Increasing the capturing angle in print-cam robust watermarking

Anu Pramila, Anja Keskinarkaus, Tapio Seppänen

Physiological Signal Analysis Team, Center for Machine Vision and Signal Analysis, University of Oulu, Oulu, Finland

Abstract

It is nowadays more probable that a print media is captured and shared with a mobile phone than with a scanner. The reasons for photographing the print range from intention of copying the image to simply sharing an interesting add with friends. Watermarking offers a solution for carrying side information in the images, and if the watermarking method being used is robust to the print-cam process, the information can be read with a mobile phone camera. In this paper, we present a print-cam robust watermarking method that is also implemented on a mobile phone and evaluated with user tests. Especially, the lens focusing problem when the picture is captured in a wide angle with respect to the printout is addressed. The results show that the method is highly robust to capturing the watermark without errors in angles up to 60° with processing times that are acceptable for real-life applications.

Keywords: focal stack optimization, mobile phone application, watermark application, camera phone, computational photography

1. Introduction

The aim of this paper is to present an algorithm for reading a watermark from a printed image with a mobile phone camera with wide angles of capture. To achieve this, three research fields are joined: print-cam robust image watermarking, mobile phone applications and computational photography.

The aim of the print-cam robust watermarking is to link the physical world with the digital without compromising the aesthetics of the artwork. The reasons for hiding information vary from increasing security to offering beneficial information to the recipient. However, in real world applications, several things make the print-cam robustness challenging. The watermark should be simultaneously robust to AD/DA transformations, rotations in 3D, scaling and translation. In addition, human interaction, JPEG distortions, lighting variations and camera related distortions, such as barrel distortions and focusing, need to be considered as well. (Pramila et al., 2007)

In the scenario presented in Fig. 1, the watermarked image is located in a magazine laying on a table. The image could also be placed on a wall and people with different heights could take pictures of it from different angles or around a cluster of other people. Specifically, the lens might not focus correctly to the image taken at a large angle causing the watermark extraction to fail. It should be noted that the watermarked image is never published electronically and therefore intentional attacks against the watermark are not considered here.

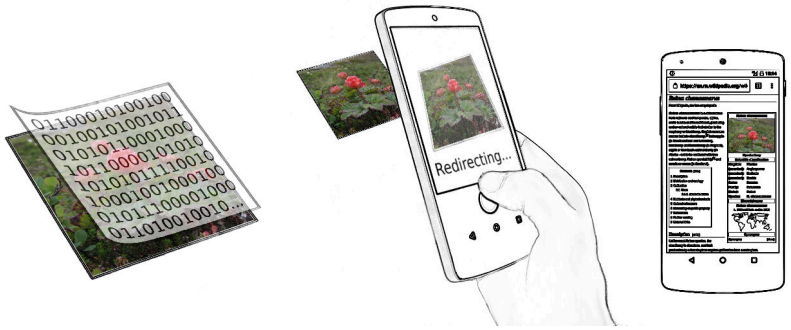


Figure 1: The scenario. The watermark is embedded in an image. The watermark contains a link to a webpage that the user can read with his/her camera phone.

Most of the proposed print-cam robust watermarking methods have been tested in relatively constricted settings and only a few of them have been proved to work on a mobile phone. Most of the methods assume that the user is able to point the camera straight in relation to the watermark and so the distortions

are minimized.

The first print-cam robust watermarking method was proposed by Katayama et al. (2004). The method was based on a sinusoidal watermark pattern and a frame for synchronization. They noted that utilizing a frame is not a problem as it shows the user that a watermark is present.

Kim et al. (2006) relied on an autocorrelation function for watermark extraction. They embedded a pseudo-random vector by tiling it repeatedly and later detected the peaks that were formed by the autocorrelation function. The method was tested on a digital camera on a tripod and human interaction was required for the final extraction of the watermark to minimize geometrical distortions. Likewise, Pramila et al. (2012) used autocorrelation although they employed directed patterns. The message was encoded in the angle of these patterns and the method was robust to $\pm 20^{\circ}$ of the tilt of the optical axis.

Yamada and Kamitani (2013) placed the information only in some locations in the image in order to preserve quality of the image. The watermark was embedded with spread spectrum techniques and read from the video feed by trying to find the watermark from each frame in turn. The method required the user to keep the camera near perpendicular to the watermark.

All these methods operate in the spatial domain and only a few have proposed methods in other domains. Delgado-Guillen et al. (2013) operated their method in log-polar transformations and Pramila et al. (2008) divided the method across several domains.

In addition, commercial applications have been launched. Most notably, Digimarc (2015) introduced an application called Discover. The aim of the application was to connect users from magazine pages to the Internet.

All the aforementioned print-cam robust watermarking algorithms rely on the autofocus feature of the camera. However, when the image is captured in a large angle, autofocus does not work. None of the methods consider the large angles at which depth of field of the camera comes into play and parts of the image are unfocused. Increasing the depth of field would solve the focusing issues and traditionally this is achieved by decreasing the size of aperture. However,

this requires more lighting or longer exposure time, which might introduce distortions such as the motion blur. In addition, mobile devices often have a fixed aperture and changing the depth of field is not possible.

Computational photography refers to techniques for capturing and processing images digitally, therefore overcoming the limitations of traditional optics. The effect of increasing the depth of field can be achieved with computational photography by capturing a focal stack, a series of images focused at different depths, and building a new image with an extended depth of field. Several methods have been proposed, unfortunately, only a few of them take into account the movement between images that occurs when images are captured free handedly. None have been suggested to be used in combination with watermarking.

One of the most promising methods was proposed by Vaquero et al. (2011) who utilized FCam programmable-camera software stack (Adams et al., 2010). For better efficiency and results, they noted that not all of the images in the focal stack are required for building an all-in-focus image. Their method was based on sweeping the focus of the camera lens and selecting the optimal set of images to be taken according to small sharpness maps.

However, all the cameras cannot focus the lens on specific focal points, nor sweep the lens. Sakurikar and Narayanan (2014) approached this problem by using camera autofocus so that the viewfinder was divided into 16 blocks and autofocus algorithm was pointed towards each block in order. Overly similar images were removed before registration. Solh (2014) selected three predetermined focal distances and captured pictures at these lens locations. The frames were then aligned and fused. Zhang et al. (2013) captured the whole focal stack but removed the worst images from the stack. They took advantage of the IMU (Inertial Measurement Unit) to receive information about the movements of the phone, so that the sharp images could be discerned from the blurry images more easily.

Here, the print-cam robust watermarking is combined with the all-in-focus imaging into a system that is robust to severe 3D distortions. To solve the focusing issues, an algorithm is proposed that employs computational photography

to improve the sharpness of the images for the watermark extraction.

The algorithm, denoted here as WCAM (Watermark CAMera), begins by collecting a focal stack that is scaled down to a fraction of a size for fast calculations. The number of images in the stack is then optimized by selecting a small amount of the images in the stack that contain most of the information about the scene. The selected images are registered and blended into an all-in-focus image. From this scaled-down all-in-focus image, an approximation of the correct watermark location and alignment in the original images can be determined.

The method takes advantage of our previous results on combining computational photography and robust watermarking (Pramila et al., 2017). An evolved algorithm is presented which is 35 times faster by taking better into account the inherent features of the selected watermarking method and demands of camera phones. The method is robust up to capturing angles of 60° with varying distances. In addition to testing capturing angles, the robustness is evaluated with user tests.

In Section 2, the embedding process of the watermark is explained. In Section 3, the implementation of the watermark extraction is presented and the required methodologies are explained. The method is implemented on a camera phone and tested with users. The results have been collected in Section 4, and finally Conclusion is given in Section 5.

2. Watermark embedding and printing

The scenario imposes requirements for a moderate amount of geometrical distortions that occur as the approximations of transformation matrices are utilized instead of actual matrices as well as medium capacity. The blind print-cam robust method by Pramila et al. (2012) fulfils most of the requirements. With few changes, mainly in extraction part, the method adapts well to the scenario.

The overview of the watermark embedding method is illustrated in Fig. 2. The watermark is embedded in the Y-channel of YCbCr transformation of the

image by dividing the image into k blocks and embedding the watermark in the blocks. Unlike in (Pramila et al., 2012) in which $k = 1...9$, in here $k = 1...16$ in order to increase the capacity. The size of the block depends, therefore, on the size of the image. For an image of size 512×512 this means a blocksize of 128×128. One block is reserved for watermark detection and synchronization. Four bits are embedded in each of the remaining blocks resulting in $4*(16-1) = 60$ bits of capacity. This enables the suggested scenario in which the watermark conains a link to a webpage.

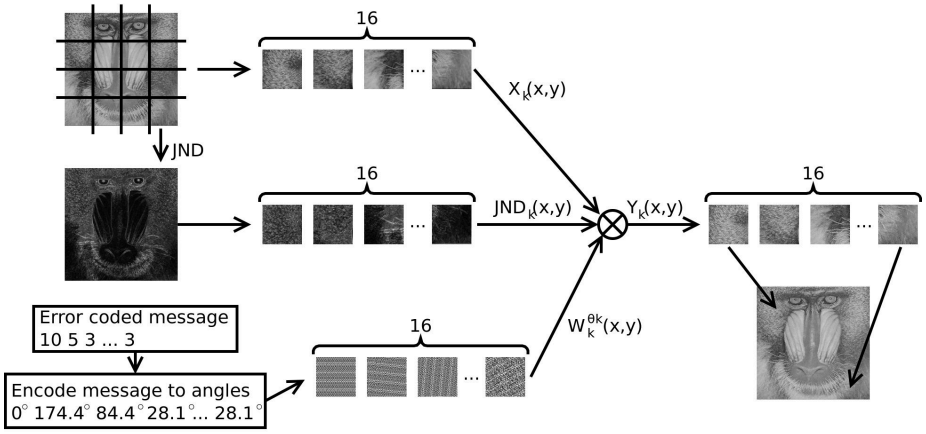


Figure 2: The algorithm for embedding the watermark.

The watermark message is encoded in the angles of pseudorandom patterns. The message was first error coded with Reed-Solomon error correction coding (Reed and Solomon, 1960) and then with Gray coding to minimize effect of errors. The error-coded message was then divided into bit sequences and each sequence of 4-bits was assigned an angle θ_k. A pattern W was then rotated according to the angles. W was build by tiling a small pseudorandom pattern, the size of which was experimentally determined to be 28×7.

The embedding strength per pixel is controlled with JND (Just Noticeable Difference) which is calculated with the method by Chou and Li (1995). The watermark is finally embedded in the image X with

$$Y_k(x,y) = X_k(x,y) + W_k^{\theta k}(x,y)(\delta_1 JND(x,y) + \delta_2(1 - JND(x,y))), \tag{1}$$

where Y_k is the kth watermarked image block, X_k is the kth preprocessed image block and $W_k^{\theta k}$ is the two-dimensional pattern that has been rotated according to the kth angle θ_k. The two parameters δ_1 and δ_2 govern the embedding strength. More details of the embedding method can be found in (Pramila et al., 2012)

In addition, a frame is placed around the image, enabling image rectification using frame corners. This is needed against severe rotations of $> 30^\circ$ in 3D, and blind block division, as suggested by Pramila et al. (2012), is not feasible.

3. Watermark extraction and focal stack optimization

The algorithm should work on a mobile phone and consequently it should be fast and efficient without compromising watermark robustness. Fig. 3 shows the flow of the watermark extraction process, which will be presented in detail in the following subsections.

3.1. Capturing process

The focal stack is built by first determining the initial focus point with the inbuilt autofocus routine of the camera and then capturing images at equal distances of

$$\begin{aligned} f_{step} &= 1.0/100.0 \\ f_{next} &= 1.0/((1.0/f_{original}) - (3.0 * f_{step}) + (f_{step} * (i-1))), \end{aligned} \tag{2}$$

in which f-values are given in dioptres and $i = 1...N$. Here, N was chosen to be 10.

To speed up processing times and save memory, each image is scaled to a size of half a megapixel, that is, 816×612 after capturing. The full sized original images are saved in memory for later processing.

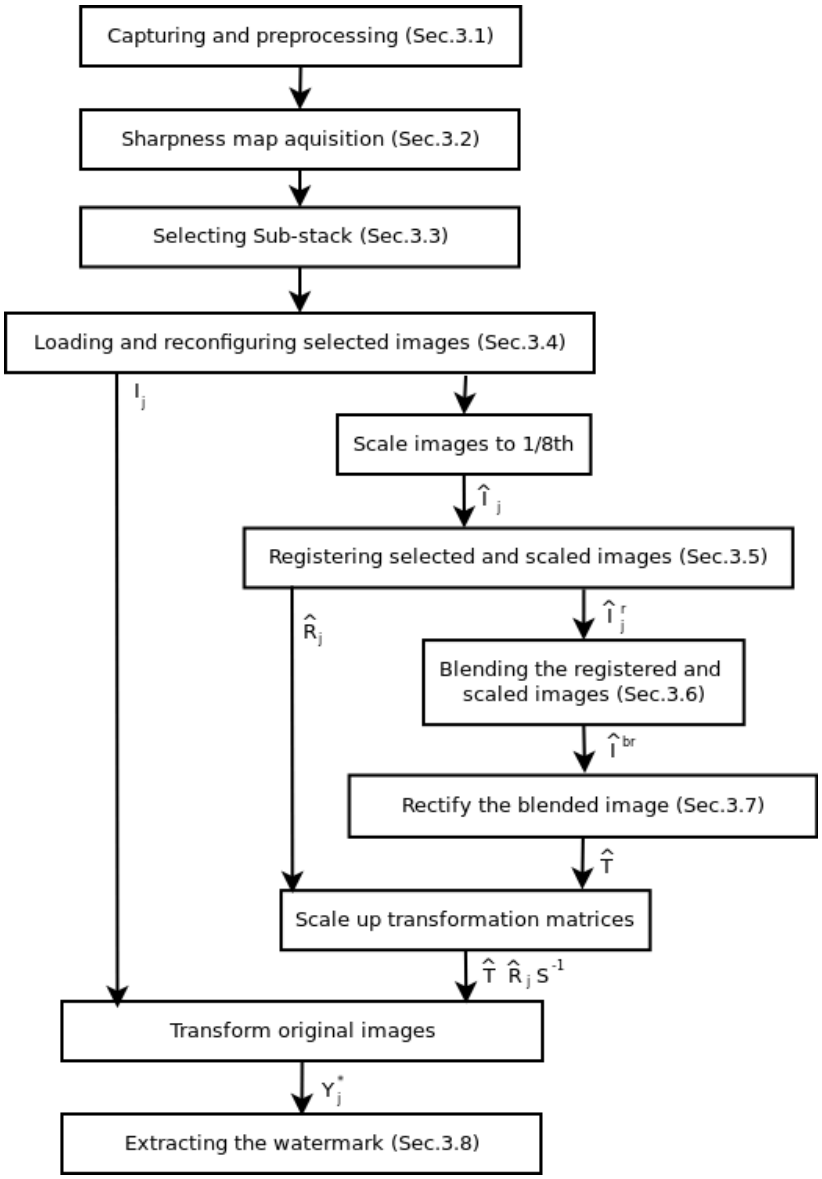


Figure 3: The flow of the watermark extraction algorithm implementation.

3.2. Sharpness map acquisition

The scaled images are used for the fast optimization of the focal stack. For this, sharpness maps are calculated from the scaled images. The sharpness maps work as a crude depth map, showing which focal lengths contain the most of the information about the scene.

Let us assume that there are N images in the focal stack and $\hat{I}_i$, $i = 1...N$ denotes a scaled image. The sharpness maps are calculated by taking the absolute values of Laplacian for each image as in Eq. 3

$$L_i = \left| \frac{\partial^2 \hat{I}_i}{\partial^2 x} + \frac{\partial^2 \hat{I}_i}{\partial^2 y} \right|. \tag{3}$$

The obtained images L_i, $i = 1...N$ are then divided into blocks so that the final sharpness maps are the size of 96×72. This is done by averaging the values in each block. Finally, the maps are normalized so that the maximum value of the maps is set to 1 and minimum to 0. Let us denote these sharpness maps so that S_i is the sharpness map of the i^{th} image in the stack.

3.3. Optimizing the size of the focal stack

Optimization of the size of the focal stack speeds up the process of watermark extraction by selecting a subset of the images that contains non-overlapping information about the scene. Also the robustness increases as having fewer images decreases the probability of making errors in registration while building the all-in-focus image. Here, the optimization process is inspired by Vaquero et al. (2011) and explained in detail in (Pramila et al., 2017).

First, the maps are segmented into foreground and background areas. Standard deviation σ is calculated for each element $S(u, v)$, in which u and v go through all the elements in S. Each element is classified as foreground if

$$\sigma > 0.1 \times \sigma_{max}, \tag{4}$$

in which the constant 0.1 was experimentally determined to yield accurate results for the segmentation.

Next, the foreground elements are divided into blurry and sharp regions. An element (u, v) in the i^{th} map is determined as sharp if $S_{max}(u, v) == S_i(u, v)$, or $S_{max}(u, v) - S_i(u, v) < S_{max} \times t$ and the element in the neighbouring map is sharp. That is, element $S_{i-1}(u, v)$ or $S_{i+1}(u, v)$ is sharp. The threshold t is described as an approximate number of images that contain valuable information about the scene. It is determined by saving for each element (u, v) the image that maximizes the sharpness and calculating the occurrences of each image. The occurrence values are saved in a vector and the values are normalized to the sum of 1. Experiments showed that all images with an occurrence number less than 0.1 can be considered noise.

After the previous steps, the sharpness maps now show in binary form which images contain sharp regions and at which locations. In reality, the sharpness changes between images gradually and therefore the binary maps B_i are first processed with a small 3×3 Gaussian kernel g with a standard deviation of $\sigma_g = 0.5$ so that

$$G_i(u, v) = \max_{(l,n) \in g} \{g(l, n),\ B_i(u + l, v + n)\}, \tag{5}$$

in which u and v go through all elements in the binary sharpness map B_i. From G_i, a greedy algorithm is used for selecting the subset of the focal stack. The starting point of the algorithm is selected with $m = argmax_{i \in N} \sum_{u,v} G_i(u, v)$. The map G_m is considered the sharpest and contain most of the information.

Next, each of the remaining maps is combined in turn with the selected map. The map that adds most to the first map is chosen for further processing. The maps are combined with the max operation and compared with

$$sum_i = \sum_{u,v} \max_{u,v} \{G_m(u, v), G_i(u, v)\}, \tag{6}$$

The index, m', that maximizes sum is determined and map $G_{m'}$ is selected and combined with G_m to to form a new combined map so that new $G_m = \max_{u,v} \{G_m(u, v), G_{m'}(u, v)\}$. The algorithm iterates until at least 80% of the foreground region is covered by the combined maps or 5 maps have been selected.

$M < N$ images have now been selected from the focal stack for watermark extraction.

3.4. Reconfigure images

The full-sized images, I_j, $j = 1...M$ are filtered with a small Gaussian kernel, the coefficient of which depends on the image distance to the camera. This is done to smooth out artefacts that originate from the halftone printing process, as shown in Fig.4. For the mobile phone used in the experiments, the coefficient was selected experimentally to be 1/8 of the image distance to the lens in dioptres.

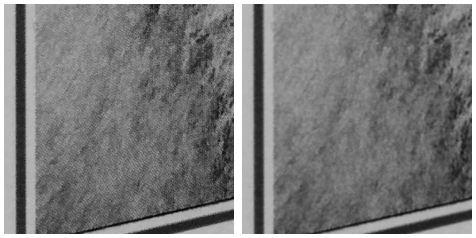


Figure 4: Smoothing effects of halftoning a) before blurring b) after blurring.

In order to speed up processing, the images are first scaled to 1/8th of the size and features are calculated from these scaled images $\hat{I}_j$, $j = 1...M$. Let us call this scale matrix H.

3.5. Image registration

As the camera may have moved during shots, the images need to be registered. The registration method should be robust to orientation changes and blurring, because images are captured at various depths. Additionally, the method should be fast to compute in order to run on a mobile device.

Accelerated-Kaze features (Alcantarilla et al., 2013) were selected for this research. The AKAZE features are blur-invariant and significantly faster to compute than, e.g. SIFT features (Lowe, 2004). The AKAZE features employ Fast Explicit Diffusion (FED) framework (Grewenig et al., 2010; Weickert et al.,

2016) and pyramidal approach to speed up non-linear scale space computations and enable robustness against blurring. Additionally, Modified-local Difference Binary (M-LDB) descriptors were used as recommended by the AKAZE developers (Alcantarilla et al., 2013).

The registration method requires few initial parameters. The parameters were chosen as 4 octaves, 2 sub-levels and 0.002 as the detector threshold. Fig. 5 shows an example of a sub-stack of three ($M = 3$) images which are registered.

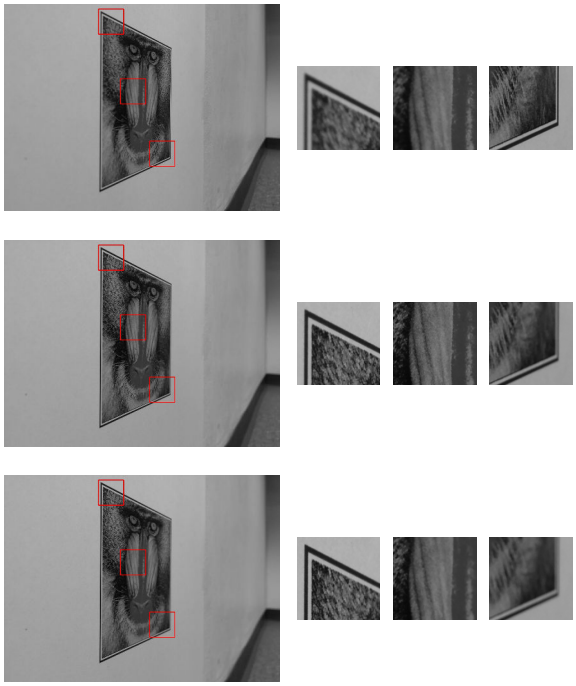


Figure 5: An example of a sub stack that has been registered with sharp regions on left, middle and right, respectively. Corresponding parts are magnified and placed next to each image.

The obtained features are then matched with a brute force matcher by selecting the closest matches with Hamming distance. In order to obtain the transformation matrices, the transformation between each image is calculated with the help of the RANSAC (Random Sample Consensus) feature selection

algorithm (Fischler and Bolles, 1981). Subsequently, registration matrix approximations $\hat{R}_j$, $j = 1...M$ are obtained.

3.6. Image blending

After registration, the images are blended in order to obtain the final all-in-focus image. Here, Laplacian pyramid blending (Burt and Adelson, 1983) with one level is applied with a sharpness mask as an alpha mask.

The aforementioned sharpness mask is built by first applying Laplacian filter to each registered image separately and taking the absolute value of the filtering as in Eq. 3. These values are then smoothed with a 5×5 averaging filter. For a number of M registered and blurred images $\hat{I}_j^r$, sharpness mask Q is calculated with

$$Q(x, y) = \underset{j \in M}{\operatorname{argmax}} \ \hat{I}_j^r(x, y), \tag{7}$$

in which $\hat{I}_j^r$ denotes jth registered image and the resulting mask Q has indices $1...M$ as its elements. Basically, the sharpness mask indicates which of the images is the sharpest at each pixel after noise is filtered out with the averaging filter. An example of the mask is shown in Fig. 6 when $M = 3$. An example of the final blended image is shown in Fig. 7.

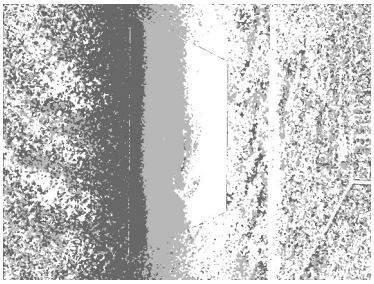


Figure 6: Sharpness mask in which the shade of each pixel indicates the index of the image which was sharpest at that pixel.

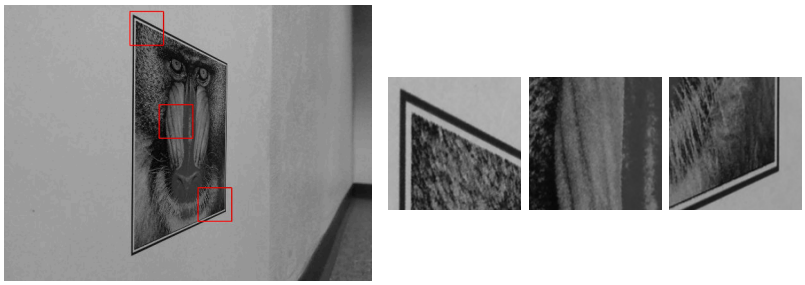


Figure 7: Final result after blending.

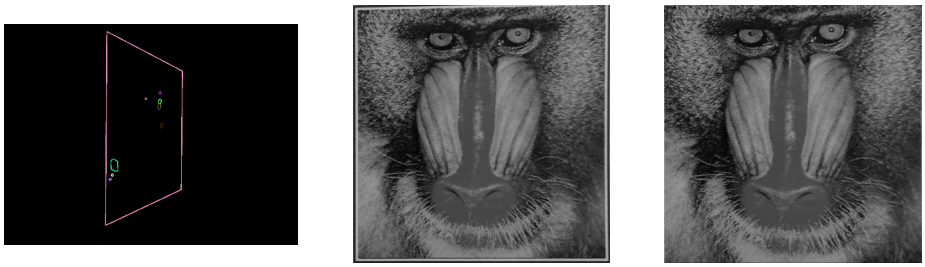
3.7. Rectifying the image and watermark synchronization

The next task is to locate the frame in the blended image. The frame is found by first thresholding the all-in-focus image with an adaptive thresholding method. The method calculates a threshold value t for each pixel by taking a weighted sum (cross-correlation with Gaussian window) of a neighbourhood. Here, an experimentally determined window size of 51×51 and standard deviation of 8 are used. Next, an elliptical structuring element of 11×11 is applied and morphological closing operation is applied to the image so that small details are removed.

Contours are found in the binary image with the algorithm by Suzuki and Abe (1985). At this point, the center-most contours are selected and those contours are ignored that have centers far off the image center. It can be assumed that the user has aimed the watermarked image towards the center of the view finder. The contours are simplified with Douglas-Peucker algorithm (Heckbert and Garland, 1997) and the largest one with four corners is found to be the frame. These image corners are then used in rectifying the image by assuming affine transformation, as shown in Fig. 8, and a matrix $\hat{T}$ for rectifying the image is obtained.

3.8. Watermark extraction

In order to preserve watermark robustness, the original images I_j are reloaded. Notably, only the selected images $j = 1...M$ and not the whole focal stack. For



a) b) c)

Figure 8: a) Searching the contours and b) an example of the rectified image with a frame and c) after the frame is removed.

watermark extraction, the obtained transformation matrices $\hat{T}$, $\hat{R}_j$ and H are combined and applied as in equation

$$p^* = \hat{T}\hat{R}_j H^{-1} p, \tag{8}$$

in which p denotes a pixel in the image I_j and p^* denotes a pixel in the image Y^*. Therefore, the images Y_j^*, $j = 1...M$ are obtained from which the watermark can be extracted.

The extraction method for one image ($j = 1$) is illustrated in Fig. 9. The method is robust to a moderate amount of error in block division. If $j > 1$, each image is divided blindly into $k = 16$ equal sized blocks, as in the embedding phase, and for each block k, the sharpest image block from j blocks is selected from which the watermark is extracted. The sharpness of a block is defined as an average of the absolute values of the Laplacians of the block.

Each block k is processed separately. The first block is considered as a synchronization block and is set to angle 0°. When the message is extracted, the angle of the fist block is used as a reference in correcting minor rotations of the image.

The blocks are each first Wiener filtered in order to decrease the impact of the cover image on the watermark. Each Wiener filtered block is then autocorrelated so that the periodic patterns embedded are revealed. In order to

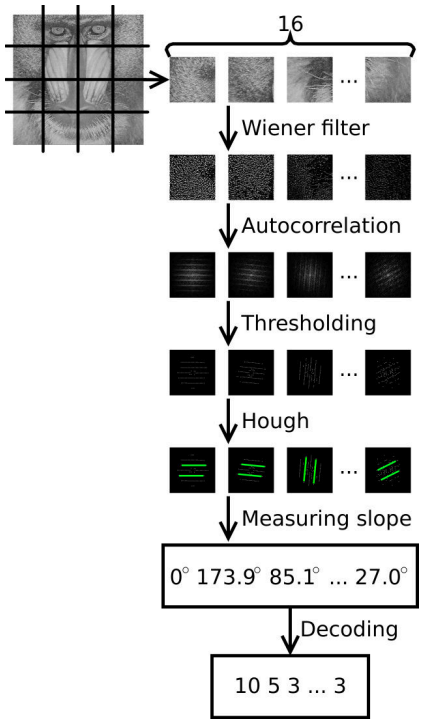


Figure 9: Overview of the method for watermark extraction.

determine the angle at which the patterns have been embedded, the autocorrelation results are then thresholded to form a binary grid that shows the peaks of the autocorrelation function. The thresholding limit is selected such that 99.7% is below the threshold and the center of the autocorrelation is masked out in computations as it contains mostly noise. Then Hough transform, that detects the direction as lines, is calculated. The slopes of the two longest parallel lines show the watermark message in angles that is further decoded into the original message by quantization.

4. Results

4.1. Watermark robustness

The tests were run on a laptop computer with a dual core 2.20GHz processor and 8GB of memory. The digital camera used for comparison tests was a Canon G7 with 10 Mpx resolution and Canon Hack Development Kit (CHDK) package installed in the camera. The Android mobile phone has Qualcomm Snapdragon 800 (2,26 GHz) processor and 2Gb of memory, as well as 8 Mpx camera. Both cameras stored the captured images in JPEG format although in the Android camera the JPEG quality was set to 100 in order to minimize the effet of JPEG compression.

The algorithms were implemented with C++ and Open Source Computer Vision Library (OpenCV). This enabled us to run exactly the same code on a computer and a mobile phone. On the computer, the code was run through QT (2016) interface. The user interface for the mobile phone was built with Android Java and connected with the C++ code with Native Development Kit (NDK).

For testing robustness at various angles, the cameras were placed on a rotating platform in turn. The cameras were then rotated around a printed and watermarked image that was placed on the wall. The robustness was tested at the angles of 0°, 20°, 40°, 50°, 60°and 70° by capturing an image at the angles and extracting the watermark. This was repeated with two different watermark strengths and three different distances. Examples of how the images look in each angle are illustrated in Fig. 10.

The six original images used for testing are illustrated in Fig. 11. All images were of the size of $512x512$ pixels, including three well known test images and three real world examples. Each image was watermarked by embedding a message of 60 bits with $(15, 8)$ Reed-Solomon error correction coding (Reed and Solomon, 1960). The robustness was tested with two watermarking strengths, $\delta_1 = 100$, $\delta_2 = 10$ and $\delta_1 = 120$, $\delta_2 = 12$ and the watermarked images were printed out with HP Color LaserJet 4650 printer.

Figure 10: Examples of captured images at a distance of 12.5 cm in a) 0°, b) 20°, c) 40°, d) 50°, e) 60° and f) 70°, respectively.

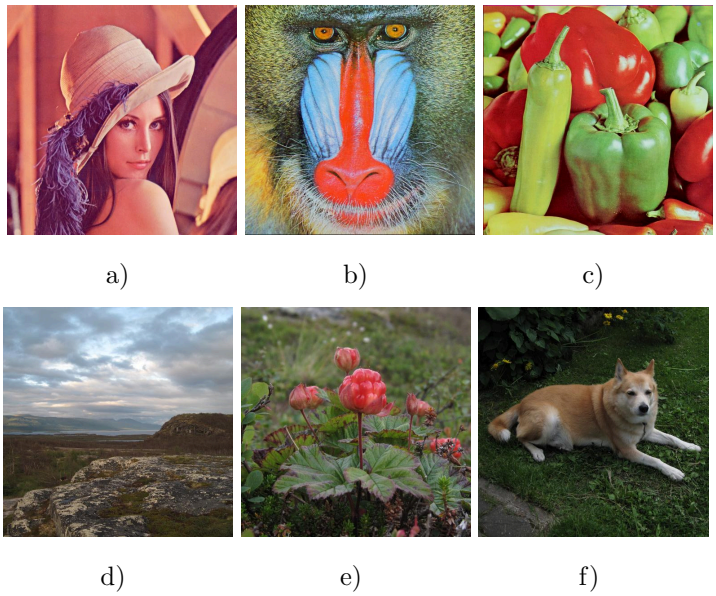


Figure 11: Test images a) Lena, b) Baboon, c) Peppers, d) Scene, e) Cloudberry and f) Dog, respectively.

For image quality analysis, we measure mean structural similarity (MSSIM) (Wang et al., 2004) with a method inspired by Eerola et al. (2009) and Eerola et al. (2010) and explained in detail in Pramila et al. (2017). The watermarked and printed images were scanned with 1200dpi and the half-toning pattern was descreened with Gaussian low pass filter. The six images were tested with two cut-off wavelengths, 0.1mm and 0.5mm, and the means of the obtained results are shown in Table 1.

Table 1: Image quality by MSSIM in total

Strength	$\delta_1 = 100$	$\delta_1 = 120$	$\delta_1 = 100$,	$\delta_1 = 120$
	$\delta_2 = 10$	$\delta_2 = 12$	$\delta_2 = 10$	$\delta_2 = 12$
Blur	0.1 mm	0.1 mm	0.5 mm	0.5 mm
MSSIM	0.91	0.90	0.95	0.95

The results are collected in Tables 2 and 3. The tests were conducted with both mobile phone camera and digital consumer camera. Each image was captured at each angle once and the combined results of the six images are shown in the tables.

Because of the differences in field of views, the methods were chosen to be tested on different distances on different cameras. The distances were selected such that the relative size of the watermarked image in each test case was the same. Therefore, the mobile phone camera was tested with distances of 12.5cm, 14.5cm and 16.5cm. The digital camera distances were selected to be 15.0cm, 17.5cm and 16.5cm, respectively.

In the first table, Table 2, the results for WCAM algorithm are shown with error correction coding. The results show that the method is rather robust even with the 60˚ of rotation and only with minor errors in the 70˚.

The next table, Table 3, has the same test but without error correction coding. In this, all the 60 bits were considered as message bits in contrast to 32 bits of message in the first table. It can be seen from the table that there

Table 2: Bit Error Rate (%) of WCAM algorithm with Error Correction Coding

(a) Phone camera

cm	$\delta_1 = 100$, $\delta_2 = 10$						$\delta_1 = 120$, $\delta_2 = 12$					
	0°	20°	40°	50°	60°	70°	0°	20°	40°	50°	60°	70°
12.5	0.0	0.0	0.0	0.0	0.0	3.6	0.0	0.0	0.0	0.0	0.0	1.6
14.5	0.0	0.0	0.0	0.0	0.0	4.2	0.0	0.0	0.0	0.0	0.0	1.0
16.5	0.0	0.0	0.0	0.0	0.0	7.3	0.0	0.0	0.0	0.0	0.0	7.8

(b) G7

cm	$\delta_1 = 100$, $\delta_2 = 10$						$\delta_1 = 120$, $\delta_2 = 12$					
	0°	20°	40°	50°	60°	70°	0°	20°	40°	50°	60°	70°
15.0	0.0	0.0	0.0	0.0	0.0	1.6	0.0	0.0	0.0	0.0	0.0	0.0
17.5	0.0	0.0	0.0	0.0	0.0	9.9	0.0	0.0	0.0	0.0	0.0	0.0
20.0	0.0	0.0	0.0	0.0	0.0	5.2	0.0	0.0	0.0	0.0	0.0	8.9

were few errors at the angles of 50° and below but those are fixed by the error correction algorithm.

Furthermore, image related performance differences are illustrated in Fig. 12. The image shows image specific values from Table 3a. That is, each data point in the images consists of the average BER of the specific image at the three distances. It is clear from the figure that there are differences between the images. The images "Peppers" and "Dog" seem to perform the worst, whereas "Cloudberry", "Lena" and "Abisko" perform the best.

4.2. User tests

To show that the method works also in more realistic settings, five volunteers were asked to take pictures of a watermarked image with a mobile phone with the developed application installed. The users were asked first to take a few

Table 3: Bit Error Rate (%) of WCAM algorithm without Error Correction Coding

(a) Phone camera

cm	$\delta_1 = 100$, $\delta_2 = 10$						$\delta_1 = 120$, $\delta_2 = 12$					
	0°	20°	40°	50°	60°	70°	0°	20°	40°	50°	60°	70°
12.5	0.0	0.0	0.0	0.0	1.9	10.6	0.0	0.0	0.0	0.0	0.3	5.6
14.5	0.0	0.0	0.0	0.0	2.2	13.3	0.0	0.0	0.0	0.0	0.8	5.6
16.5	0.0	0.0	0.2	0.6	5.0	16.9	0.0	0.0	0.0	0.0	1.9	15.6

(b) G7

cm	$\delta_1 = 100$, $\delta_2 = 10$						$\delta_1 = 120$, $\delta_2 = 12$					
	0°	20°	40°	50°	60°	70°	0°	20°	40°	50°	60°	70°
15.0	0.0	0.0	0.0	0.0	0.8	6.9	0.0	0.0	0.0	0.0	1.4	3.1
17.5	0.0	0.0	0.0	1.1	1.7	9.2	0.0	0.0	0.0	0.8	0.6	3.3
20.0	0.0	0.0	0.3	1.9	2.5	9.7	0.0	0.0	0.0	0.0	0.0	10.6

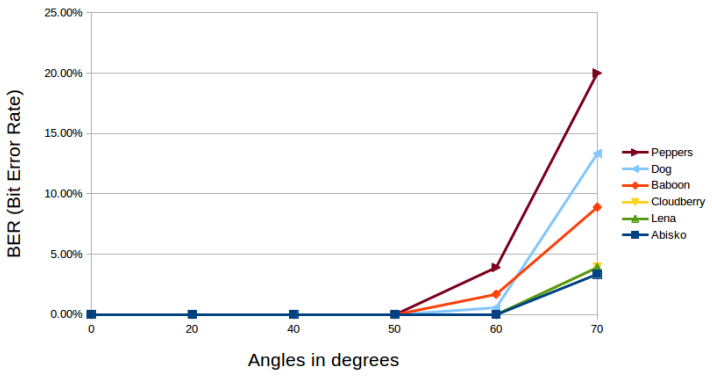


Figure 12: Comparison of image performance. (WCAM without ECC, camera phone images, distances combined).

images in order to familiarize themselves with the software, and then they were asked to test the limits of the application following the rules. These rules were:

1. The frame must be visible in the viewfinder.
2. The camera must be kept still. (The method is built to withstand naturally occurring hand movements but not if the camera is deliberately moved around.)
3. Distance should not exceed a predefined limit of approximately 30cm.

In the end, there were 24 sets of images, with average BER of 0.7%. When error correction was not used, the BER was 2.6%. The results are shown with more detail in Table 4. In the table, the only deviation seems to be on the last user. By looking at the images taken it can be seen that the images were taken at the further end of the predefined limit and there were some involuntary hand movements causing the errors reported. Some of the images taken by various users are collected in Fig.13.

Table 4: Bit Error Rate (%) of WCAM algorithm in user tests

User	Amount of images taken	with ECC	without ECC
1	4	0.0	1.7
2	3	0.0	0.6
3	7	0.0	1.1
4	7	0.0	0.8
5	3	5.2	8.9
Total	24	0.7	2.6

4.3. Performance of the algorithms

The performance of the algorithms was tested on the computer and the mobile phone and results were collected in Table 5. The algorithms were tested by selecting a sub-stack size of 1 or 3 images. The algorithm by Pramila et al. (2017) was re-implemented but tested only on the computer as it was deemed too slow for mobile phone use.

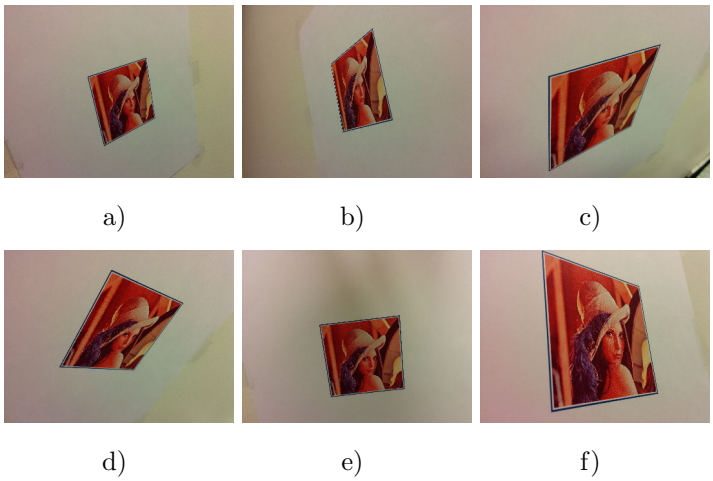


Figure 13: Examples of pictures the users took from which the watermark was read successfully.

Table 5: Performance of the algorithms

Sub-stack size	Pramila et al. (2017) on comp.		WCAM on comp.		WCAM on phone	
	1	3	1	3	1	3
handling images	0.10s	0.10s	0.10s	0.10s	0.37s	0.36s
sub-stack	0.01s	0.01s	0.01s	0.01s	0.07s	0.04s
reconfigure images	0.10s	0.25s	0.09s	0.26s	0.37s	1.23s
registering	-	4.19s	-	0.04s	-	0.30s
blending	-	1.17s	-	0.02s	-	0.14s
synchronization	0.29s	0.28s	0.01s	0.02s	0.23s	0.28s
extract watermark	0.37s	0.37s	0.37s	0.38s	2.63s	2.73s
Total	0.87s	6.37s	0.58s	0.83s	3.69s	5.08s

The results show that with the WCAM method, the registration is no longer an issue. The method by Pramila et al. (2017) was implemented for reference

because part of the speed gain occurred due to the update of the OpenCV library. AKAZE features have been included in the library thus affecting the implementation of the method. However, the WCAM algorithm is still multiple times faster. In WCAM method, most of the time is spent on extracting the watermark and memory accessing, as the 'reconfigure images'-stage includes reloading of the full sized images in memory.

5. Conclusion

Although it is often assumed in literature addressing print-cam robust methods that the user is able to take a picture perpendicularly to the watermark, this is rarely the case in real-life. The print containing the watermarked image, like in a poster, could be placed on suboptimal height or the image is placed in a magazine to be held on a table or in hand. In this paper, a method that is highly robust to geometrical distortions occurring when the user takes the picture in an angle is proposed. The method is based on optimizing a focal stack and building an all-in-focus image from the stack. In the process, the inherent features of the watermarking method are taken advantage of to speed up the processing times. The method is implemented on a mobile phone and proven fast and ready for real-life applications. Results are reported with full robustness to capturing angles of $<60^{\circ}$ and further confirmed with user tests with voluteers.

The method consists of two relatively separate parts, namely, the all-in-focus imaging method and the watermarking method. Thus, it would be possible to replace either part with some other existing method although the outcome could be seen as a new method and the research is thus left for future work. In addition, as a future work, it would be advantageous to study watermark extraction from curved surfaces. This would make the method even more robust for real life applications, like reading the watermark from a magazine with different layouts and also make it usable in a wider scope of applications.

References

Adams, A., Talvala, E.-V., Park, S. H., Jacobs, D. E., Ajdin, B., Gelfand, N., Dolson, J., Vaquero, D., Baek, J., Tico, M., Lensch, H. P. A., Matusik, W., Pulli, K., Horowitz, M., Levoy, M., 2010. The frankencamera: An experimental platform for computational photography. In: SIGGRAPH 2010. ACM, pp. 1–12.

Alcantarilla, P. F., Nuevo, J., Bartoli, A., 2013. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In: British Machine Vision Conf. (BMVC).

Burt, P. J., Adelson, E. H., 1983. The laplacian pyramid as a compact image code. IEEE Transactions on Communications 31 (4), 532–540.

Canon Hack Development Kit (CHDK), 2015. `http://chdk.wikia.com/wiki/CHDK`.

Chou, C.-H., Li, Y.-C., 1995. A perceptually tuned subband image coder based on the measure of just-noticeable-distortion profile. IEEE Transactions on Circuits and Systems for Video Technology 5 (6), 467–476.

Delgado-Guillen, L. A., Garcia-Hernandez, J. J., Torres-Huitzil, C., 2013. Digital watermarking of color images utilizing mobile platforms. In: 2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS). IEEE, pp. 1363–1366.

Digimarc, 2015. Discover application `https://www.digimarc.com/products/discover/mobile-app`.

Eerola, T., Kämäräinen, J.-K., Lensu, L., Kälviäinen, H., 2009. Framework for applying full reference digital image quality measures to printed images. In: 16th Scandinavian Conference on Image Analysis. SCIA '09. Springer-Verlag, pp. 99–108.

Eerola, T., Lensu, L., Kälviäinen, H., Kämäräinen, J.-K., Leisti, T., Nyman, G., Halonen, R., Oittinen, P., 2010. Full reference printed image quality: Mea-

surement framework and statistical evaluation. Journal of Imaging Science and Technology 54 (1), 1–13.

Fischler, M. A., Bolles, R. C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24 (6), 381–395.

Grewenig, S., Weickert, J., Bruhn, A., 2010. From box filtering to fast explicit diffusion. In: Joint Pattern Recognition Symposium. Springer, pp. 533–542.

Heckbert, P., Garland, M., 1997. Survey of polygonal surface simplification algorithms. Tech. rep., DTIC Document.

Katayama, A., Nakamura, T., Yamamuro, M., Sonehara, N., 2004. New high-speed frame detection method: Side trace algorithm (sta) for i-appli on cellular phones to detect watermarks. In: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia. ACM, pp. 109–116.

Kim, W.-G., Lee, S. H., Seo, Y.-S., 2006. Image fingerprinting scheme for print-and-capture model. In: Advances in Multimedia Information Processing-PCM 2006. Springer, pp. 106–113.

Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints. International journal of computer vision 60 (2), 91–110.

Native Development Kit (NDK), 2016. `https://developer.android.com/ndk/index.html`.

Open Source Computer Vision Library (OpenCV), 2015. `http://opencv.org/`.

Pramila, A., Keskinarkaus, A., Seppänen, T., 2007. Camera based watermark extraction-problems and examples. In: Proceedings of the Finnish Signal Processing Symposium.

Pramila, A., Keskinarkaus, A., Seppänen, T., 2008. Watermark robustness in the print-cam process. In: Proc. IASTED Signal Processing, Pattern Recognition, and Applications (SPPRA 2008). pp. 60–65.

Pramila, A., Keskinarkaus, A., Seppänen, T., 2012. Toward an interactive poster using digital watermarking and a mobile phone camera. Signal, Image and Video Processing 6 (2), 211–222.

Pramila, A., Keskinarkaus, A., Takala, V., Seppänen, T., 2017. Extracting watermarks from printouts captured with wide angles using computational photography. Multimedia Tools and Applications 76 (15), 16063–16084.

QT, 2016. `https://www.qt.io/`.

Reed, I. S., Solomon, G., 1960. Polynomial codes over certain finite fields. Journal of the society for industrial and applied mathematics 8 (2), 300–304.

Sakurikar, P., Narayanan, P., June 2014. Dense view interpolation on mobile devices using focal stacks. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on. pp. 138–143.

Solh, M., 2014. Real-time focal stack compositing for handheld mobile cameras. In: Proc. of IS&T/SPIE Electronic Imaging. International Society for Optics and Photonics, pp. 90200Z–90200Z.

Suzuki, S., Abe, K., 1985. Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing 30 (1), 32–46.

Vaquero, D., Gelfand, N., Tico, M., Pulli, K., Turk, M., Jan 2011. Generalized autofocus. In: 2011 IEEE Workshop on Applications of Computer Vision (WACV). pp. 511–518.

Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P., 2004. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 13 (4), 600–612.

Weickert, J., Grewenig, S., Schroers, C., Bruhn, A., 2016. Cyclic schemes for pde-based image analysis. International Journal of Computer Vision, 1–25.

Yamada, T., Kamitani, M., 2013. A method for detecting watermarks in print using smart phone: finding no mark. In: Proceedings of the 5th Workshop on Mobile Video. ACM, pp. 49–54.

Zhang, C., Bastian, J. W., Shen, C., van den Hengel, A., Shen, T., 2013. Extended depth-of-field via focus stacking and graph cuts. In: 2013 IEEE Conference on Image Processing (ICIP). pp. 1272–1276.