Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

Extracting watermarks from printouts captured with wide angles using computational photography

Anu Pramila · Anja Keskinarkaus · Valtteri Takala · Tapio Seppänen

Received: date / Accepted: date

Abstract Thus far the research of print-cam robust watermarking methods has focused on finding new methods for embedding and extracting the watermark. However, the capturing process itself, has been neglected in scientific research. In this paper, we propose a solution for the situation when the watermarked image has been captured in a wide angle and the depth of focus of the camera is not deep enough to capture the whole scene in-focus resulting in unfocused areas. The solution proposed here relies on a subfield of computational photography, namely all-in-focus imaging. All-in-focus images are generated by fusing multiple images from the same scene with different focus distances together, so that the object being photographed is fully in focus. Traditionally, the images to be fused are selected by hand from the focal stack or the whole stack is used for building the all-in-focus image. In mobile phone applications, computational resources are limited and using the full focal stack would result in long processing times and the manual selection of images would not be practical. In addition, we propose a method for optimizing the size of the focal stack and automatically selecting appropriate images for fusion. It is shown here that a watermark can still be recovered from the reconstructed all-in-focus image accurately.

A. Pramila · A. Keskinarkaus · V. Takala · T. Seppänen
Center for Machine Vision and Signal Analysis, University of Oulu, Finland
E-mail: anu.pramila@ee.oulu.fi

A. Keskinarkaus
E-mail: anja.keskinarkaus@ee.oulu.fi

V. Takala
E-mail: imv.takala@gmail.com

T. Seppänen
E-mail: tapio.seppanen@ee.oulu.fi

Keywords Print-cam · camera phone · digital watermarking · computational photography · all-in-focus imaging · focal stack optimization

1 Introduction

Digital image watermarking is the process of embedding information on a digital image in such a way that human eye cannot discern the differences but a computer can still read the embedded data. The applications range from carrying simple copyright information to authentication and value-adding services.

Digital watermarking is usually defined by three properties: robustness, imperceptibility and capacity. The final system will always be a trade-off of these three. Here our main focus is on robustness of watermarking and especially on print-cam robustness. In the print-cam process [18], the watermark is first embedded on the digital image, then the image is printed, and finally the watermark is extracted from the printed image with a digital camera or a camera phone. The process resembles that of 2D-barcodes but unlike barcodes, watermarks are not easily removed without destroying the work.

Fig. 1 shows an example scenario for the print-cam robust watermarking system. In the scenario, the user takes a picture of the watermarked image and the watermark application directs the user to the product website for reward. In this case, the user does not want to destroy the watermark intentionally as it is beneficial for him/her. However, the watermarked image is located on a magazine and which introduces challenges for the watermark application, such as geometric distortions and curved surface of the paper.

Fig. 1 A scenario for the proposed system. In this scenario, image is embedded to an image which is then used on a travel magazine. The user can then take a picture of the image and is directed to a website for more information.

The aim of this research is to focus on one aspect of print-cam robustness inherent to any kind of camera application, namely focusing. Until now, print-

cam robust watermarking research has been based on the assumption that the picture is taken as perpendicularly as possible to the watermarked image and that autofocus handles focusing [15, 28, 7]. This is not the case in general when people take images from papers laying on desks or from the side while standing by their friends. In real life, the user points the camera to the object without caring if the image is levelled straight, or if all the parts of the object are properly focused.

In this paper, we will propose a method that improves focusing and increases the watermark robustness by providing a sharper image. This is done by applying computational photography techniques and especially by building an all-in-focus image. Eventually, we hope to introduce our method for mobile devices but to test and evaluate the idea and prototyping the method we begin here with an ordinary camera and a computer.

Traditionally, to achieve an all-in-focus image, especially of objects close to the camera lens, the depth of field must be increased. This requires decreasing the size of aperture and subsequently more lighting that might be difficult to achieve, or longer exposure that might introduce distortions such as motion blur. In addition, the mobile devices on the market have often fixed aperture and changing the depth of field is not possible.

Another way of increasing the depth of field is to use computational methods to build a focal stack, a series of images focused at different depths, and combining the focal stack into a new image with an extended depth of field. The basic approach would be to capture the focal stack and fuse all the obtained images together. This works if the camera and scene are stationary. However, when the camera is held on hand, small shakes and movements of hand during the capturing process make image registration of the focal stack necessary [25, 29].

The reason for optimizing the focal stack, i.e., selecting only a part of the focal stack, for the final all-in-focus image composition is two folded. First, when the camera is held on hand, there will be movement between frames. The more there is movement between focal stack images, the more there is need for image registration, and the risk for incorrect image registration grows. Secondly, image registration is potentially computationally expensive operation, especially if no prior-knowledge is taken into account. The bigger number of images are combined, the more memory and resources are required, increasing processing time. In addition, any small indistinguishable errors in the fusion process affect the watermark end result and therefore, minimizing the number of images also preserves the watermark information.

To our knowledge, this is the first paper to study focusing phase on the print-cam process and the first paper to solve this issue with focal stack optimization and computational photography. We begin the research with a digital camera in order to simulate camera phone use cases and to build a working prototype. The full process is explained with more detail in Section 3 by first describing the focal stack optimization in Subsection 3.1 and then the watermarking process in Subsection 3.2. The results of the research are reported in Section 4.

2 Related work

There are several aspects that separate print-cam robust watermarking methods from general digital watermarks. The foundation is on print-scan robust watermarking methods that are build to be robust against printing and scanning of the watermarked image. This means that the message survives e.g. AD/DA transformations, rotation, scaling and translation. On the other hand, the print-cam robust watermarking methods should be robust to print-scan, rotations in three dimensions, some degree of human interaction, reflections, lighting variations and camera related distortions, such as barrel distortion and focusing problems. [18].

One of the first print-cam robust watermarking methods was proposed by Katayama et al. in [14]. They embedded a message as a periodic watermark pattern and installed a frame around the image for synchronization.

Kim et al. [15] had a different idea and used a pseudo-random vector for message embedding. This vector was embedded repeatedly, and later the message was detected with an autocorrelation function. However, their method required a tripod and some human interaction to minimize geometrical distortions. Pramila et al. [20] also employed autocorrelation but in a form of directed patterns. The message was encoded in the angles of these patterns. The method did not require a frame nor any specific size or shape of the image.

Yamada et al. [28] divided the image into regions and embedded the message into these regions with spread spectrum techniques. The locations of the watermark were then detected and the message was extracted only in case of positive detection. The method required the user to keep the camera as perpendicular as possible to the watermark for the extraction to be successful.

As aforementioned research illustrates, most of the research of print-cam robust methods is focused on embedding the message into spatial domain. Few have researched other domains. Delgado-Guillen et al. [6] utilized log-polar transformations whereas Pramila et al. [19] opted to divide the watermark into several domains using multiple watermarks.

Some commercial applications have also been presented. E.g. Digimarc [7] introduced an application called Discover to connect users through watermarking from magazine pages to Internet. The method requires the user to take the image as perpendicularly to the watermarked image as possible.

Thus there are a few viable print-cam robust watermarking methods but most of them have been tested in relatively constricted settings. Many of the methods assume that the image is taken as straight as possible which is not unreasonable if the user sees the watermark as a beneficial source of information worth retrieving. Unfortunately, it cannot always be assumed that the user is able to stand right in front of the image, or that the surface where the watermarked image is set on is flat. The curved surface of a magazine page or a bottle can be too demanding for the build-in autofocus of a camera or the properties of the lens.

Depth of focus can be drastically increased with computational photography. Several methods have been proposed for generating all-in-focus images.

However, few of these take into account movement between images that occurs when images are captured with free hand, and none are proposed for watermark applications. In watermarking, the main goal is to extract the watermark correctly instead of viewing an appealing image. In addition, many of the all-in-focus algorithms are developed for desktop computers and cameras with a tripod or other fixed settings, averting image registration. These two facts make these methods impractical for hand-held cameras.

Vaquero et al. [25] presented a method for Linux based mobile phones capable of taking advantage of FCam programmable-camera software stack [1]. They surmised that not all of the images in the focal stack are needed for the all-in-focus image. Thus, they created a method for focus sweep and selecting an optimal set of images to be taken according to small sharpness maps. The sharpness maps were provided by the FCam library.

Focus sweep and focusing the lens to specific focal points is not available on all cameras as noted by Sakurikar et al. [22]. On the other hand, most of the cameras on mobile phones have capabilities to direct the autofocus towards different locations on a scene. Their approach was to divide the view-finder into 16 blocks and the autofocus algorithm was pointed towards each block in order. Overly similar images were removed before registration.

Solh et al. [23] opted not to use the focal stack but only pre-determined focal distances of 0, 50 and 100. The frames taken in these locations were then aligned and fused.

The work by Zhang et al. [29] recognised the need for focal stack registration and utilised SIFT (Scale Invariant Feature Transform) [16] features. They also took advantage of the IMU (Inertial Measurement Unit) of the phone to gather information about movements of the phone during capture so that good images could be discerned more easily from the blurry ones and sharp focal stack collected. However, apart from removing the worst images, they did not optimize the focal stack size in any way.

In this paper, we combine two thus far separate concepts, print-cam robust watermarking and all-in-focus imaging, into a watermarking system that is robust to severe 3D rotations. We propose a method for selecting only part of the focal stack for building an all-in-focus image, out of which the watermark can be extracted. The method for focal stack optimization is inspired by the method by Vaquero et al. [25] and further developed with watermarking in mind.

3 Methodology

The overall process presented in this paper is shown in Fig. 2. This includes automatic selection of images from the focal stack for fusion and generating an all-in-focus image for extracting the watermark.

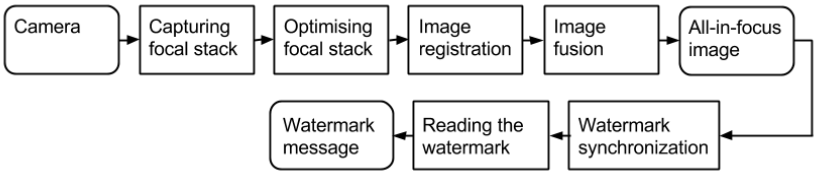

Fig. 2 Overall view of the all-in-focus image acquisition process

3.1 The method for focal stack optimization

In this section, the method for focal stack optimization is presented. It should be noted that the final all-in-focus image will not be presented to the user, nor the focal stack saved. In overall the method for focal stack optimization is developed for watermark robustness and efficiency.

3.1.1 Analysing focal stack

Let N be the number of images in the focal stack and S a sharpness map then S_i determines a sharpness map of i^{th} image in the stack, where $i = 1...N$. The sharpness maps can be considered as a coarse depth map of the scene. The sharpness maps are built by calculating the absolute value of Laplacian for each image in the focal stack and normalising the sharpness maps to values between 0 and 1 using the maximum and minimum value of the whole stack. In order to find gradual changes in the sharpness of the scene, the sharpness map size was experimentally selected to be 96 x 72.

The sharpness maps are then used for segmenting the image into foreground and background and further blurry and non-blurry regions at each focal distance. Foreground is determined as an area that appears sharp in at least one of the images in the focal stack. For the detected foreground area, sharp/blurry computation determines in which of the images the area is sharp.

The foreground/background segmentation is done by computing the standard deviation σ of each of the elements $S(u, v)$, where u and v go through all elements in S, and classifying each element as foreground if

$$\sigma > 0.1 \times \sigma_{max} \tag{1}$$

where the constant of 0.1 was experimentally determined to approximately discern watermarked image from the background. This results in a foreground/ background division map.

The foreground elements are then selected and the stack divided into sharp and blurry regions. First, at each foreground element, the image M that maximises the sharpness is selected and placed in a matrix. From the matrix, we can count the occurrences of each image M. The number of images, T, with

more than 50 occurrences is used as an estimate of how many images are needed to build an all-in-focus image, i.e., how many of the images in the focal stack contain valuable information about the scene. The number 50 was experimentally determined by observing a list of number of occurrences of all images. It seemed that there is an incline around 100 and so anything less than 50 can be considered noise.

This number, T, is used in thresholding the foreground elements into sharp and blurry regions. An element (u, v) at i^{th} map is determined as sharp if $S_M(u, v) == S_i(u, v)$, or $S_M(u, v) - S_i(u, v) < S_M \times T$ and the element at the neighbouring map is sharp. That is, element $S_{i-1}(u, v)$ or $S_{i+1}(u, v)$ is sharp.

3.1.2 Optimizing focal stack

In the next phase, the obtained binary sharpness maps are processed and the optimal stack composition is determined. The sharpness maps are now in binary form but in the real world sharpness varies between images more gradually. This is taken into account by using a Gaussian kernel g (see Eq. 2) that is applied to each map using a max function as is shown in Eq. 3.

$$g(j, k) = \frac{1}{2\pi\sigma_g^2} e^{-\frac{j^2-k^2}{2\sigma_g^2}} \tag{2}$$

$$G_{i_2}(u, v) = \max_{(j,k)\in g} \{g(j, k),\ B_{i_2}(u + j, v + k)\} \tag{3}$$

for each $B_i(u, v) = 1$, where u and v go through all pixels in the binary sharpness map B. B_i is the binary sharpness map of the i^{th} image in the stack and G_i is the final sharpness map after the Gaussian kernel application. In the experiments, a Gaussian kernel of size 3 x 3 ($j = -1; 0; 1$, $k = -1, 0, 1$), and standard deviation σ_g of 1 was experimentally found to provide enough accuracy.

After building sharpness maps and processing them with Eq. 3, a greedy algorithm is applied to select a subset of the focal stack. The process is illustrated in Fig. 3. First, for each map G_i, $i = 1...N$, all elements are summed together and the image with the maximum sum is selected to be included in the subset and as a starting point for the algorithm (corresponding map is highlighted with the blue colour in Fig. 3). This is done to determine which images of the stack contain most of the information. Then, each of the remaining sharpness maps are combined to the map of the selected image in turn and the maximum sum is computed. The image that adds the largest amount of sharp regions to the selected image is selected to be included in the subset (highlighted with red in Fig. 3). The combination of the images is then selected as the new starting point and the algorithm continues until a predetermined maximum number of images is selected or a threshold is reached. The threshold determines the total area of sharp regions on the combined sharpness map versus the amount of foreground regions. It was set to include at least 95% of the foreground regions in our experiments.

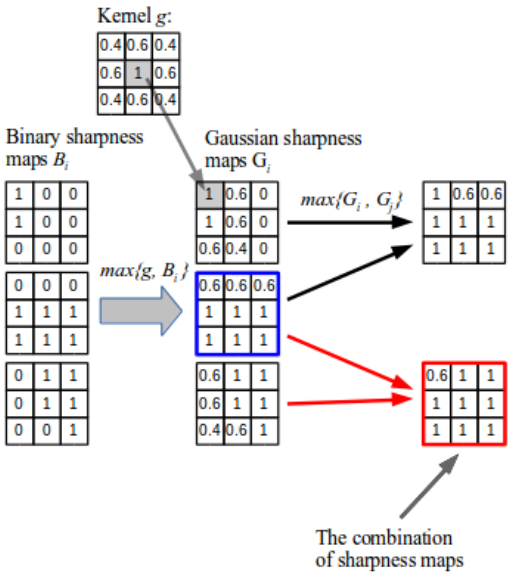

Fig. 3 An algorithm for selecting focal stack.

3.1.3 Image registration

The images in the selected sub-stack of the focal stack are then registered before blending. Fig. 4 shows an example of a selected and registered sub-stack and magnification images of some location on the images.

The registration method must be invariant to blur, scale and orientation changes, and relatively fast to compute. We decided to use Accelerated-KAZE features [2] that use Fast Explicit Diffusion (FED) framework [27,11] and pyramidal approach to speed up non-linear scale space computations. The features are blur invariant and significantly faster to compute than e.g. SIFT [16] features. Additionally Modified-Local Difference Binary (M-LDB) [2] descriptors were used as is recommended by Alcantarilla et al. [2]. The initial parameters for feature extraction were selected as 4 octaves, 2 sublevels and 0.002 as the detector threshold. Other parameters were the default parameters for the method. The obtained features were matched with a brute force matcher which selected the closest matches according to Hamming distance. The perspective transformation between two images was finally calculated with the help of RANSAC (Random Sample Consensus) feature selection algorithm [10].

3.1.4 Image blending

At this stage, the sub stack has been selected and its images have been registered. The last step is to blend them into one. For blending, we apply Laplacian pyramid blending [3] with one level and a sharpness mask as an alpha mask.

The sharpness mask is calculated by first applying Laplacian filter to each image and taking the absolute value of the filter response. Then a Gaussian

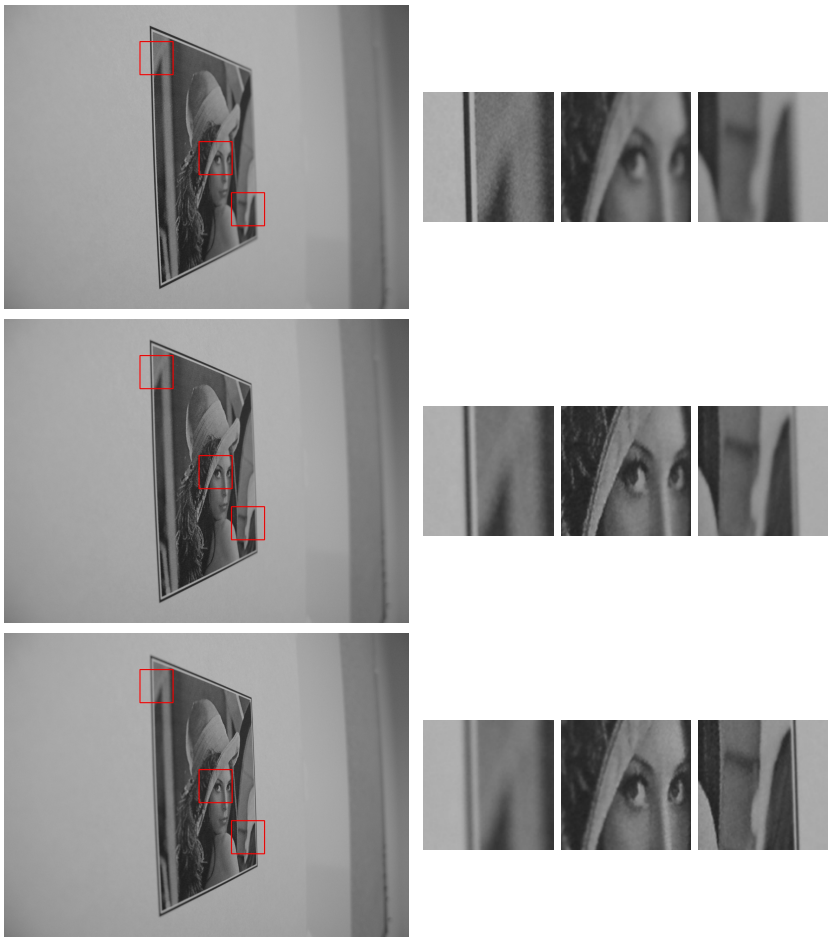

Fig. 4 An example of selected and registered sub-stack with sharp regions on left, middle and right respectively. Corresponding parts are magnified and placed next to each image.

blur is applied to the Laplacian values. Next, the images are compared pixel-wise and a maximum value for each pixel is determined. The masks are built in such a way that the pixel in the mask that corresponds to the maximum value is set to one. Fig. 5 shows the sharpness masks as indices to the corresponding image. The mask is calculated from a sub-stack shown in Fig. 4. Fig. 6 shows the final blended result.

3.2 Watermark embedding, synchronization and extracting

The watermarking method is briefly explained in the following subsections. The full process is illustrated in Fig. 7 and covered in detail in [20].

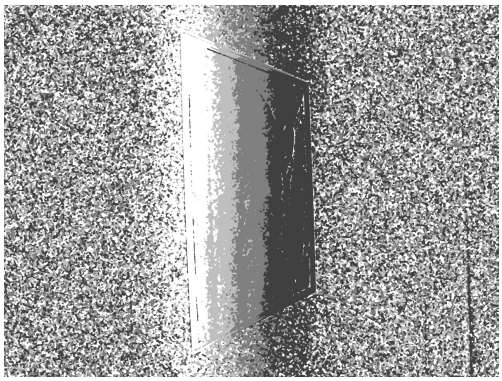

Fig. 5 Sharpness mask in which the shade of each pixel indicates the index of the image which was sharpest at that pixel.

Fig. 6 Final result after blending.

3.2.1 Embedding

The watermark is embedded on a computer to the digital image. A frame is placed around the image, and the watermarked and framed image is printed out.

Combining multiple images into one can induce small alignment errors and therefore it would be preferable to employ watermarking methods that can sustain small errors. In here we are using a modified version of the method by Pramila et al. [20] that relied on pseudo-random sequences and an autocorrelation function, thus being resilient to small errors that might occur in the alignment process.

The main difference between our method and the original is the increased capacity. The method begins by dividing the image into nine blocks and each block can hold four bits. Because one block is reserved for synchronization,

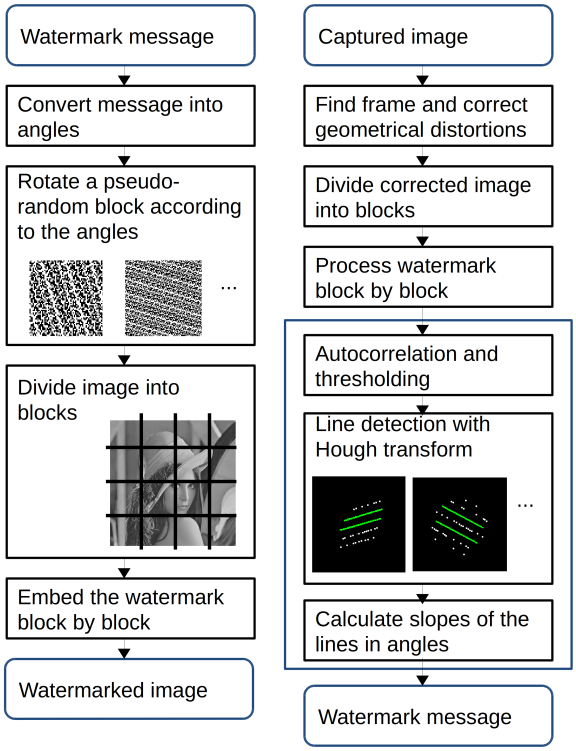

Fig. 7 Overall view of the watermark embedding and extracting process

this results in the capacity of 32 bits. Here we divide the image into 16 blocks which results in the capacity of 60 bits.

Increasing the amount of blocks not only boosts the capacity by nearly doubling it but potentially affects the robustness as well. As the number of blocks is increased, the block size is reduced. Thus the pseudo-random sequences are embedded in a smaller area which reduces the overall watermark strength per bit.

The usage of more blocks is justified here due to a different scenario. In this case, the image will be rotated heavily in 3D which results in the camera not being able to bring the whole image into focus at the same time. While multiple images are combined into one, there is always a risk that the images do not align perfectly. With smaller block size, the alignment errors to each watermark block can be decreased.

The watermark message is embedded in the luminance domain. The message itself is quantized into degrees. A pseudo-random sequence is multiplied into a 2D periodic pattern which is then rotated respectively to each of the message degrees. The patterns are embedded with

$$\begin{aligned} Y_i(x,y) = X_i(x,y) + \delta_1 JND(x,y) W_i^{\theta i}(x,y) \\ +\delta_2 (1 - JND(x,y)) W_i^{\theta i}(x,y) \end{aligned} \tag{4}$$

in which $JND(x, y)$ (Just Noticeable Difference) values are calculated with the method by Chou and Li [5]. The JND values regulate the embedding strength of the watermark by calculating how much each pixel can be changed before the change becomes noticeable. In the equation, Y_i is the ith block of the image, X_i is the preprocessed image, and the W_i^θ represents the coded watermark information in the form of directed periodic pattern and δ_1 and δ_2 are scaling factors for JND values. $JND(x, y)$ is the JND value in pixel (x, y).

a) b)

Fig. 8 Example of watermarked image with watermark strength of a) $\delta_1 = 100$ and $\delta_2 = 10$ b) $\delta_1 = 120$ and $\delta_2 = 12$. These images will be printed and the printing process will further smooth the images.

3.2.2 Watermark synchronization

Even though the watermarking method does not require a frame for the moderate rotations of the camera, we are here going to test the method with severe rotations in 3D with angles of $> 30°$ so that depth of focus comes into play. Therefore a frame was added to the method.

The frame also indicates to the user that a watermark is present, as they are invisible by default. Multiple value-adding applications require a way to tell the user that there is invisible content available although the artwork itself is not damaged.

The watermark extraction is begun by first taking a picture of the printout with a camera. After the image is captured, it is convolved with a small Gaussian kernel in order to smooth the distortions. Then the image is thresholded with an adaptive threshold and a black and white image is obtained. Finally, contours are found in the binary image with the algorithm by Suzuki and Abe [24]. These contours are then translated into polygons which are further processed in order to obtain simple representations of the curves in the image. This simplification is obtained with Douglas-Peucker algorithm [13]. This is done so that the rectangle of the frame is identified by searching through the

obtained polygons and finding the largest polygon with four corners. Four image corners can then be used in finding the projective transformation. Fig. 9 illustrates the image after the frame has been located and the image rectified.

a) b)

Fig. 9 Example of rectified image a) with a frame and b) after the frame is removed.

3.2.3 Extracting

The watermark is extracted by first dividing the image into 16 blocks. Each block is then processed separately and the direction of each pattern is determined. A block is filtered with a Wiener filter in order to reduce the effect of the original image from the watermark. Autocorrelation function is calculated and possible peaks are enhanced with the Laplacian of Gaussian filtering and morphological operations. The peaks are detected with

$$G^*(u,v) = \begin{cases} 1, \ when\ M(u,v) \times R^{**}_{\tilde{W}_i\tilde{W}_i} \geqslant \gamma \\ 0, \ when\ M(u,v) \times R^{**}_{\tilde{W}_i\tilde{W}_i} < \gamma \end{cases} \tag{5}$$

where $M(u,v)$ is a circular masking operator and R^{**} is the image after it is processed with a Gaussian operator. The masking operator masks the center area of the autocorrelation image which contains noise and can cause errors in line detection. The threshold γ is chosen so that approximately 99.7% of the data lies below the threshold.

The detected peaks are aligned according to the direction of the pseudo-random sequence pattern and this alignment is detected with Hough transform and line detection. These detected lines then give the angle of the pattern from which the message can be interpreted.

4 Experimental results

4.1 Test case

The focal stack for the experiments was captured by programming the camera to take pictures with a small focal step size from initial lens position. The camera was a Canon G7 digital camera with the Canon Hack Development Kit package [4] installed. The algorithms were implemented using C++ and OpenCV library [17] and run on a laptop computer with a dual core (four hyper-threads) 2.20GHz processor and 8GB of memory.

The method was tested by rotating the camera around a printed and watermarked image and capturing the focal stack at each angle. Examples of how the images look like in each angle are illustrated in Fig. 10.

Fig. 10 Examples of captured images at a) 0°, b) 20°, c) 40°, d) 50°, e) 60° and f) 70°

The reason for omitting the angles of 10° and 30° is that testing those angles would be redundant. Preliminary tests suggested that the method is robust to at least 40° and therefore angles less than 40° were tested more sparsely. Angles larger than 70° were omitted due to practical reasons: the camera was too close to the wall.

4.2 Performance evaluation

The performance of the algorithm was measured and example results were collected into Table 1. The tests were first performed with the image size of ten megapixels. However, the processing time for these images was long and therefore we also tested the method by first scaling the images to the quarter-

Table 1 Performance of the algorithm in seconds.

Resolution	10MP	10MP	2.5MP	2.5MP
Sub-stack size	1	3	1	3
Optimizing Sub-stack	1.2 s	1.2 s	1.2 s	1.2 s
Registering	-	28 s	-	6.9 s
Blending	-	1.7 s	-	0.4 s
Extracting the watermark	0.4 s	0.4 s	0.4 s	0.4 s
Total	2.1 s	32 s	2.1 s	9.2 s

size before registering. Registering the images was the most time consuming operation depending heavily on the image size and the size of the sub-stack.

4.3 Image quality analysis

Six color images were used for testing watermark robustness: three well known images and three real life examples. All images were the size of 512 x 512 pixels. The images are shown in Fig. 11.

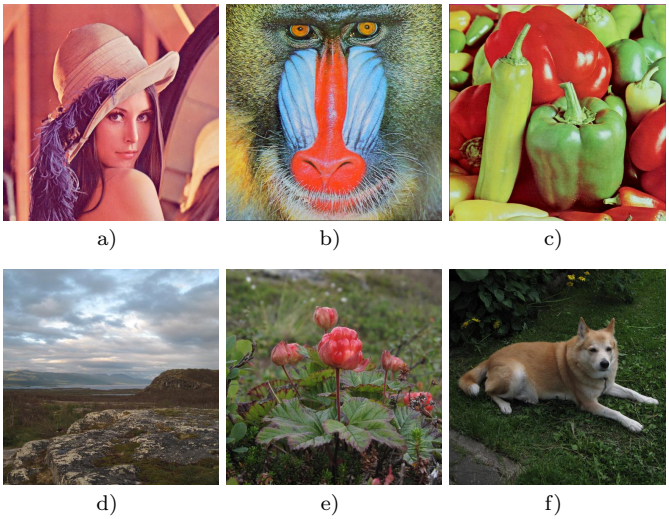

Fig. 11 Test images without watermarks a) Lena, b) Baboon, c) Peppers, d) Scene, e) Cloudberry and f) Dog

The message embedded was 60 bits and no error-correction coding was used. The watermark robustness was tested with two watermarking strengths and the watermarked images were printed on HP Color LaserJet 4650 printer. The watermarking strengths selected were $\delta_1 = 100$, $\delta_2 = 10$ and $\delta_1 = 120$,

Table 2 Image quality by MSSIM

Strength	$\delta_1 = 100$ $\delta_2 = 10$	$\delta_1 = 120$ $\delta_2 = 12$	$\delta_1 = 100$, $\delta_2 = 10$	$\delta_1 = 120$ $\delta_2 = 12$
Blur	0.1 mm	0.1 mm	0.5 mm	0.5 mm
Scene	0.83	0.83	0.93	0.93
Baboon	0.93	0.92	0.96	0.96
Cloudberry	0.88	0.87	0.93	0.94
Lena	0.97	0.96	0.98	0.98
Peppers	0.97	0.97	0.98	0.98
Dog	0.88	0.87	0.94	0.94

$\delta_2 = 12$. The printed image size was 6.5 cm x 6.5 cm. Because the printed image is significantly smaller on a paper than while viewed from a standard computer screen on which you can zoom, and because the printing process affects the watermark visibility, the watermark strength could be increased without compromising the image quality.

There exist several objective image quality metrics for evaluation of digital image quality. However, only few of them are suitable for evaluating quality of printed images, and most common ones, such as MSE (Mean Square Error) and PSNR (Peak Signal to Noise Ration), do not take into account human visual system [8]. Therefore, they do not describe well the visual quality of the watermarked images. Especially when the aim is to measure the quality of the printed images while the images are being scaled and moderated by the printing process [12].

Quality of printed images was measured with mean structural similarity (MSSIM) [26] with a method inspired by Eerola et al. [8] [9]. Three versions of each image were printed: an unwatermarked image, a watermarked image with $\delta_1 = 100$, $\delta_2 = 10$ and a watermarked image with $\delta_1 = 120$, $\delta_2 = 12$. The images were first scanned with 1200 dpi. As the images were printed with an electrophotographic printer, half-toning pattern is visible in the scanned images and therefore a descreening operation is required. This was achieved by blurring the images with a Gaussian low-pass filter. The cut-off wavelength of the filter should not exceed 0.5 mm [21], so that it does not destroy details that can be discerned with human eye. Therefore the image quality was tested with two cut-off wavelengths, 0.1 mm and 0.5 mm. The obtained results are contained in Table 2.

For better image quality, MSSIM should approach 1. The obtained MSSIM results show that the qualities of the watermarked images are sufficient for the intended use case.

4.4 Sub-stack size determination

The number of images selected for the sub-stack depends on the angle of the camera as illustrated in Fig. 12. The curves show a number of images selected when the camera is at distance of 15 cm and 20 cm. The full focal-stack contained approximately 15 images in each case.

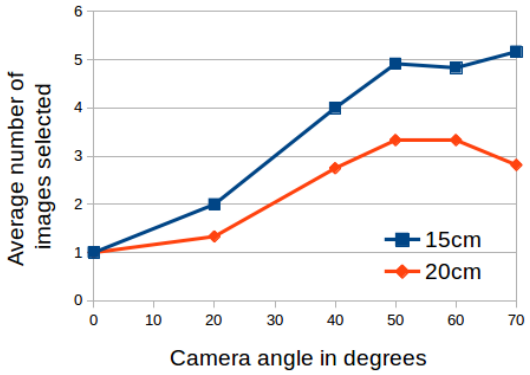

Fig. 12 Number of images selected per each testing angle.

When the picture is taken closer to the object, here at 15 cm, the depth of field is narrow and thus more images are needed for building an all-in-focus image. As the distance grows to 20 cm also the depth of field grows and less images are required.

The reason that the curves do not rise indefinitely but level or even turn down, is because of the selected sharpness map size. As the camera rotates, the watermarked image covers less of the sharpness map area. Because less area is used, the fewer images are selected that cover the determined foreground area. If the sharpness map size is increased so that even smaller details would be found, the more processing power would be required. This might also lead into imprecisions in sharpness map values as less data is available for computing sharpness values.

4.5 Watermark robustness

The watermark robustness was tested so that the printed and watermarked image was placed on the wall and the camera was turned around the image to different angles. At each angle, a focal stack was captured and the watermark extracted. The watermark was tested at the angles of 0°, 20°, 40°, 50°, 60° and 70°, in x and y directions (yaw and pitch).

The watermarking results are collected into the following tables that show the BER (Bit Error Rate) percentages for each watermarked image in various angles. Tables 3 to 6 show results for ten megapixel images whereas Tables 7 to 10 contain the results for 2.5 megapixel images.

The watermark robustness was also tested at two camera distances, 15 cm and 20 cm. Tables 3, 4, 7 and 8 contain the results for the distance of 15 cm and the rest for the distance of 20 cm.

The watermarking strength varied from $\delta_1 = 100$, $\delta_2 = 10$ in Tables 3, 5, 7 and 9 to $\delta_1 = 120$, $\delta_2 = 12$ in Tables 4, 6, 8 and 10.

Table 3 BER(%) at different yaw angles with distance of 15 cm, $\delta_1 = 100$, $\delta_2 = 10$, and picture resolution of 10 megapixels.

	0°	20°	40°	50°	60°	70°
Scene	0.0	1.7	5.0	6.7	6.7	5.0
Baboon	3.3	0.0	0.0	3.3	5.0	20.0
Cloudberry	0.0	0.0	0.0	0.0	0.0	6.7
Lena	0.0	0.0	0.0	0.0	0.0	13.3
Peppers	0.0	0.0	0.0	0.0	1.7	16.7
Dog	0.0	0.0	0.0	0.0	5.0	31.7
Total	0.6	0.3	0.8	1.7	3.0	14.7

Table 4 BER(%) at different yaw angles with distance of 15 cm, $\delta_1 = 120$, $\delta_2 = 12$, and picture resolution of 10 megapixels.

	0°	20°	40°	50°	60°	70°
Scene	0.0	0.0	1.7	5.0	5.0	11.7
Baboon	0.0	0.0	0.0	0.0	3.3	6.7
Cloudberry	0.0	0.0	0.0	0.0	0.0	0.0
Lena	0.0	0.0	0.0	0.0	0.0	1.7
Peppers	0.0	0.0	0.0	0.0	0.0	6.7
Dog	0.0	0.0	0.0	0.0	0.0	36.7
Total	0.0	0.0	0.3	0.8	1.4	10.6

Tables 11 and Tables 12 contain results for testing in y (pitch) direction for one image. This was done to illustrate that the method works correspondingly also on y direction.

The results show that the method is robust up to the angles of 50° and performs well with the angles of 60°. With appropriate error correction coding, the method has potential to be robust at even higher angles of rotation.

The tables 3 and 4 show the effect of different watermarking strengths. Unsurprisingly, the watermark is more robust when embedded more strongly. The difference is small, nevertheless, and only visible after 60°. It seems that picture 'Scene' performs the worst. This is most probably due to the bright areas of the sky, where watermark embedding strength is necessarily low so as not to be visible.

Tables 5 and 6 demonstrate the watermark robustness at slightly longer distance. When compared with Tables 3 and 4 we can see that the difference is marginal but existing for all the images at angles less or equal to 60°. However, the difference is more visible at angle of 70°.

The rest of the tables (7 - 10) illustrate the effect of scaling the images into quarter of their size. The same captured images were used for testing as in Tables 3 to 6 but the images were resized after the focal stack optimization phase. Therefore the results are directly comparable.

The obtained results in Tables 7 and 8 show that with the distance of 15 cm, scaling the image beforehand has no effect on the results. At some cases, the results seem even slightly improved. This is most probably due to the fact that images taken at distance of 15 cm from the printed image can capture also parts of halftone pattern of the printer. This pattern can disrupt the watermark extraction whereas in the scaled image the effect of the pattern is reduced.

Table 5 BER(%) at different yaw angles with distance of 20 cm, $\delta_1 = 100$, $\delta_2 = 10$, and picture resolution of 10 megapixels.

	0°	20°	40°	50°	60°	70°
Scene	1.7	10.0	8.3	6.7	6.7	23.3
Baboon	0.0	0.0	0.0	3.3	8.3	26.7
Cloudberry	0.0	0.0	0.0	0.0	0.0	18.3
Lena	0.0	0.0	0.0	0.0	3.3	25.0
Peppers	0.0	0.0	0.0	0.0	5.0	23.3
Dog	0.0	0.0	0.0	0.0	6.7	36.7
Total	0.3	1.7	1.4	1.7	5.0	25.8

Table 6 BER(%) at different yaw angles with distance of 20 cm, $\delta_1 = 120$, $\delta_2 = 12$, and picture resolution of 10 megapixels.

	0°	20°	40°	50°	60°	70°
Scene	0.0	1.7	0.0	10.0	6.7	13.3
Baboon	0.0	0.0	0.0	0.0	5.0	18.3
Cloudberry	0.0	0.0	0.0	0.0	0.0	5.0
Lena	0.0	0.0	0.0	0.0	0.0	3.3
Peppers	0.0	0.0	0.0	0.0	1.7	18.3
Dog	0.0	0.0	0.0	0.0	1.7	21.7
Total	0.0	0.3	0.0	1.7	2.5	13.3

Table 7 BER(%) at different yaw angles with distance of 15 cm, $\delta_1 = 100$, $\delta_2 = 10$, and picture resolution of 2.5 megapixels.

	0°	20°	40°	50°	60°	70°
Scene	0.0	1.7	10.0	6.7	6.7	6.7
Baboon	3.3	0.0	3.3	3.3	5.0	16.7
Cloudberry	0.0	0.0	0.0	0.0	0.0	6.7
Lena	0.0	0.0	0.0	0.0	0.0	6.7
Peppers	0.0	0.0	0.0	0.0	3.3	11.7
Dog	0.0	0.0	0.0	0.0	0.0	23.3
Total	0.6	0.3	2.2	1.7	2.5	12.5

Table 8 BER(%) at different yaw angles with distance of 15 cm, $\delta_1 = 120$, $\delta_2 = 12$, and picture resolution of 2.5 megapixels.

	0°	20°	40°	50°	60°	70°
Scene	0.0	0.0	1.7	6.7	1.7	10.0
Baboon	0.0	0.0	0.0	0.0	3.3	8.3
Cloudberry	0.0	0.0	0.0	0.0	0.0	0.0
Lena	0.0	0.0	0.0	0.0	0.0	5.0
Peppers	0.0	0.0	0.0	0.0	1.7	13.3
Dog	0.0	0.0	0.0	0.0	1.7	15.0
Total	0.0	0.0	0.3	1.1	1.4	8.6

At distance of 20 cm, the differences are bigger but still acceptable considering that the bit error rates are well within boundaries of simple error correction coding.

The previous tables have shown that the method works on multiple images on large angles. However, it can be questioned if it is enough to test only yaw rotations. In many real world applications, the watermarked image can be located in difficult height for the user. In Tables 11 and 12 it is shown that the method works similarly in this case. Only Lena image was tested for rotations in y (pitch) angles and the results correspond with the results in x (yaw) direction.

Table 9 BER(%) at different yaw angles with distance of 20 cm, $\delta_1 = 100$, $\delta_2 = 10$, and picture resolution of 2.5 megapixels.

	0°	20°	40°	50°	60°	70°
Scene	10.0	5.0	8.3	1.7	10.0	21.7
Baboon	0.0	0.0	0.0	3.3	15.0	31.7
Cloudberry	0.0	0.0	0.0	0.0	0.0	26.7
Lena	0.0	0.0	0.0	0.0	5.0	26.7
Peppers	0.0	0.0	0.0	0.0	8.3	28.3
Dog	0.0	0.0	0.0	0.0	6.7	36.7
Total	1.7	0.8	1.4	0.8	7.5	28.1

Table 10 BER(%) at different yaw angles with distance of 20 cm, $\delta_1 = 120$, $\delta_2 = 12$, and picture resolution of 2.5 megapixels.

	0°	20°	40°	50°	60°	70°
Scene	1.7	0.0	0.0	0.0	1.7	18.3
Baboon	0.0	0.0	0.0	3.3	6.7	31.7
Cloudberry	0.0	0.0	0.0	0.0	0.0	6.7
Lena	0.0	0.0	0.0	0.0	1.7	13.3
Peppers	0.0	0.0	0.0	1.7	5.0	20.0
Dog	0.0	0.0	0.0	0.0	1.7	21.7
Total	0.3	0.0	0.0	0.8	2.8	18.6

Table 11 BER(%) of the Lena image at different pitch angles with picture resolution of 10 megapixels.

	0°	20°	40°	50°	60°	70°
100 15cm	0.0	0.0	0.0	0.0	3.3	6.7
120 15cm	0.0	0.0	0.0	0.0	0.0	6.7
100 20cm	0.0	0.0	0.0	0.0	1.7	13.3
120 20cm	0.0	0.0	0.0	0.0	0.0	13.3

Table 12 BER(%) of the Lena image at different pitch angles with picture resolution of 2.5 megapixels.

	0°	20°	40°	50°	60°	70°
100 15cm	0.0	0.0	0.0	0.0	0.0	11.7
120 15cm	0.0	0.0	0.0	0.0	0.0	3.3
100 20cm	0.0	0.0	0.0	0.0	1.7	16.7
120 20cm	0.0	0.0	0.0	0.0	0.0	15.0

5 Conclusion

In this paper, we propose a scheme for recovering the watermark from high amounts of camera rotation in the print-cam process. The method is based on recovering a subset of a captured focal stack and composing an all-in-focus image out of which the watermark can be extracted.

The results show that even without error correction coding the method is able to extract the watermark with reasonable accuracy at as large as 60° of rotations around the optical axis of the camera.

In future work, we will address the issues that rose in the research. Most notably, a mobile phone implementation will be constructed and evaluated. Introducing an error correction coding would also increase robustness values. In addition, computational complexity could be reduced to make the method more viable for mobile platforms.

Acknowledgements This work was supported in part by Finnish Cultural Foundation.

References

1. Adams, A., Talvala, E.V., Park, S., Jacobs, D.E., Ajdin, B., Gelfald, N., Dolson, J., Vaquero, D., Baek, J., Tico, M., Lench, H.P.A., Matusik, W., Pulli, K., Horowitz, M., Levoy, M.: The frankencamera: an experimental platform for computational photography. In: ACM. Trans. Graph. (Proc. SIG-GRAPH) (2010)
2. Alcantarilla, P.F., Nuevo, J., Bartoli, A.: Fast explicit diffusion for accelerated features in nonlinear scale spaces. In: British Machine Vision Conf. (BMVC) (2013)
3. Burt, P.J., Adelson, E.H.: The laplacian pyramid as a compact image code. In: Communications, IEEE Transactions on, vol. 31, pp. 532–540 (1983). DOI 10.1109/TCOM.1983.1095851
4. (2015). http://chdk.wikia.com/wiki/CHDK
5. Chou, C.H., Li, Y.C.: A perceptually tuned subband image coder based on the measure of just-noticeable-distortion profile. Circuits and Systems for Video Technology, IEEE Transactions on **5**(6), 467–476 (1995)
6. Delgado-Guillen, L.A., Garcia-Hernandez, J.J., Torres-Huitzil, C.: Digital watermarking of color images utilizing mobile platforms. In: Circuits and Systems (MWSCAS), 2013 IEEE 56th International Midwest Symposium on, pp. 1363–1366. IEEE (2013)
7. (2015). https://www.digimarc.com/products/discover/mobile-app
8. Eerola, T., Kämäräinen, J.K., Lensu, L., Kälviäinen, H.: Framework for applying full reference digital image quality measures to printed images. In: Image Analysis, pp. 99–108. Springer (2009)
9. Eerola, T., Lensu, L., Kälviäinen, H., Kämäräinen, J.K., Leisti, T., Nyman, G., Halonen, R., Oittinen, P.: Full reference printed image quality: Measurement framework and statistical evaluation. Journal of Imaging Science and Technology **54**(1), 10,201–1 (2010)
10. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
11. Grewenig, S., Weickert, J., Bruhn, A.: From box filtering to fast explicit diffusion. In: Joint Pattern Recognition Symposium, pp. 533–542. Springer (2010)
12. He, D., Sun, Q.: A practical print-scan resilient watermarking scheme. In: Image processing, 2005. ICIP 2005. IEEE International Conference on, vol. 1, pp. I–257. IEEE (2005)
13. Heckbert, P., Garland, M.: Survey of polygonal surface simplification algorithms. Tech. rep., DTIC Document (1997)
14. Katayama, A., Nakamura, T., Yamamuro, M., Sonehara, N.: 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, pp. 109–116. ACM (2004)
15. Kim, W.G., Lee, S.H., Seo, Y.S.: Image fingerprinting scheme for print-and-capture model. In: Advances in Multimedia Information Processing-PCM 2006, pp. 106–113. Springer (2006)
16. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**(2), 91–110 (2004)
17. (2015). http://opencv.org/
18. Pramila, A., Keskinarkaus, A., Seppänen, T.: Camera based watermark extraction-problems and examples. In: Proceedings of the Finnish Signal Processing Symposium (2007)
19. Pramila, A., Keskinarkaus, A., Seppänen, T.: Watermark robustness in the print-cam process. In: Proc. IASTED Signal Processing, Pattern Recognition, and Applications (SPPRA 2008), pp. 60–65 (2008)
20. Pramila, A., Keskinarkaus, A., Seppänen, T.: Toward an interactive poster using digital watermarking and a mobile phone camera. Signal, Image and Video Processing **6**(2), 211–222 (2012)

21. Sadovnikov, A., Salmela, P., Lensu, L., Kämärainen, J.K., Kälviäinen, H.: Mottling assessment of solid printed areas and its correlation to perceived uniformity. In: Image Analysis, pp. 409–418. Springer (2005)
22. Sakurikar, P., Narayanan, P.: Dense view interpolation on mobile devices using focal stacks. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on, pp. 138–143 (2014)
23. Solh, M.: Real-time focal stack compositing for handheld mobile cameras. In: Proc. of IS&T/SPIE Electronic Imaging, pp. 90,200Z–90,200Z. International Society for Optics and Photonics (2014)
24. Suzuki, S., Abe, K.: Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing **30**(1), 32–46 (1985)
25. Vaquero, D., Gelfand, N., Tico, M., Pulli, K., Turk, M.: Generalized autofocus. In: Applications of Computer Vision (WACV), 2011 IEEE Workshop on, pp. 511–518 (2011)
26. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. Image Processing, IEEE Transactions on **13**(4), 600–612 (2004)
27. Weickert, J., Grewenig, S., Schroers, C., Bruhn, A.: Cyclic schemes for pde-based image analysis. International Journal of Computer Vision pp. 1–25 (2016)
28. Yamada, T., Kamitani, M.: A method for detecting watermarks in print using smart phone: finding no mark. In: Proceedings of the 5th Workshop on Mobile Video, pp. 49–54. ACM (2013)
29. Zhang, C., Bastian, J.W., Shen, C., van den Hengel, A., Shen, T.: Extended depth-of-field via focus stacking and graph cuts. In: Image Processing (ICIP), 2013 IEEE Conference on, pp. 1272–1276 (2013)