be the message image, and $C(x,y)$ be an image of random noise the same size as the message image, known as the carrier image.

$$I'(x,y) = \alpha(M(x,y) * C(x,y)) + I(x,y) \quad (1)$$

where * represents cyclic convolution (or multiplication of the two images in the Fourier domain) and α is a scaling constant chosen for robustness.

Some who are familiar with this method have suggested that the carrier image could be a flat field, rather than noise. Through experimentation, I have found that after combining the signals of I and $M*C$, having more and varied frequencies in a carrier image allows more detail to be extracted, rather than a flat field, which upon extraction produces more of a low frequency outline of the message image. Flat fields are easier to produce, and doesn't require the need for a specific key, and that is why some people like to use them, but I find that this is much more practical when embedding photos or logos as a watermark. Noise should for embedding a message

image representing binary data in which detail is key. We don't just want to get the gist of what the message image is. In the end, we will need to actually identify each bit as zero or one.

More algorithmically stated, *M* is convolved with *C*, and then scaled by a factor of α. The resultant image is then scaled and tiled so that it covers the same area as *I*. Finally the tiled convolutions are overlaid on *I* in all color records.

Note that this causes the watermark to look like noise, and if the random noise used for *C* is generated correctly (with values between -1 and 1), the added signal will have a zero mean thus not affecting the mean level of the base image.

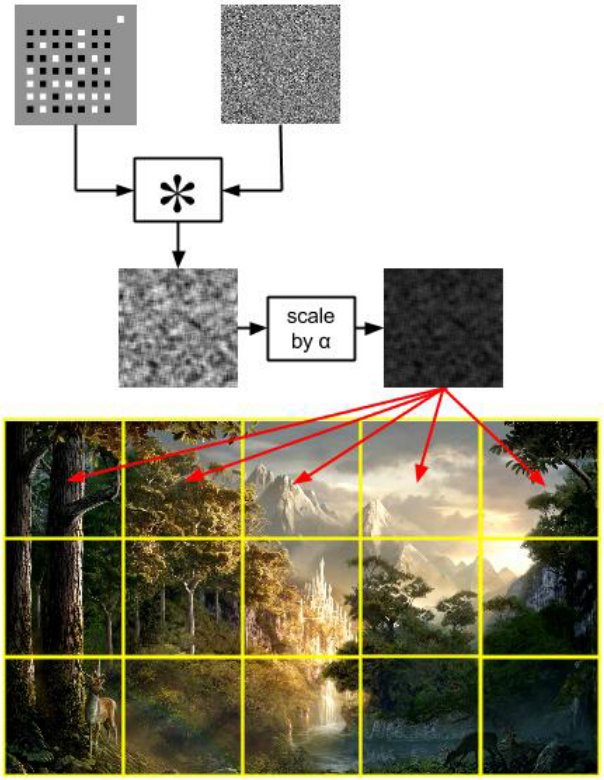

Fig. 3. The workflow of embedding proposed by Honsinger and Rabanni. The message image *M(x,y)* is convolved with a carrier image *C(x,y)* made of random noise. The resultant image is scaled by α, and overlaid in tiles on *I(x,y)*.

In Section III, extraction will be discussed, and it involves averaging the tiles found in the image. The more tiles there are in the frame, the more likely we are to extract a strong message image, since the signal in *I'* that comes from *M* will remain constant in each tile while the signal that came from *I* will vary from tile to tile and "even out" when averaged. However, it is important that a camera taking a video of the screen has a resolution high enough to capture the detail of the noise; therefore, the tiles need to be of a certain size. Furthermore, some pirates prefer to make certain that the entire screen is captured by keeping the edges of the screen in frame. Others (but fewer, as I discovered through my practical experiment at RIT) are more risky and prefer to zoom in on the screen, which captures the screen in higher detail, yet may cause them to lose some of the image on the edges. It is important that we do not begin tile placement with the first tile starting at (0,0) on the base image, but rather position tiles in regard to the center of the image because we need as many full tiles as possible to appear in the area captured by the camera.

It should also be noted that the exact pixel dimensions of the video do not matter when determining the size of each tile. Honsinger and Rabanni's application for embedding and extracting data from digital still image files could have set a fixed size, but this can't be done when dealing with a secondhand video capture. Whether a film is being shown on a classroom projector or an IMAX screen, the pirate will always try to fill his camera's field of view with the frame. Therefore, it doesn't matter if the video to be watermarked is 2k or 8k – it is the unknown resolution of whatever camera is used to shoot the secondhand copy that matters. Watermark tile size should be thought of as a percentage of the frame, not a fixed number of pixels. The only constraint I have made on tile size is that they must be square.

Keeping in mind that tile size must be determined as a percentage of the original video's dimensions, and that the edges of the frame may or may not be captured, I propose that we fill base image *I* in such a way that the watermark is placed in regard to the center of the image. I chose to fill *I* 4 watermark tiles high, not including the top and bottom 5%. Since our message image is a square, the tiles are square, and the length *S* a side should be resized (in pixels) as:

$$S = \left\lfloor \frac{Y - 2(Y \cdot d)}{h} \right\rfloor = \left\lfloor \frac{Y - 2(Y \cdot 0.05)}{4} \right\rfloor \tag{2}$$

where *Y* is the height of *I* in pixels. I find always a floor operation to be a less risky choice than rounding the final result. *d* and *h* can represent (if desired) alternatives to the constants I have already chosen: the percentage of the image untouched by full tiles at the top and bottom (0.05 as a decimal) and the number of tiles high (4), respectively.

The first full tile can be placed with its top at row:

$$y_{start} = \lceil 1 + y \cdot d \rceil = \lceil 1 + y \cdot 0.05 \rceil \tag{3}$$

where *Y* is the height of *I* in pixels. 4 (or your own *h*) squares of height *S* will cover the desired area in the center – vertically speaking.

Horizontally, our goal is to also to fill the center, so that cameras that may zoom in on the center of the image will obtain as many tiles as possible. We will want as many tiles of width *S* as will fit on the image horizontally, and centered horizontally. The number of tiles that can fit can be found by:

$$w = \left\lfloor \frac{X - 2(X \cdot d)}{S} \right\rfloor = \left\lfloor \frac{X - 2(X \cdot 0.05)}{S} \right\rfloor \tag{4}$$

where *X* is the width of *I* in pixels. Then, the first tile can be placed such that the left side of the tile is at row:

$$x_{start} = \left\lfloor \frac{C - (S \cdot w)}{2} \right\rfloor \tag{5}$$

Using my variables – forcing 4 tiles high, and 5% untouched on the sides – the tile placement of every image and the resulting noise look like as it does in figures 4a and 4b:

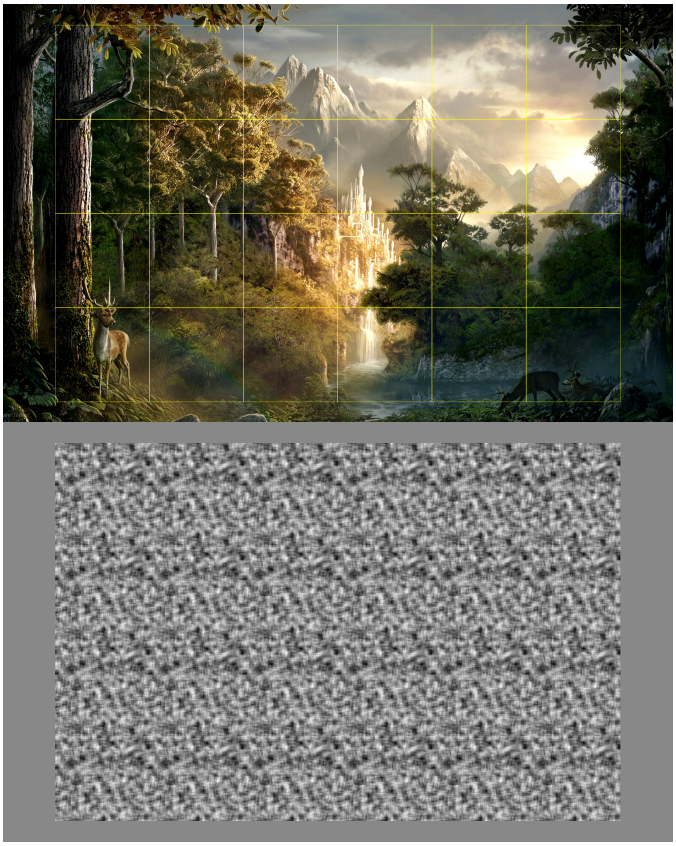

Fig. 4a. Tiling scheme and resultant noise for a high resolution image. The system of tiles makes the noise seem continuous.

Fig. 4b. Tiling scheme and resultant noise for a wider, but lower resolution image. Notice that the tiles are smaller, but the noise is still distinguishable. Since the image is wider, this allows for more tiles horizontally.

Ensuring that tiles fit in the center does not mean we should just leave the edges alone. Should the noise from 4a and 4b be overlaid on *I*, we would undoubtedly notice a sharp line between where there is noise and where there is not. Filling the empty areas with "half-tiles" will serve two purposes: first, it will prevent this visual problem that will show obvious watermarking, and second, it is very possible to include "half-tiles" in the extraction process, as will be discussed in a later section. The process of placing half-tiles is trivial, but necessary.

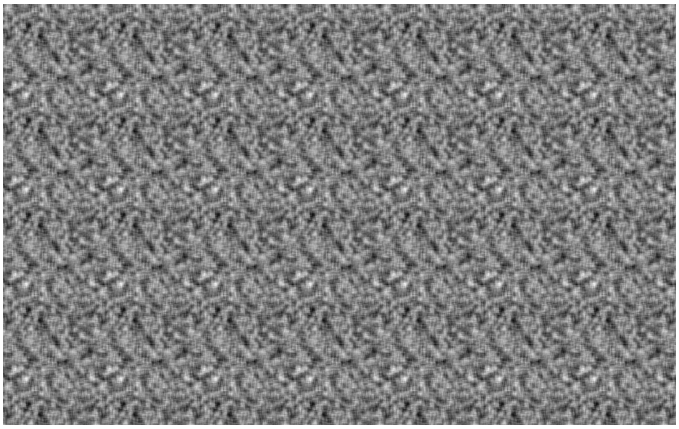

Fig. 4c. The resulting noise from image 4a, with half-tiles filled in.

The following is very important, and while it may be intuitive for some, I feel this should be said: The message image *M* should be resized from 17x17 to *S*x*S* pixels **before** generating a random noise of *S*x*S* and performing convolution, since performing the resizing on a small convolved image would both limit effective extraction and cause the resultant noise to appear very "boxy". Before convolution, *M* ought to be resized with a nearest-neighbor algorithm rather than attempting interpolation, since maintaining squares with sharp edges will be vital to determining the meaning of the image once it has been extracted.

The carrier image *C*, on the other hand, should always be interpolated whenever there is a need to resize it. In my application, I ended up not randomly generating *C* every time I made a new watermarked image. Instead, I stored a preset one in the application and resized it to *S*x*S* as needed. It is important to interpolate *C* whenever we resize it, so that all frequencies are maintained. Throughout the extraction process (as well as by the camera – in capture through rotation and perspective, and in signal processing through demosaicking, anti-aliasing, and such), *I'* will be resized, and when it is, it will be done smoothly, without hard edges. As both the watermarked image and the carrier image used to extract through cross-correlation are resized, the frequencies represented in both will become lower. If this change happens to both at the same rate, extraction will still be successful, but hard edges due to lack of interpolation on one and not the other is simply unacceptable during this extraction step.

A note on terminology: I did not refer to the **geometrical resizing** of the image as "scaling" as to spare confusion with the purpose of α, which scales the noise to a small **amplitude.** The constant α can be thought of as the strength of

watermarking. Generally, a lower α would make the watermark less visible, while a higher α would make the watermark more visible yet easier to extract. Figures 5a and 5b are two examples of the visibility determined by α, as applied to a sample double precision base image.

Fig. 5a. Base Image *I* with no watermarking

Fig. 5b. Image *I'* watermarked with α = 0.1

Fig. 5c. Image *I'* watermarked with α = 0.2

Note that for α ≥ 1, this operation would cause *I* to become nothing more than tiles of *M(x,y)*C(x,y)*, with little to no resemblance of *I*.

D. Adaptive Embedding – Hiding the Watermark

Initially, after successfully watermarking a few still images, I made my first attempts to find the level of watermarking (α) that was needed for successful extraction. After performing simple extraction (as will be discussed in section III), some results took me by surprise. It was odd to find that some images could get away with a very small α, and others required an amplitude so high that the image was entirely unpleasing to look at. The following figures are examples of what I came across:

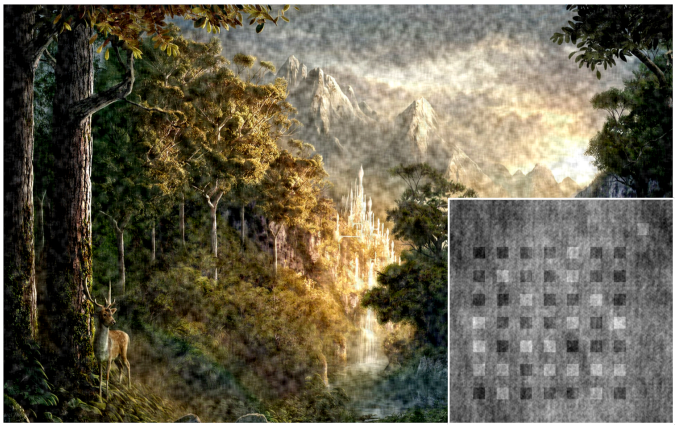

Fig. 6a. A watermarked image *I'* made with α = 0.5, along with the extracted message *M'*

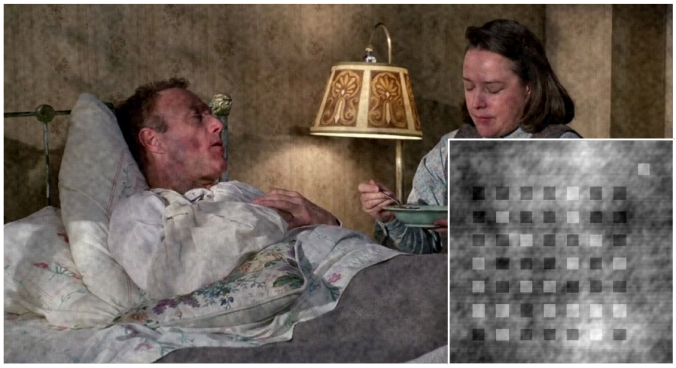

Fig. 6b. Another watermarked image (from a different *I*). This time the amplitude of the noise is only .15, and yet, *M'* is significantly clearer.

The reason for this soon became blaringly obvious. In figure 6a, for example, the busy detail of *I* in certain places (such as the bark and shrubbery) overcomes the added noise. This doesn't happen in flat areas of the image such as the sky. In figure 6b, on the other hand, *I* seems to be flatter almost everywhere, giving the added noise more of an impact. This issue called for a method to "even the playing field" and make the added noise more powerful in areas of high detail and more hidden in areas of low detail. I refer to this adjustment as "adaptive embedding".

The first step of adaptive embedding is to determine which part of any original image *I* has the most detail. There are several ways of accomplishing this, but I find that the one that works best is using gradient approximation filters as an energy metric, a method that was proposed for determining seam carving for content-aware resizing by Shai Avidan and Ariel Shamir of Mitsubishi Electric Research Labs. An image gradient by definition is:

$$\nabla = \sqrt{\frac{dx}{dI}^2 + \frac{dy}{dI}^2} \tag{6}$$

The approximation of the image gradient through filters is done as so:

First, we approximate the horizontal derivative (change) across an image through the filter

$$horz = I * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{7}$$

Then, we approximate the vertical derivative with

$$vert = I * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{8}$$

Finally, we find the overall energy of the image by adding the absolute values of the two approximations at each pixel like so:

$$E = |horz| + |vert| \tag{9}$$

Taking the absolute value of the derivatives ensures that edges will end up positive, regardless of what direction the change happens. The highest value an energy image E can contain is 6, and the results look something like the result in figure 7:

Fig. 7a. An image of a flower, with little energy in the background, hard edges between the petals, and a great amount of detail in the eye.

Fig. 7b. An energy image of the flower, where areas of detail have been found.

So now that we have an energy image, what should we do with it, in order to determine the strength of noise that should be added to each area of *I*? First, I am going to simplify the original embedding equation from

$$I'(x,y) = \alpha(M(x,y) * C(x,y)) + I(x,y) \tag{1}$$

to

$$I'(x,y) = I(x,y) + \alpha \cdot N(x,y) \tag{10}$$

where

$$N(x,y) = M(x,y) * C(x,y)$$

according to the previously mentioned algorithm.

Now, it has been established that it is not good enough to just multiply all of *N* by one value, because that doesn't consider the busy or flat content of *I*. I have implemented an adaptive embedding using:

$$I'(x,y) = I(x,y) + \left(\alpha \cdot \left(m \cdot E(x,y) + b\right) \cdot N(x,y)\right) \tag{11}$$

where *E* is the energy image determined in the previous step using the very same *I*, *b* (a scalar) is the minimum amplitude of noise required in any part of the image, and *m* (also a scalar) is the gain applied to the noise in areas with greater detail.

Using various images as *I*, I performed an optimization such that all images were equally extractable after being embedded using adaptive strength as in equation 11. With this optimization, I found that the best value for *m* is 1.0, and the best value for *b* is 0.1.

Now, instead of adding a constant amount of noise to the image of Middle Earth as in figure 4a, I will instead be adding the noise shown in figure 8.

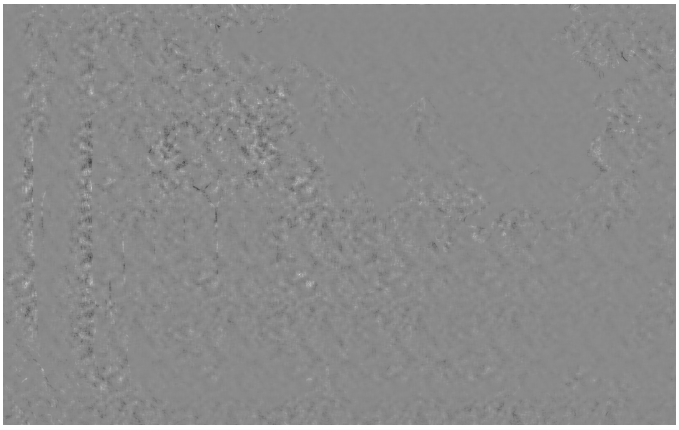

Fig. 8. Noise to be added to the image of Middle Earth, altered by the energy image calculated from the original. Note how much less the noise is in areas of sky (top right), as opposed to areas of foliage (left and top-left).

I have now accomplished two goals in one: I have hidden the watermark in areas of greater detail, which makes the image quality much more pleasing, and I have made extractability independent of image content. The portion of the calculated amplitude ($mE + b$) should not be thought of as a replacement for α, but rather an additional factor to ensure that the watermark is both hidden and extractable regardless of image content. α is still important for robustness and is the ultimate determination of whether a watermark can be completely extracted or not.

E. *Interframe Embedding for Video*

For the purpose of robustness, and since the theater location is static and the precision of showing time is hourly, the same message image will be applied to all frames. This is done with the hope that a pirated video will have so many frames that we will be able to get away with making α smaller on each frame.

Throughout my work on this project, some people have asked me why I didn't just put the noise on every 10 or 30 frames. After all, if the watermark exists in some places, it doesn't need to be everywhere, right?

There are two main reasons that the same watermark must be on every frame, in every color channel. We must never forget the reason why we are doing this. The video on screen is ultimately going to be captured by a camera. What if the camera's shutter speed and the display's refresh rate are not in sync? It would be a disaster if we had to average a frame with a watermark with a frame without, or worse yet – if we had to average two frames with two different watermarks!

Secondly, in the process of extraction, all frames will be averaged together. If all frames had the tiniest bit of constant noise, the average of all frames will be very close to that noise, and from there, extraction is a breeze. I would rather give every frame a little bit of noise and work with the average of all frames, than give every few frames a great deal of noise and have the horrendously long task of determining if each frame (of a secondhand video!) contains a watermark or doesn't.

III. Extracting Data

A. *Simple Extraction*

Extraction of the message image can be done with a simple reversal of the embedding process. In the following equation that represents the extraction process, *M'* represents the extracted message image, which is different from *M*, since we are performing this extraction on just *I'*, which now has inseparable contributions from both *I* and *M*. This operation also requires *C* – the same carrier image that was used in the original extraction process – no other carrier image will work. This is why the carrier image can be thought of as a key.

$$M'(x,y) = I'(x,y) \otimes C(x,y) \tag{12}$$

where ⊗ represents cyclic correlation.

To be more exact, this is done on each tile of *I*, as was done in the embedding process. Every tile from the embedding step has the same noise pattern. Summing enough tiles of *I'* can make the signal from *I* less significant and can bring out the convolved embedded image. When all of the tiles are summed, we can correlate with *C*, which will give us *M'*. The larger α was when embedding, the more like *M* that *M'* will appear.

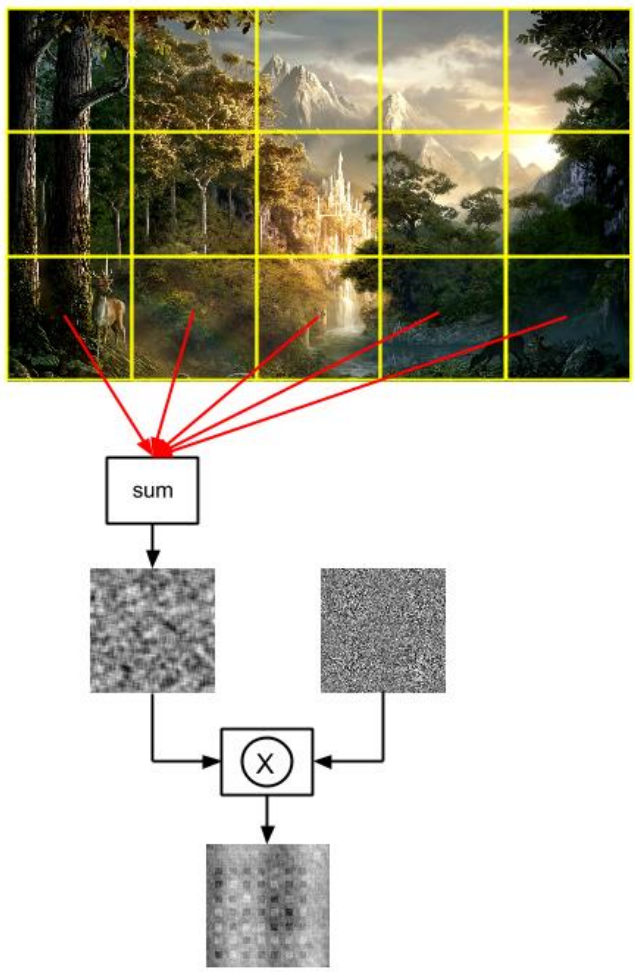

Fig. 9. The workflow of extraction proposed by Honsinger and Rabanni. The watermarked tiles of *I'(x,y)* are summed and correlated with carrier image *C(x,y)*. The resultant image is *M'*, the closest we will get to *M*.

Be aware that this extraction works, given that we know the exact position of the watermarked tiles. Unfortunately, in the process of shooting the video on screen with a camera and possible digital manipulation such as cropping, the position of the watermark is something that will need to be determined.

B. Finding the Position

Fortunately, convolution and correlation have a certain property that works in the best interest of this application. Since these operations are working in the Fourier (frequency) domain, we do not need to perfectly line up the key with the tiled message image locations in order for correlation to output a clear image – albeit the wrong one – **as long as all frequencies are represented**! This is best illustrated in the following example:

Fig. 10a. The correct tiling scheme yields a correctly positioned message image after sum and correlation.

Fig. 10b. Summing or averaging the tiles at an incorrect position still leaves the noise of each sampled area constant, and correlation does yield an image that is just as clear (since the same frequencies exist), yet is improperly positioned.

I propose that in order to find the position, we create an area that is twice the size of our expected *M'* and fill it with correlations made with all possible tiling patterns. I will refer to this area as the "message world", since it contains all possibilities of a result from tiled correlation, and it contains, without a doubt, the *M'* we are looking for. To make the message world image, we need to average each *M'* resulting from a number of tiling schemes (in a certain interval, since we don't need all tiling schemes starting at every pixel),

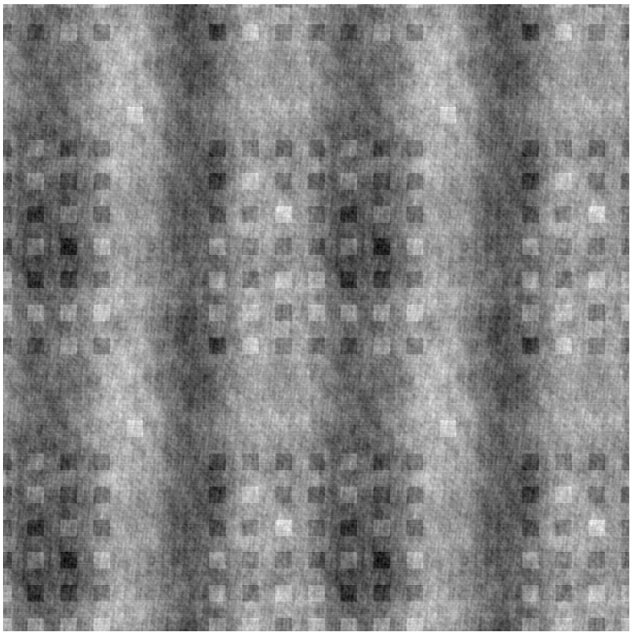

Fig. 11. A sample "world" image.

Sometimes, when trying a series of tiling patterns, there are times when it seems that there is less area available to us. This is when the half-tiles that we placed in the outer parts of the image can come in particularly useful. To add partial tiles to an average of tiles, we can't just add all tiles and divide by a scalar – if that were the case, we would have a lot of black space to average in! Rather, we would have two separate images that start at zero: one representing the values, and one representing the count of images taken into consideration for that pixel.

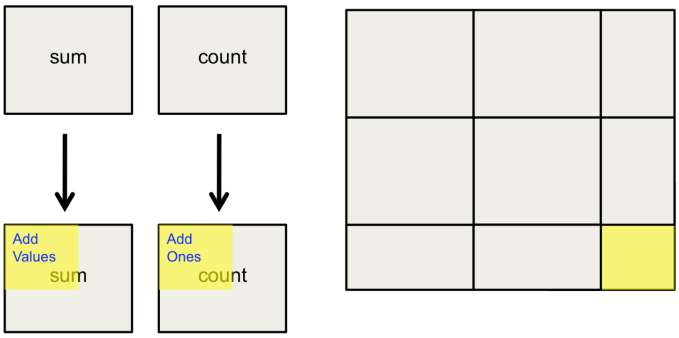

Fig. 12. When taking a partial tile into consideration for extraction, we first need to figure out which side of the final image it belongs on. Then, we add the values to that part of the sum image, and then we add ones to the count image. For values in the final image that don't belong to the half tile, we don't need to add anything.

After adding the every tile in the tiling scheme to these two images, we can divide the sum image by the count image and perform the correlation. Having the software remember where the starting coordinate for this tiling scheme was can help us place it in the correct location on the message world image.

Following the creation of the world image, we can find M' by cropping the world image to its expected size and running a variety of tests on the cropped area to see if it is in fact a valid message image. The cropped area that yields the best results is

chosen as our extracted *M'*.

An early method that I tried was to step through the world image pixel by pixel from 1 to the halfway point in each dimension, and crop to a square image of half the size (height and width) of the world image starting there. I would then determine the "likeness" of each square to a message image. I would split the image into 17 equal segments both vertically and horizontally. I would then check to see if each segment where there is expected to be a black or white square was greater than or less than the average of the squares that were expected to be gray. The amounts by which they were different were saved with each set of crop coordinates, and after all crops had been tried, the crop that yielded the highest difference between expected grays and expected blacks and whites was named the most likely *M'*. Unfortunately, this method was not only very imprecise (as blotches throughout the world image from averaging *I* made some data misleading), but it was also incredibly slow. I quickly realized I needed a faster and more accurate method.

The best method, however, for cropping the world image down to M' is by first determining the marginal standard deviations of every row and column. These results, much like the image itself, repeat themselves halfway through. The next step is to find the longest space between standard deviations over the 25th percentile. 3/5 through that dark area is the place to crop. This method has proved to be both fast and reliable. A demonstration is below:

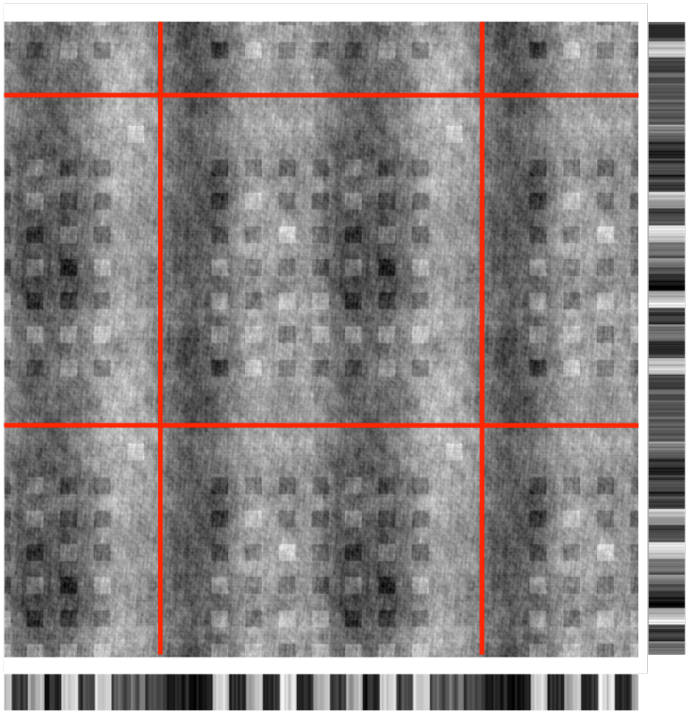

Fig. 13. Demonstration of the use of marginal standard deviations to perform the correct crop. The image in the center is the world image. The bars to the bottom and left represent the vertical and horizontal marginal standard deviations.

Now that we can find the watermark without knowing what position the embedding starts at, we can say that extraction is currently impervious to operations such as positioning and cropping. Now, we need to deal with other things that can happen through a secondhand recording, such as rotation, scale, and perspective.

C. Correcting for Rotation

Before finding the position, we can't just assume that linearly moving the carrier image through the image and correlating as we go along will yield any results, since a rotation after correlation would cause attempting correlation with a non-rotated key to fail. This situation calls for a special kind of correlation… and therefore, we have to change a few things about the embedding process.

To solve the problem of rotation invariance, I will be making use of the radon transform. The Radon transform of an image or shape simply sums the image over a series of projections. Imagine that we place a plane in front of every image and draw lines perpendicular from that plane through the shape, and then we sum everything that the line crosses and place the answer on that line. The result of each projection sum, then, is a 1d vector. Through many projections, the Radon transform places lines after each other, until the result looks something like the following:

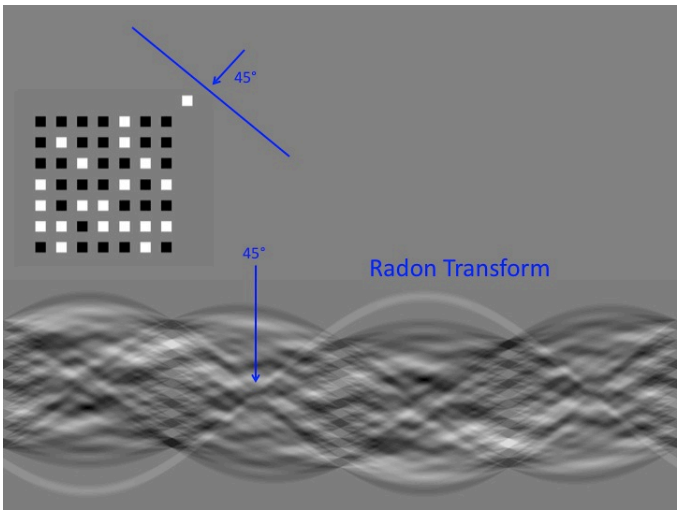

Fig. 14. Above, a message image. Below, a radon transform of the image, with projection sums rotated vertically and arranged one after another horizontally. Two corresponding positions: the projection plane of the original image and the sum vector of that plane, are pointed out.

After the Radon transform has been performed, the original image can be reconstructed from the transform, using a certain filter. The reason that the Radon transform is used for situations that need correction of or invariance to rotation is that no matter how the original image is rotated, all of the same projection sums will appear in the transform, just in a different order. Here, using certain details, such as the highest, lowest, or center point of the outermost white pixel, we can determine the correct rotation that is needed for the image.

Performing convolution and correlation in the Radon space is much more straightforward than one may think. Instead of using a 2d array of noise as *C*, I instead used a 1d vector of noise. Every time a projection sum was determined, I performed a cross convolution of that vector with a predetermined (and of course, resized, according to my

previous recommendation) noise vector, and after all projection sums had been made and placed into the transform, I performed a reconstruction.

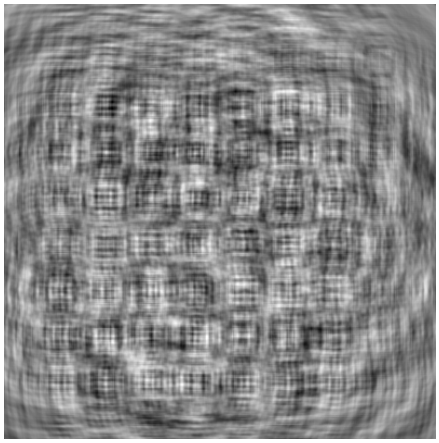

Fig. 15a. The message image *M*, convolved with a vector of random noise *C* in Radon transform space.

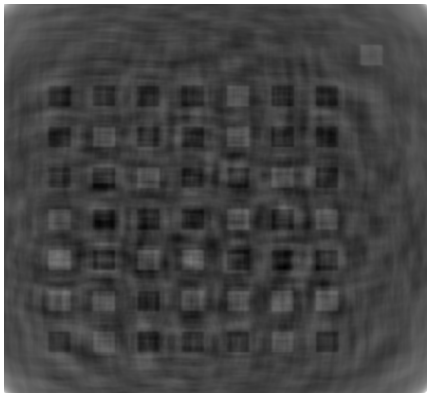

Fig. 15b. The convolved image, cross-correlated with C in radon transform space.

Note that there is some loss of detail between convolution and correlation in Radon transform space. This is because when we take the projection sums, we do it in discrete steps, and so some information is lost for good. Either way, regardless of how the image is rotated after convolution, we can always perform this new type of correlation with it, and we always get out a clear image.

This works well for one message image, but let's not forget that we are required to tile the image for robustness, and recall that all frequencies used in convolution must be present in correlation.

Imagine that in the following images, the black lines represent the tiling scheme, and the red box represents a single correlation.

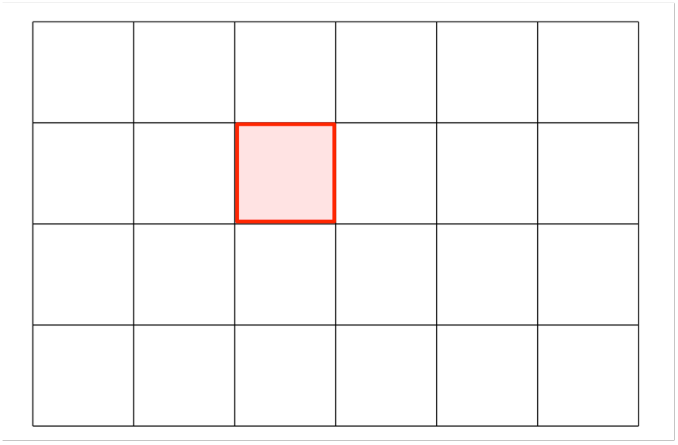

Fig. 16a. In this case, the correlation operation is lined up perfectly with the tiling scheme, and the watermark would be extracted without a problem. The "kernel" or image can be rotated 90 degrees in any direction, and operation would be just the same.

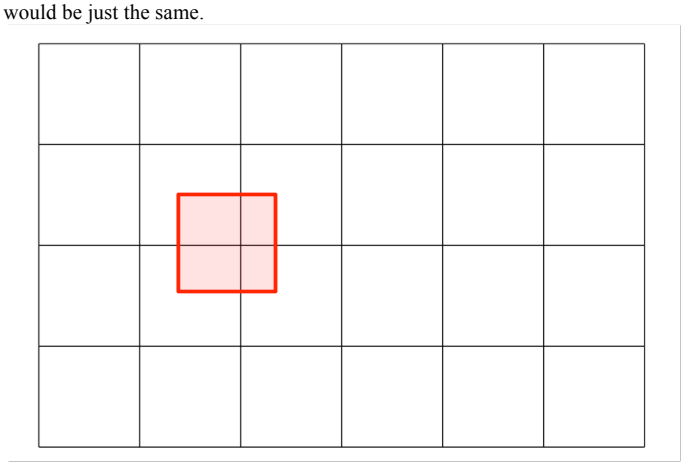

Fig. 16b. In this case, the correlation operation isn't quite lined up with the tiling scheme, but all frequencies used in correlation are represented, and the position invariant qualities of Fourier operations still hold true. The watermark, though not the correct one, would be perfectly clear, and the "world" method could be used to find the message. The "kernel" or image can be rotated 90 degrees in any direction, and operation would be just the same.

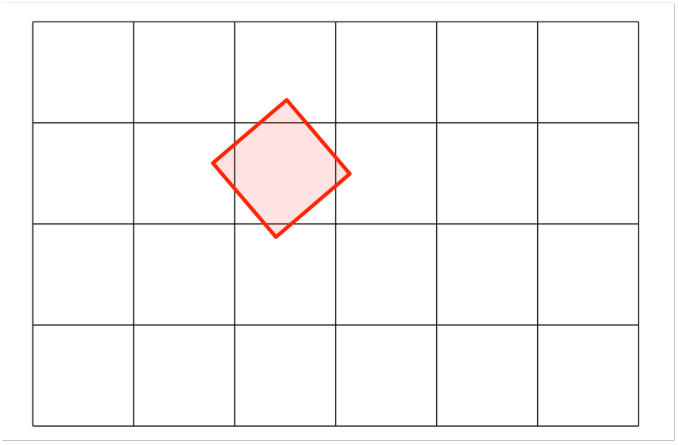

Fig. 16c. Unfortunately, this is the breaking point of the Radon transform, used in tiling. There is no guarantee that the exact frequencies used outside the central tile in this image match up with the ones unused within the central tile. This image shows a straight-on image with a rotated "kernel", but the same concept applies to rotated images.

So, given that we must use tiles, the Radon transform is only good to this project in the sense that we are invariant to

90 degrees of rotation. There is no need to worry about anything in between though … our next step, correcting for perspective, will ensure the image is rotated straight.

Before moving forward, though, performing correlation and convolution using the Radon transform makes the edges of our tiles seem blurry. This requires us to increase the number of tiling schemes we use to generate the world image, so the middle of the image is not blurry as well as the edges. Furthermore, the noise takes on a very very different appearance, but with a reasonably low α, no one seems to notice the difference.

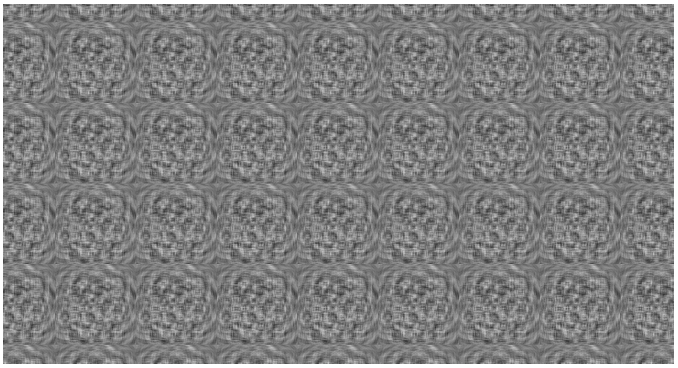

Fig. 17. The noise pattern seems to be more circular when performing convolution in Radon transform space. However, at a low amplitude and hidden well through adaptive embedding, the difference on an actual *I'* is near unnoticeable.

D. Correcting for Perspective

Correction for perspective is actually a very simple task. Basically, I run code that finds the corners of the frame with a Harris Corner Detector. I require the outside to be significantly darker than the inside for it to qualify as a frame edge. Granted, there are some frames throughout a video where there is a dark shadow or someone's hair in the edge of a frame, and the edge is near impossible to discern. I don't worry if I can't find 4 corners in a frame. If that is the case, then I just don't include that frame in the averaging of all frames. If I can find all four corners of the frame, I give the image a perspective transform such that all 4 corners get translated to the 4 corners of a frame whose size is determined by the resolution of the secondhand video. Instead of doing this for every frame, I sometimes choose to do this for every 5 or 10 frames, so that instead of looking for the four corners in every frame, I can interpolate where the corners should be between known frames, and I adjust them before I even find them, in order to save time. By this point, all frames have been averaged and turned into grayscale, and they should all be of a rectangular shape, so that there is no need to rotate before trying the different tiling schemes.

E. Correcting for Scale

Unfortunately, this must be the one part of extraction that is most demanding of the computer's processing power. We were able to get around needing to rotate the image 90 degrees, but unfortunately, we can't get around trying new scales, not while the algorithm still involves tiles.

At this stage, the video might be a clear average of all frames and look a lot like the noise in Figure 17, but there is no way of knowing if the tiling scheme matches the size of the carrier image *C* (the key) – or even if it has the correct aspect ratio to get any results out of convolution with spare keys. I propose that while keeping the size of *C* constant (let's call the length of *C* "*S*" as before), we change the height of the image from 2*S* to 5*S* while changing the width of the image from 3*S* to 9*S* in small intervals, in hopes that *C* will line up with *I*. To test how well each resizing performs, we can calculate a world image but not try to crop. Instead, we can use the one with the highest average marginal standard deviations after correlation to figure out which one is the best. When we have our best choice out of all the resizings, we can perform the crop out of the world image as we had before, and that should give us our extracted message image, *M'*.

F. From Message Image to Original Data

Since we can't help but pick up pieces of *I* along with our determination of *M*, a variety of factors (including α, the level of detail in the original base image *I*, the number of good frames available, and the amount of geometrical corrections that were required) can force *M'* to look quite different from our original *M*.

However, what matters is not how the *M'* looks, but rather if it can be decoded as the original data put in to *M*. In most cases, the following algorithms can do a better job at determining the meaning of an extracted message image than the human eye can.

As I mentioned before, with the failed attempts to find "likeness" of a message image, it is not enough to divide the image into 17x17 regions, figure out what pixels should be gray and which should be black and white. The absolute values don't tell the entire story because *M'* could be covered with "blotches" of black and white that have nothing to do with the squares representing bits. This is even more certain to happen when we are dealing with the effects of lost detail from performing two Fourier operations in Radon transform space. To find where squares begin and end, I prefer to use edge detection methods.

When we finally have the number, we can simply decode it back to its original data. This is best illustrated in our original example.

18834114623394

1) The last 3 digits represent the theater ID: 394.
2) The previous 5 digits represent the location: 14623 – which tells us to find the 394th theater in Rochester, New York.
3) All remaining digits at the front of the string are the number of hours that have passed since the reference date: 188341, and using the same reference date, will tell us that the movie was stolen at 12:00PM on December 25, 2013.

G. Correcting for Lens Distortion

It wasn't until I performed my practical demonstration of actually recording footage of a watermarked video in a theater that I was faced with the unexpected. On many ocassions, I

had simulated making and undoing changes to image rotation, scale, and perspective, digitally. Extracting all of these have proven to be successful. In real image capture, I forgot to think of one very important thing: distortion caused by the lens.

Fig. 19. An average frame from a secondhand copy of a watermarked film, captured with a Sony FS100 camera on a tripod. This is the average of only 15 frames which were watermarked very lightly. It bears a lot of resemblance to the noise in Figure 17, and it is for certain that there is a watermark on this film. The shot on the top, after undergoing perspective transformation, becomes the shot on the bottom. Because the shot on the top was never adjusted for lens warp, the shot on the bottom seems to bow inward if you look closely enough.

The most significant kind of lens distortion, and the one that is most trivial to correct is radial distortion, which is pictured below. When working with a particular camera, radial distortion is often solved by shooting a test chart and calibrating, but in this case, we never have any idea what camera a stolen video online originated from, so our best option is to simply tweak the variables used in calibration until it is fixed. It is also important that this step, if necessary, is done before anything else in the extraction process, since it is not realistic to undo lens distortions after transforming the image out of the original camera space.

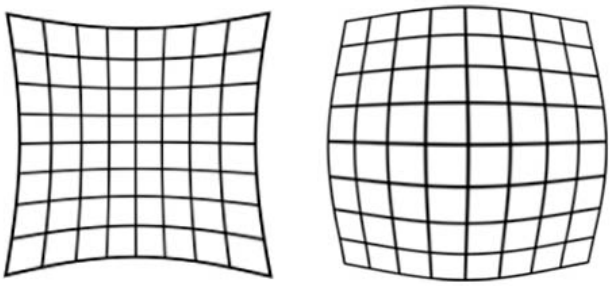

Fig. 20. Lens distortion represented by the manipulation of a square grid. On the left is negative radial distortion, also known as "pincushion", and to the right is positive radial distortion, also known as "barrel". *Source: Starizona: Adventures in Astronomy & Nature (www.starizona.com)*

The equations and calibration variables that model the warp created by radial lens distortion are as follow:

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (13)$$

$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (14)$$

where x and y are undistorted pixel coordinates, where (0,0) represents the center of an image, and r^2 is the distance of (x,y) from (0,0), found using $r^2 = x^2 + y^2$.

Tweaking k_1 and k_2 undo lens warp to a large degree, but at this point it is something that I must do manually. I have not yet thought of a way to auto-tune the process like I have with scale and crop.

IV. Robustness and Noticeablity

A. *The Effect of the Scale Constant α on Extraction*

Adaptive embedding has made all types of footage (live action, 3d animation, 2d animation, fast action, and slow drama) all extractable from a firsthand digital copy of a video that has a minimum α of about 0.2, but that is just for one still frame. If we use an average of only 10 frames, we can get away with a watermarking of less than 0.1 for each frame!

It is currently difficult to say what level of watermarking is needed for successful extraction of a secondhand video, since I am still having difficulty with manually correcting lens warp and have not yet been able to determine the meaning of an extracted *M'*, regardless of α. I continue to make progress every day.

About the nature of secondhand capture, I will say this: For each frame, watermarked with a very small amplitude as it should be, it is difficult for a camera to capture the noise in just one frame, as cameras will introduce their own noise, demosaicking, and the like. From my visual observations, the more frames that are averaged from a secondhand copy, the less of a problem this becomes. The noise resulting from figure 19 (a secondhand copy), and that which would result from a firsthand copy of the same video are actually quite similar. The difference, of course, would be lens warp that prevents correlation from being successful. In fact, digitally

rotating, resizing, cropping, and applying perspective transforms in Photoshop on firsthand copies have all been done and able to be undone in extraction. Section V. contains images of results from secondhand extraction thus far.

Since the watermark is embedded in all color channels, and before trying different tiling positions, *I'* is converted to grayscale and normalized 0 to 1 (linear), extraction is successful in firsthand copies, regardless of color correction. This includes adjustments to brightness, contrast, and saturation. It was even successful after bringing the blue channel completely down to 0 or completely up to 1. Blurring the image had a greater effect on the extraction, but a significant amount needed to be done for it to fail. Where the extraction failed the most was on resizing videos to lower than about 500 pixels wide. After averaging frames of videos with lower resolutions you can tell that there is a watermark there, since the circular noise pattern from figures 17 and 19 can clearly be seen), but after correlation, decoding it back to a number is near impossible, no matter how high α is. Had the embedded message been less than 49 bits, I am certain that even lower resolutions would have yielded better results.

B. *Noticeability in Video – A Psychophysical Experiment*

Now that I know what level of watermarking is needed on a video in order to be extracted, at least for a firsthand copy, I needed to determine what level of watermarking is unpleasant to the human visual system, and hope that the threshold is greater than what is needed to successfully extract.

For the experiment, I would have people sit in what was as close to theater conditions as possible. They would sit in a dark room, in front of a display calibrated to Rec. 709, 3 pictures heights away from the screen, and then they would watch their choice of live action, 3d animation, or 2d animation. 20 clips would be shown, looping 3 times each. Each clip was the same content, but one side of the video (left or right) was watermarked with a certain amplitude, and the other was not. The subjects had to choose which side of the video they thought was watermarked. This was a forced choice experiment, which means that they had to make a decision. If they didn't know which side of the video was watermarked, they had to give their best guess.

I made an effort to include people of all ages in this experiment. Though most of the individuals that took this experiment are people in my field who evaluate image quality almost every day, I made an effort to include some non-experts as well.

In forced choice psychophysical experiments such as this one, the Just Noticeable Difference is found where 75% of subjects give the correct response. In this case, it is where 75% of subjects guess the correct side that the watermark is on. It is accepted that 50% of the time, an observer will get the answer correct by chance. This means that when 75% of the responses are correct, that is where the average person begins to notice a difference. In all three cases, the bare minimum strength constant needed to extract the watermark successfully is less than the Just Noticeable Difference, but not by very much.

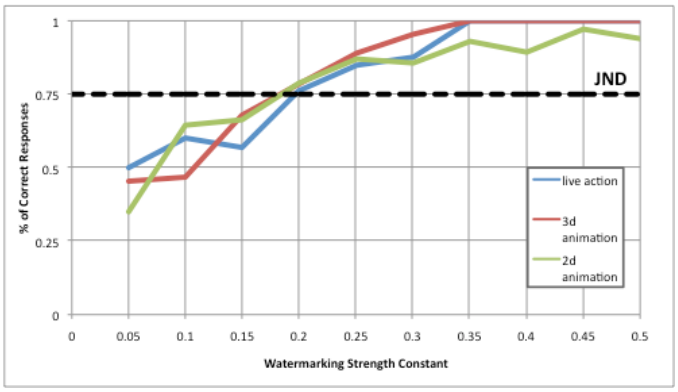

Fig. 21. The results of my psychophysical experiment: the percentage of correct responses vs. the watermarking strength constant . For live action, the just noticeable difference was $\alpha = 0.2$. For 3d and 2d animation, it was $\alpha = 0.175$ – approximately where 75% of correct responses were.

Furthermore, I asked my subjects to comment on what they noticed the most – in what parts of the video did the noise stand out the most? The answer was almost the same for everyone: it was when the camera moved. Since the noise is constant on the frame, the noise truly stands out when the entire background shifts, especially if there is a motion blur. This leads me to believe that it may not be such a bad idea for artists to manually choose α for each frame.

Subjects also mentioned that (especially for the live action clip), they probably would not have noticed anything wrong unless I had told them exactly what the fixed noise pattern looks like.

V. Practical Demonstration

A. *Objectives*

As I have already mentioned, I performed a simulation of piracy where I would play a watermarked film in a theater on campus and have people attempt to record it with their phones and cameras. The objective of performing a practical demonstration is to observe what techniques – if any – people use when taking a secondhand copy of a video, to try to capture the video on a variety of devices, and last but not least to see if we can actually extract the watermark from a secondhand video.

B. *Design*

In a dark theater, I played a five minute video consisting of both live action and 3d animation clips, both watermarked with a level of $\alpha = 0.1$ per frame. I asked everyone in the room, who were all in different locations throughout the room, to take out their cell phones and tablets and try to record the video. Meanwhile, I set up a Sony FS100 and a Canon 5D on two far sides of the room, to also capture the video on screen.

C. *Results*

Before I begin discussing the successes and failures of extracting the watermark from these secondhand videos, I must mention how interesting it was to find that every single person that I got footage from made an effort to make sure the edges of the video frame were in the camera's field of view.

No one, not one person, zoomed in. I gave no instructions one way or the other.

At this stage, I am still trying to figure out exactly how much lens warp is needed for extraction. The closest I have come to success is as follows:

Fig. 22a. A single frame of a secondhand video of a watermarked film, captured with a Canon 5D MK III. The noise is hardly noticeable.

Fig. 22b. An average of 14 frames, having undergone a perspective transformation after a radial lens correction of k1 = $1X10^{-8}$. A noise pattern becomes evident, and whether we can extract meaning from it or not, we can tell that there is a watermark.

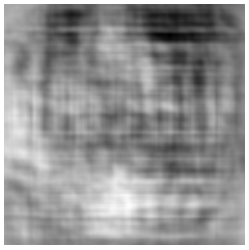

Fig. 22c. From 22b, this image came from extraction as *M'*. The meaning of the image is not clear, but we can just begin to make out the hard outline of squares in the places we might expect them to be.

I expect that as I continue to explore lens distortion, the results of this and similar videos will continue to improve.

VI. Practical Applications

After this has been tested a little more, and on different kinds of footage, I would like to see watermarking become more commonplace in theaters. I imagine that my extraction software would exist out on the Internet somewhere like a bot and would hop from site to site, looking for videos that could possibly have the watermark on it. When a watermark like the one in this paper is found on such a video, we know immediately that it is copyright material. The video ought to be immediately taken down, we can try to find where the video came from, as well as the IP Address from which the video was uploaded.

As far as the embedding process, it is a good idea to consider where this happens, whether it is something to be added in post production or something that is done inside of the projector as the film plays. I am leaning towards the latter, because then it would be easy to automatically generate the time and location of the screening, as opposed to having to make countless versions for countless show times. Although on the other hand, I think it should be the decision of filmmakers to choose the strength constant α (for each frame), rather than being stuck with what the projector chooses for them. Perhaps they could upload a decision list of a sort to the projector to accompany the film.

Finally, for this to work exactly as is, there would need to be a census of all movie theaters in the world, so that the organization overseeing this system would be able to associate every theater with an ID number.

VII. Conclusion

As a senior project and an important requirement for graduation, I felt that this project was a good choice, and I don't regret choosing it for one minute. I spent most of my time working on this in programming and problem solving. Around every corner, there was a new puzzle to solve, and I very much enjoy that. It was also good to have the chance to combine new and old ideas into an application that could one day save my beloved industry a great deal of money.

As with all research, there is always plenty of room for improvement and elaboration. I am particularly excited about pursuing this project further, and I can't wait to see all of this application's potential. Some of the things I hope to accomplish in the future include, but are not limited to:

- While performing perspective transformations, analyzing the x, y, and z rotations to determine the position of the pirate in the theater.
- Creating a watermarking interface in which the watermarking level α can be automatically determined by analyzing the motion of the scene (i.e. by how much the current frame is different than the previous ones), or where α can be manually set by a user based on artistic preference.
- Automatically tuning the lens warp adjustments.
- Figuring out the appropriate course of action for a secondhand video in which the edges of the frame are not visible.

Demonstrations of the algorithms presented in this paper, as well as future work, can and will be found online at www.jakedeboer.com/piracy. On this website, watermarked videos will also be available for download, so one can get a sense of what different levels of watermarking look like on

video as opposed to the still images in print here. I also gladly accept emails with questions or suggestions.

ACKNOWLEDGMENTS

I would like to first and foremost thank my thesis advisor and mentor, Dr. Carl Salvaggio, for his limitless supply of information regarding anything and everything related to the field of imaging.

I would like to thank my professors David Long and Ricardo Figueroa and my classmates Joshua Berkowitz, Chris Brands, Michael Richos, David Villareal, Jaclyn Pytlarz, Colleen DiVincenzo, and Kyle Mooney for their valuable feedback, input, and fresh ideas.

I am additionally grateful for the 51 individuals who have volunteered their time to participate in my psychophysical experiment and for those who also participated in the practical demonstration of piracy. I would also like to acknowledge the still frames from the films *Misery* (1990) and *Lord of the Rings: The Fellowship of the Ring* (2001) that I used to demonstrate watermarking throughout this paper. I should also mention that I chose to use video clips from well known films in my psychophysical experiments so as not to distract my subjects with unfamiliar content – for 2D animation, *The Lion King* (1994); for 3D animation, *Wreck-It Ralph* (2012); and for live action, *The Avengers* (2012). I would like to thank my fellow RIT alums Steven Markowitz and James Nevada for allowing me to embed a watermark on their films for my practical demonstration of theater piracy, and finally the federal government website www.federalgovernmentzipcodes.us for providing a database of US ZIP codes.

REFERENCES

[1] C. Honsinger and M. Rabbani, "Data Embedding Using Phase Dispersion," Eastman Kodak Company, 2000.

[2] K. M. Iftekharuddin, F. Ahmed, and M. A. Karim, "Rotation-invariant target recognition using an amplitude-coupled minimum-average ocrrelation-energy filer," Society of Photo-Optical Instrumentation Engineers, 1996.

[3] J. Rosen and J. Shamir, "Circular harmonic phase filters for efficient rotation invariant pattern recognition," Optical Society of America, 1988.

[4] A. Coronel-Beltran and J. Alvarez-Borrego "Invariant Nonlinear Correlations via Fourier Transform," *Fourier Transform - Signal Processing*, Dr Salih Salih (Ed.), 2012. ISBN: 978-953-51-0453-7.

[5] G. F. Schils and D. W. Sweeney, "Rotationally invariant correlation filtering," Scandia National Laboratories, Livermore, California. 1985.

[6] C. Salvaggio and P. Salvaggio "Perspective Transformation," Rochester Institute of Technology, 2011.

[7] L. Kong, W. Zhang, S. Zhang, B. Zhou, "Radon transform and the modified envelope correlation method for ISAR imaging of multi-target," Research Institute of Electronic Science and Technology, Chengdu, China, 610054.

[8] R. L. Easton Jr., A. J. Ticknor, and H. H. Barrett, "Two-dimensional coplex Fourier transform via the Radon transform," University of Arizona, Optical Sciences Center, Tuscon, Arizona 83721. 1985.

[9] P. Wolf, *Elements of Photogrammetry*. (pages 74-79, 102-104). McGraw-Hill, Inc. 1974. ISBN 0-07-071345-6

[10] S. Avidan and A. Shamir. "Seam carving for content-aware image resizing," *ACM Translations on Graphics – Proceedings of ACM SIGGRAPH 2007*. Volume 26, Issue 3, July 2007.

Jacob S. DeBoer is an undergraduate student at Rochester Institute of Technology in Rochester New York, expecting to receive his B.S. in motion picture science in May 2014.

He is a freelance database developer and consultant and has most recently been employed as a Post Production Assistant at IMAX in Santa Monica, CA.

Mr. DeBoer is additionally a member of the Society of Motion Picture and Television Engineers.