BlurTags: spatially varying PSF estimation with out-of-focus patterns

Alexander Reuter
Goethe University Frankfurt, Germany
reuter@vsi.cs.uni-frankfurt.de

Hans-Peter Seidel
MPI Informatik, Germany
hpseidel@mpi-inf.mpg.de

Ivo Ihrke
Saarland University/ MPI Informatik, Germany
ihrke@mmci.uni-saarland.de

ABSTRACT

Current research is targeting the estimation and correction of lens imperfections. often modeled as a set of spatially varying point spread functions (PSFs). One way to measure these PSFs is their calibration with checkerboard patterns. Previous work, however, does not fully exploit all benefits of using a checkerboard. In particular, we show in this paper that the pose of the checkerboard with respect to the camera can be exploited to yield information on the circle of confusion, and thus the image blur of an ideal camera. By removing this expected blur, we can estimate residual PSFs that are due to the deviation of the optical system from a thin-lens model. The residual PSFs can then be used to sharpen images at comparable lens settings.
Practical side effects of our method are the design of a self-identifying pattern that can be robustly detected even in the case of image blur, and a corresponding algorithm for its detection.

Keywords

spatially varying PSF, self-identifying markers, deconvolution

1 INTRODUCTION

2012June 25 – June 28

The optical resolution of camera images is often limited by the quality of the lens system used for taking the picture. Even expensive lens systems show aberrations, especially at high aperture settings (low f-number). Alternatively, cheap plastic lenses are mass-produced and show less then ideal imaging capabilities. The deviation from perfect imaging is usually expressed by the system's point spread function (PSF). Though it is often assumed to be invariant with respect to shifts on the sensor and the settings of the camera, this is in general an invalid assumption. The space of PSFs is high-dimensional, varying with position on the sensor, aperture, focus and, potentially, zoom-setting of the camera. Knowing the distribution of the PSFs enables the computational removal of aberrations from the image via deconvolution.

It is therefore important to calibrate the PSFs, ideally for all possible combinations of camera settings. Using the EXIF information of an image taken with such a calibrated lens, it can be sharpened in a computational post-process. Recently, the work of Kee et al. [KPCW11] has introduced a method for estimating the PSF variation for aperture and zoom changes. However, they do not address the focus-setting of the camera. This leads to the problem that the calibration pattern has to be kept in-focus during the calibration process and thus, has to be moved.

In our approach, we take a different route. We exploit the planarity of the checkerboard to estimate its pose with respect to the camera. This, in turn, allows for estimating the circle of confusion, the PSF of an ideal thin lens. The observed calibration pattern is convolved with a different PSF, which we consider to be a combination of the ideal PSF and the in-focus PSF due to lens aberrations. Our goal is the estimation and removal of the latter. The resulting images are approximations to images that would be taken by a perfect lens system with a finite aperture.

Our algorithm relies on the ability to detect blurred patterns. For this reason, we design a checkerboard pattern that is well suited to estimate PSFs and which can be robustly detected in the presence of blur. We call the resulting pattern and detection routines *BlurTag*.

2 RELATED WORK

The correction of image blur is content of much research today. Usually, the goal is to correct for scene related degradations like motion blur [SJA08], out-of-focus blur [VBC+02], or camera shake [FSH+06]. Blur removal is a deconvolution problem which comes

in two flavors: blind deconvolution e.g. [AD88, Car01] and deconvolution based on a known kernel e.g. [Ric72, Wie49]. Joshi et al. [JSK08] determine the kernel by edge prediction using the rationale that part of the blur kernel can be observed in a direction orthogonal to the edge. Based on this approach, Kee et al. [KPCW11] propose a non-blind approach to estimate and remove lens aberrations. They employ checkerboard-like patterns including circles that provide edges in all orientations to estimate the kernel. As mentioned earlier, the pattern has to be in focus for the method to yield unbiased estimates. Point spread functions may change when the focus changes. If the calibration image has been taken with a different focus setting from the one that is to be corrected the results will be inaccurate. Our work aims at relaxing this requirement.

Our approach is based on pose estimation with respect to a planar target, a standard problem in computer vision [Zha00]. An implementation is available in Bouguet's calibration toolbox [Bou02]. The toolbox estimates the common parameters related to a pinhole camera such as focal length, camera orientation, as well as radial distortion. In this area, it is common to use uncoded checkerboards where the user has to define the outer checkerboard corners manually.

In the field of augmented reality, tag-based systems such as [GGSC96, ZFN02, dIMH02] are often employed. These methods address the automated pose estimation problem, but do not consider the aspect of calibration. The focus is on the automatic detection and identification of markers. In general, the tags of augmented reality can be adapted to checkerboards by using them as codes. The approach of Atcheson et al. [AHH10] places two-dimensional codes into the squares of the checkerboard. To enhance the detection quality in blurry regions, the authors introduce a new design where codes are rotated and separated from the feature points. Due to this, the resolution of the codes is reduced. In addition, the pattern employs straight edges which are not suitable to sample general two-dimensional blur kernels densely. We therefore opt for an alternative design.

Our goal is the automatic estimation of spatially varying PSFs which are distributed over a captured image. We capture an image which provides a checkerboard pattern. This checkerboard pattern is optimized for

1. automatic detection as in the CALTag implementation [AHH10],

2. accurate PSF estimation as in [KPCW11], and

3. blur robust detection for out-of-focus checkerboards.

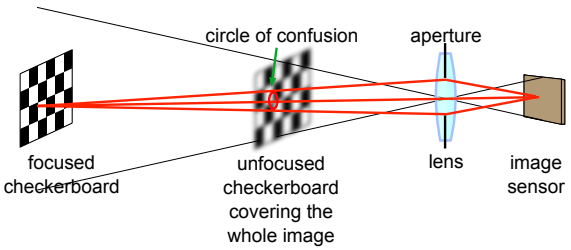


Figure 1: In our setup a calibration pattern is imaged at an out-of-focus position. The circle of confusion is blurring our image in addition to the lens PSF at the in-focus setting.

Further, we want to estimate the pose of the checkerboard. Knowing the distance of the focal plane as well as the aperture and focal length, we are able to estimate the circle of confusion of the image regions covered by the checkerboard pattern. These circles of confusion offer the ability to compute in-focus PSFs from out-of-focus PSFs.

In summary, we combine and extend different aspects of previous work in order to provide a practical solution to spatially varying PSF estimation for lens-induced aberrations.

3 OVERVIEW

The goal of our approach is the calibration of the spatially varying PSFs for different focus settings of an optical system using a static calibration pattern at a fixed position. The main motivation for this setting is that it is difficult to move a single calibration pattern to different focal planes while maintaining a full coverage of the image and a comparable apparent resolution of the target at different distances without changing the pattern on the target.

We therefore choose to image the calibration pattern in an out-of-focus position, see Fig. 1. Consider first the checkerboard to be placed in the focal plane. In this case, an ideal, finite aperture, thin-lens imaging system would produce a perfect image since the system response of the thin lens is Dirac. In a real system, however, the system response is different and the observed image I_{obs}^{IF} is given by the object space pattern I_{obj}, convolved with the system PSF p:

$$I_{obs}^{IF} = I_{obj} \otimes p, \tag{1}$$

where IF stands for "in-focus". In general, the convolution has a spatially varying kernel $p(x,y)$. If the pattern is placed in a different position, a finite aperture, thin-lens system will produce out-of-focus blur which is usually described by the circle of confusion, a special PSF which is a scaled version of the aperture shape. The ideal image I_{obs}^{TLOF} under these circumstances is given by

$$I_{obs}^{TLOF} = I_{obj} \otimes p_{coc}. \tag{2}$$

Here, p_{coc} is the circle of confusion PSF and TLOF stands for "thin-lens out of focus". Again, as in the in-focus case, a real system does not produce I_{obs}^{TLOF} since its PSF is not an ideal circle of confusion which would have a pill-box shape (assuming a circular aperture). Instead, the out-of-focus PSF of the real system is given by $p_{coc} \otimes p$. The observed out-of-focus image I_{obs}^{OF} in a real system is thus given by

$$I_{obs}^{OF} = I_{obj} \otimes p_{coc} \otimes p. \quad (3)$$

Our method aims at estimating p, given the observed out-of-focus image I_{obs}^{OF} and the pattern definition I_{obj}. In order to make this problem tractable we need a reliable way of estimating p_{coc}. We also aim at relaxing the condition that the pattern has to be placed parallel to the image plane.

The above requirements can be fulfilled if the pose of a (planar) target with respect to the camera can be estimated from the calibration image I_{obs}^{OF}. Knowing the pose, and the aperture and focus settings of the camera, $p_{coc}(x,y)$ can be computed at every position in the image. This implies, that we can recover the PSF of the in-focus plane $p(x,y)$ by inverting Eq. 3.

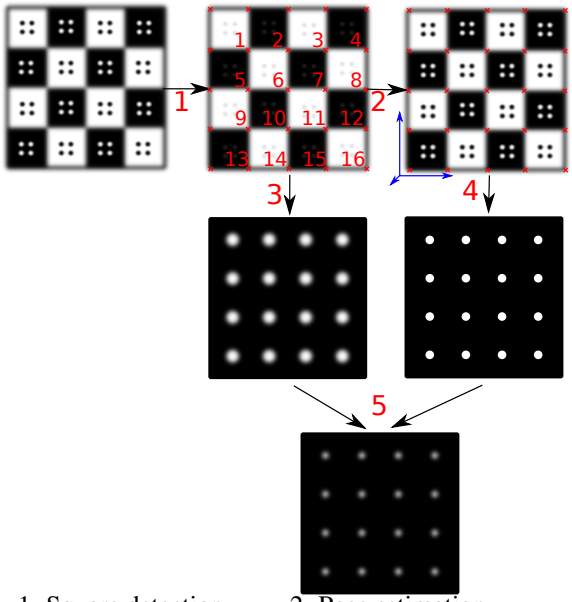
In practice, we compute the orientation of the planar target by tracking a checkerboard coded with self-identifying markers (tags). This automates the calibration process which is important if several focus settings are to be processed, potentially for several aperture/zoom settings of the camera. In this article, we only discuss the computation of the PSFs from a single image. Calibrating an arrangement of settings would apply our method to each image independently.

The processing pipeline is shown in Fig. 2. First, we detect the individual squares of the checkerboard pattern and verify the contained tags. Then we estimate the PSFs, which are blurred by the circle of confusion. In parallel, we estimate the pose of the checkerboard, which can also be regarded as the extrinsic parameters of the camera with respect to the checkerboard. We can extract the focal plane parameters from the EXIF header provided by the captured image. The combination of pose information and focal plane parameters allows us to compute the circle of confusion for each position covered by the checkerboard pattern. Finally, we deconvolve each PSF with the related circle of confusion. The result is a full image covering set of PSFs for an image in the focal plane of the camera.

In order for this process to work well, we have to regard two aspects: First, a checkerboard design that is well suited to PSF estimation. Second, a detection algorithm that is robust to blur.

4 CHECKERBOARD DESIGN

Estimating the PSF distribution based on a captured checkerboard pattern requires its initial detection. In our approach, the checkerboard detection fulfills two goals at once: First, we want to estimate one PSF for each checkerboard square, and second, we want to estimate the pose of the checkerboard for estimating the circle of confusion. Based on these two goals, we can derive criteria, which have to be met by our checkerboard design:



1. Square detection
2. Pose estimation
3. PSF estimation
4. CoC estimation
5. PSF sharpening

Figure 2: Overview of our approach

1. We seek to estimate accurate PSFs,
2. the detection must be robust to blur, and
3. the process should be automatic.

Based on these criteria, we derive a checkerboard pattern (Fig. 5, upper left) providing a code for each checkerboard square, for an example see Fig. 3.

For our initial tests we use a simple binary code. This code could be made error-correcting as e.g. in [AHH10] by sacrificing some bits. The codes, however, are not only used to identify a marker, but also to serve as a pattern for PSF estimation, i.e. I_{obj} in Eq. 3. It has been observed in previous work that it is important to use patterns that provide edges in every direction [JSK08]. We therefore choose circles as the basic elements of our codes. Instead of coding a bit as circle present (1) vs. circle absent (0) in a particular position, we use an inner-/outer-circle hierarchy, where the presence of an inner circle indicates a logical 'one', see Fig. 3. This measure ensures the presence of edges even for codes that contain many zeros and thus improves the stability of the PSF estimation.

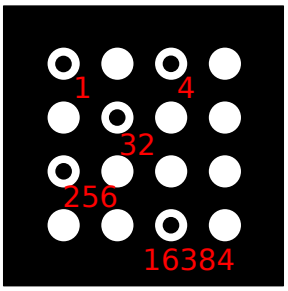


Figure 3: Sample code with active bits 2^0, 2^2, 2^5, 2^8 and 2^{14} leading to the ID 16677. The bits are using a lexicographical order. Codes are unique under rotation and mirroring.

Further, we decided to make the codes unique in terms of rotation and mirroring by removing symmetric ones from the list of available codes. This property is simplifying the detection process. Knowing the orientation helps when assembling a complete checkerboard pattern from single detected squares. We are dealing with two dimensional codes providing w horizontal and h vertical code elements. For two reasons, we decided to set $w = h = 4$. First, we are dealing with squares so we set $w = h$ to distribute the corresponding objects homogeneously. Second, regarding uniqueness with respect to symmetries and robustness to errors, 3×3 codes offer too few candidates to fill a reasonably sized checkerboard pattern with unique codes. On the other hand, 5×5 codes become too dense for small resolutions in the captured image.

5 IMPLEMENTATION

The implementation of our algorithm is split into several parts. At first, we detect the checkerboard and identify the markers, Sect. 5.1. The important aspect in this step is the robustness of the detection routine to blur. Next, we compute the combined PSFs $p_{coc} \otimes p$ from the image and the pre-defined pattern, Sect. 5.2. In order to remove the influence of the circle of confusion p_{coc}, we compute the pose of the checkerboard and estimate p_{coc}, Sect. 5.3. Finally, we compute the in-focus PSFs p by deconvolving the combined PSFs $p_{coc} \otimes p$ with the circle of confusion PSF p_{coc}, Sect. 5.4.

5.1 Detection process

The critical aspect in the detection process is the identification of the individual checkerboard squares in the presence of blur. The main difficulty is the apparent merging of neighboring regions of the pattern [AHH10]. The main differentiating aspect of our detection scheme is that we interpret the thresholded image as a hierarchical tree structure: a connected component is a child of another if it is completely included in the other. This feature lets us remove parts of the image (the codes) for the early parts of the detection routine. This way, we can focus on separating the checkerboard squares without fearing to damage the codes inside, see Fig. 4.

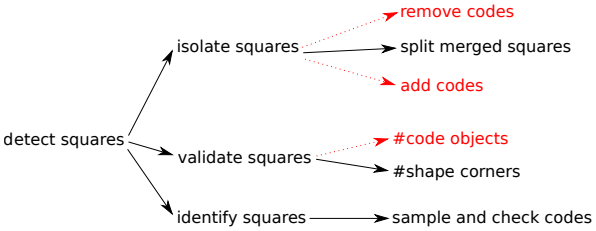


Figure 4: Overview of the detection process.

An overview of the processing pipeline is shown in Fig. 5. The number of the processing step refers to the number of the subfigures in the pipeline overview. In the following, we describe the individual steps in more detail.

1. Thresholding the image

Checkerboard patterns are usually black and white to simplify the detection process. During capture, ambient light, shades and light gradients lead to a gray-scale checkerboard pattern. Our first step is to binarize the image via thresholding. We use a median based adaptive thresholding algorithm [Jai89]. The definition of the window size is the most critical parameter within the process. Too small windows lead to noisy results while too large windows may lead to shrunk, open, or vanishing circles. In our implementation, we use a window width of about 4 to 5 percent of the image width. After that we remove noise from the thresholded image.

2. Removing circles

In this step, we remove the circles that represent the code of a square from the binary image. The result is a binary image without code structures as in Fig 5 (2). This leads to significant improvements in the following *splitting* step because the fine detail of the code structures can be ignored. As discussed above, we interpret the thresholded image as a tree of connected components, Fig. 6. The tree construction is performed in two steps. First, the binary image is decomposed into connected components. Second, the connected components are organized in the component tree [NC06]. The component tree encodes the complete interrelations between the nested structures. Using it, we can remove the circles by removing the corresponding nodes. In our pattern, we remove the nodes which provide exactly one leaf and the leafs themselves. After that, we draw the tree again as a thresholded image, this time without the circles.

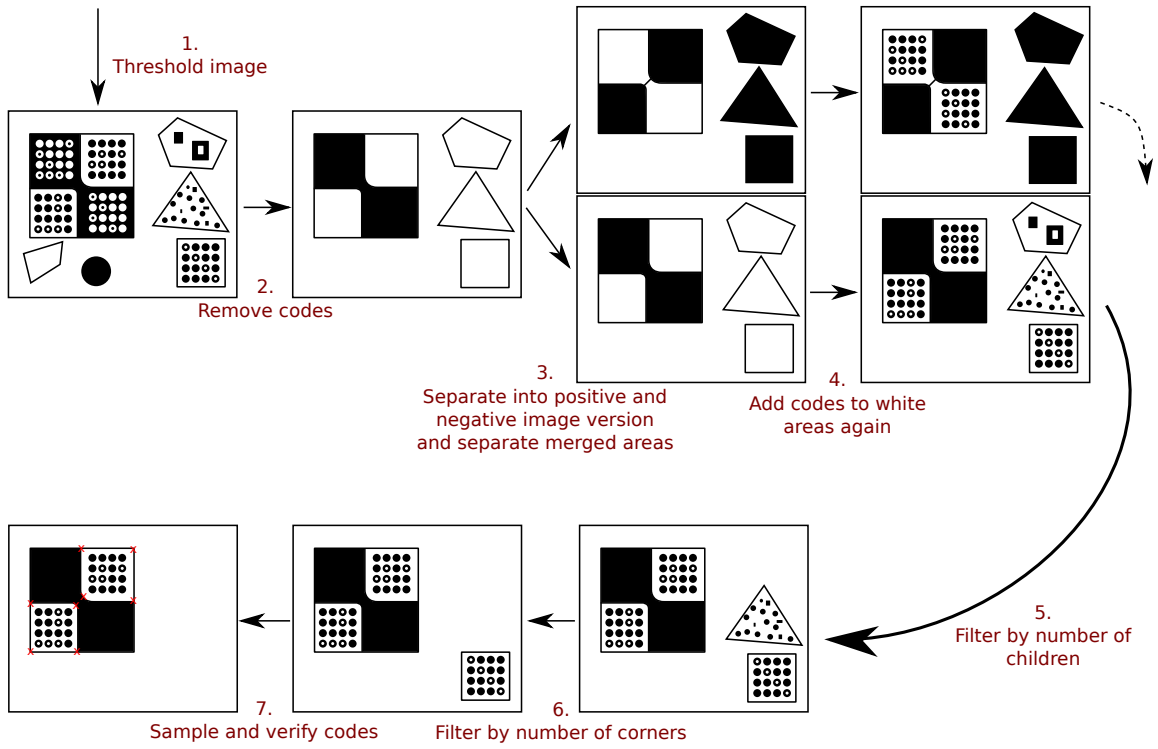


Figure 5: The tracking pipeline.

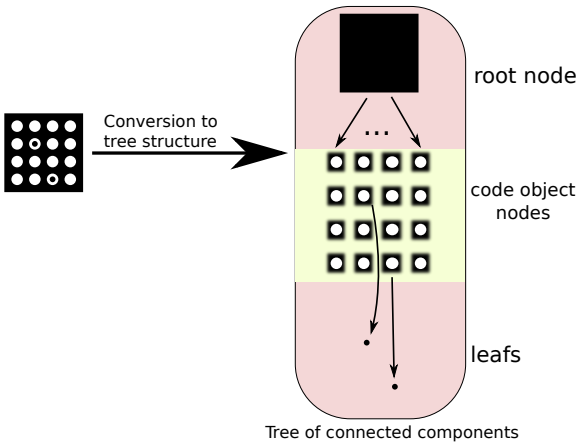


Figure 6: Conversion from pattern to tree structure. The hierarchy of the connected components is encoded as parent/children relations in the tree.

3. Splitting adjacent squares

At this step we have a binary image of the checkerboard without the circles. However, due to image blur the initial thresholding step results in merged squares, i.e. squares of equal color meeting at one corner appear as a single connected component. The goal in this step is to split these structures.

First, we determine the dominant points for each structure in the image by using the IPAN algorithm [CV98]. We use an angle threshold of 160°. Now we search for pairs of dominant points that only have a small distance from each other. To specify this distance we use a threshold T_x. The goal is to identify point pairs on opposite sides of a "bridge", i.e. an area where squares are merged. If T_x is chosen too large, squares can be split diagonally which has to be avoided to preserve their quadrilateral shape. For this reason we set $T_x = 20$ pixels which is related to the minimal diagonal size of usable squares. A square is usable if each circle can be detected. Consider the minimal case: the cross-section along the diagonal of a square provides 4×2 outer circle edges + 4 inner circles + 3 spaces between the circles and 2×2 parts between the outer circles and the square border. Assuming that each part consists of at least one pixel, the resulting minimal diagonal length is $2 \times 4 + 4 + 3 + 2 \times 2 = 19$ pixels.

After having identified the candidates for splitting we perform this operation by drawing lines between neighboring dominant points. The resulting binary image contains what our algorithm believes to be squares. In order to avoid separate implementations for the white and the black squares we work with two copies of the image, one being the inverse of the other. In the following, we describe the further processing for only one of those images.

4. Adding circles

At this point, we have a superset of square candidates. Our goal is to filter out all false positives. We achieve this by observing the inner structures of each square candidate. For this, we add the circles that were removed in the second step back into the image. This can be done by taking the original thresholded image (step 1) and the negated result of step 3 and performing a logical 'and' operation between the two images.

5. Filtering by number of circles

Our checkerboard pattern provides only squares with 16 circles. We take advantage of this fact by filtering out all candidates which do not meet this requirement. In our implementation, we convert the square candidate image to a component tree as in step 2. Now, we can traverse the tree and check the number of children of all connected components. All squares which do not provide 16 children are filtered from the tree. After that, the filtered tree is converted back to an image.

6. Filtering squares by number of corners

In this step we filter out all shapes that do not have four corners. They tend to be no squares. We compute the potential corners by a variant of the IPAN algorithm. We then identify the point with the steepest angle less than 135°. This threshold is used to avoid treating edges that have accidently been split by a dominant point. These points usually make angles close to 180° with their neighbors. In addition, if a perspective projection is so strong as for the 90° corner to project to an angle of larger than 135°, then the internal code of the square is likely to be of very low resolution and thus unusable. The output of this step is an image that contains square candidates with exactly 4 corners and 16 code objects.

7. Sampling the codes and verification of the squares

Next, we estimate the homography between the detected quadrilateral and an ideal two-dimensional $[0,1]^2$ space, [HZ10] which is used for sampling the code.

However, the corners computed so far are of low quality since they have been computed from the binary image. We increase the quality of the estimated points by merging close points of different squares to their mean position. It is important to ensure, that these points share the same real point. In addition we apply the Harris corner detector [HS88] on the gray value image.

5.2 PSF estimation

As outlined in Sect. 3, we can compute the combined PSF $p_{coc} \otimes p$ by deconvolving the observed image I^{OF}_{obs} with the definition of the pattern I_{obj}. For this it is necessary to remove the perspective projection of the recorded pattern, or, alternatively to warp the pre-defined pattern into the pose of the checkerboard. We choose the latter option to compute the PSFs directly in screen space. We do this by applying the inverse of the homographies computed in step 7 of the detection scheme to the pre-defined pattern. This process is performed independently for each square of the checkerboard. We use Wiener deconvolution [Wie49] to solve for $p_{coc} \otimes p$.

5.3 Pose and CoC estimation

As outlined above, the pose of the checkerboard allows for the estimation of the circle of confusion, the PSF of an ideal finite aperture camera system. Computing the checkerboard pose is equivalent to determining the camera's extrinsic parameters. We re-use the estimated homographies for this purpose. Coupled with the known real world size of each checkerboard square and the camera intrinsics, we derive the position and orientation of the checkerboard with respect to the camera [HZ10].

Based on the pose of the checkerboard, we compute the distance d between a selected point on the pattern and the camera. With the known distance of the focal plane S, the aperture diameter A and the focal length f we can compute the circle of confusion as

$$c = A\frac{|d-S|}{d}\frac{f}{S-f} \tag{4}$$

The circle of confusion diameter c needs to be converted to pixel size. Therefore, we need to regard the pixel dimensions of the camera sensor (e.g., Canon EOS 5D Mark II: 6.41 µm). The circle of confusion estimation can be applied at each position within the image, which is covered by the captured checkerboard pattern. The information on aperture size and focal length, and focal distance can be extracted from the EXIF tags accompanying the captured images.

5.4 In-focus PSF estimation

Once the circle of confusion p_{coc} and the combined PSF $p_{coc} \otimes p$ are known, the in-focus PSF p can be recovered by deconvolution. We again employ Wiener deconvolution for this task. A validation example of our procedure is shown in Fig. 7. In the first row we show ground truth PSFs obtained by applying Eq. 1. The checkerboard is placed in the focal plane which is placed at a distance of about $2.5m$. Due to the large field-of-view, the checkerboard cannot cover the complete image. In the second row we have moved the checkerboard to a much closer distance of about $30cm$. Here, the checkerboard covers the full image but appears blurry. The insets show the same region on the checkerboard. Correspondingly, the estimated combined PSFs are much larger than in the first row. The third row of the figure shows the estimated circles of confusion as well as the in-focus PSFs computed using our method. They agree quite well with the ground truth of the first row.

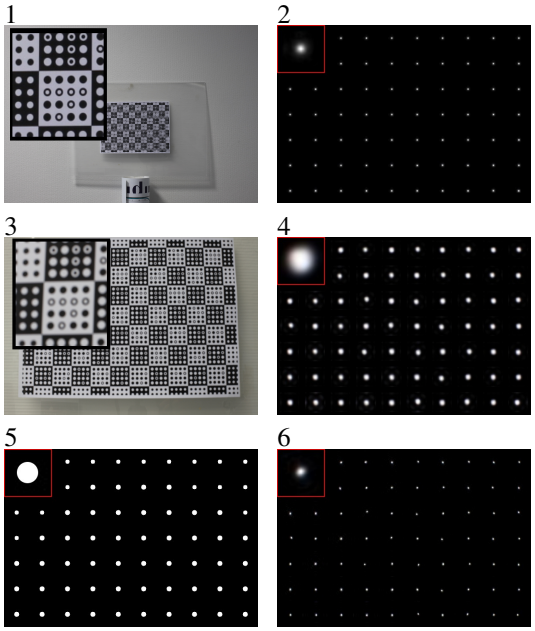


Figure 7: Estimation process of in-focus PSFs
1. Well focused, checkerboard in focal plane (~250cm)
2. Estimated PSFs from 1. incl. close-up
3. Same focus as in 1., but differently positioned checkerboard (~30 cm)
4. Estimated PSFs of 3.
5. Estimated circles of confusion of 3.
6. In-focus PSF estimated from 3.

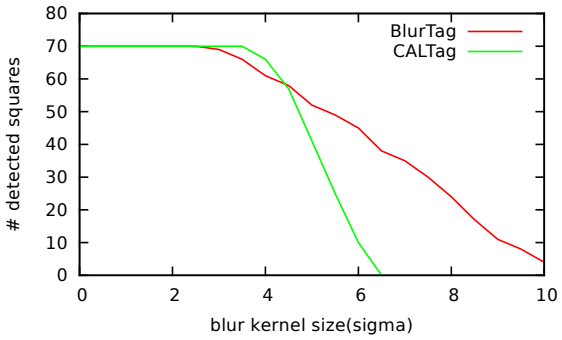


Figure 8: Comparison of BlurTag and CALTag

6 RESULTS AND DISCUSSION

We perform a number of tests to evaluate our algorithm. First we verify the detection process. Each square, which we fail to detect leads to a missing PSF. So the quality of our detection algorithm is critical. Further discussion about interpolating missing PSFs can be found in [KPCW11]. As mentioned in the overview section, we tend to capture blurry images which are challenging to process. For this reason, we implement tests to compare its stability with recent work.

6.1 Robustness to blur

Our first test compares our BlurTag with CALTag, the recent approach of [AHH10]. We generate synthetic images using our pattern and the one proposed in [AHH10]. Both synthesized images provide the same amount of squares and the same perspective transformation. During the test, we successively increase the blur. The number of detected squares versus the size of the blur kernel (sigma) is shown in Figure 8.

Figure 9: Detected squares in CALTag vs. BlurTag.
Left: Detected squares of CALTag-based checkerboard.
Right: Result of BlurTag-based checkerboard.

We see that CALTag and BlurTag have comparable detection results when the kernel size is small. Initially, BlurTag's performance decreases earlier than CALTag's due to the increase complexity of the circle pattern as compared to the squares of CALTag. Circles require a higher resolution to work properly compared to square based codes. With larger blur kernels, however, we see that CALTag's results deteriorate more quickly as compared to BlurTag. The conclusion of this test is that CALTag is preferable when the camera resolution is limited and the amount of blur is small. BlurTag shows its strengths with an increasing amount of blur at sufficient image resolution.

This observation is also manifest in a real-world test, Fig. 9. Large variations in square resolution, combined with significant amounts of blur lead to a reduced performance. However, The impact on BlurTag's performance is smaller than for CALTag. Whereas CALTag is able to detect $29/70 \approx 41\%$ squares, BlurTag extracts $52/70 \approx 74\%$ squares.

6.2 PSF quality for blurred capture vs. ground truth

The major goal of our approach is the estimation of circles of confusion to avoid their impact on the estimated PSFs. In Fig. 7 we show the final, in-focus PSFs (6) as compared to PSFs estimated from a pattern recorded in the focal plane (2). We see that the in-focus PSFs are in close agreement with those, which we estimated with a well focused checkerboard. It is important to keep in mind, that the focus was not changed during the experiment.

For numerical comparison, we compute the RMS error between the ground truth PSFs p of the in-focus recorded checkerboard (computed via Eq. 1) against the combined PSFs $p_{coc} \otimes p$ as well as the estimates in-focus PSFs using our method, Fig, 10. We see that the quality increase obtained from our method becomes better with an increasing diameter of the circle of confusion, i.e. the larger the blur, the more benefit to using

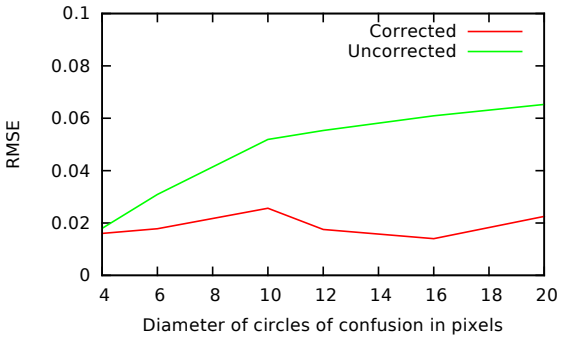


Figure 10: Root mean square error test:
Error between corrected or uncorrected PSFs and in-focus PSFs

our method. Another insight is that our computed in-focus PSFs have an approximately constant quality with respect to the ground truth over the tested blur sizes.

6.3 Video rates

While our main focus is on improved and more flexible PSF estimation, our method is fast enough to run at interactive rates and could thus be of interest for blur-resistant tracking in applications where AR-markers are traditionally used. We run on a test on a video with size 640×480 pixels. We achieve a mean speed of 13.04 fps. Our unoptimized implementation runs in a single-threaded process on an AMD Athlon(tm) 64 X2 Dual Core Processor 4600+. It should be possible to parallelize the algorithm on a GPU to achieve real time performance.

6.4 Application

Finally, we show results for sharpening a given target image using our estimated PSFs. Fig. 11 shows the application to a 22 Mpixel image taken with a Canon 5D mark II camera and a 50mm f/1.4 lens. We see that blur can be removed and the final image becomes sharper. Especially structures with high gradients like the handrails and the flowers improve in sharpness.

7 CONCLUSIONS AND FUTURE WORK

In this work, we introduced a new checkerboard pattern which is well suited to PSF estimation. Further, we introduced a new tracking algorithm, which is robust to resolution variations and large amounts of blur. In addition, we exploited the planarity of checkerboards to estimate out-of-focus blur of the calibration pattern. This allowed us to generate 'sharp' PSFs independent of the focus setting of the camera.

Our tracking approach is directly tailored to the designed checkerboard pattern. Therefore, changes in the design require different detection algorithms as well. In general, it would be possible to analyze the checkerboard patterns automatically to derive an appropriate tracking algorithm by analyzing its tree of connected components, as we did in the second step of our pipeline.

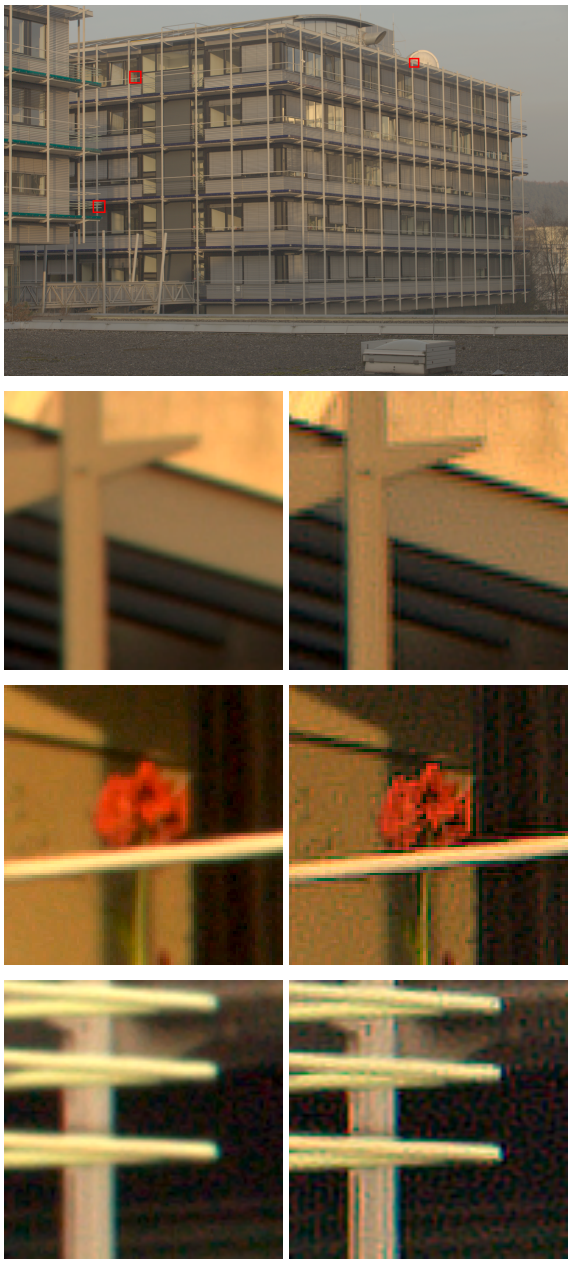


Figure 11: Results after deconvolution
Top: Target image with marked regions (best viewed in color)
Left: Blurred captured image regions
Right: Corresponding deconvolved image regions

Further, it would be interesting to analyze the shape of a PSF under varying focus settings. This could help manufacturers to enhance the quality of their lens systems with respect to different focus settings.

8 REFERENCES

[AD88] G.R. Ayers and J.C. Dainty. Iterative blind deconvolution method and its applications. *Optics letters*, 13(7):547–549, 1988.

[AHH10] Bradley Atcheson, Felix Heide, and Wolfgang Heidrich. Caltag: High precision fiducial markers for camera calibration. In *VMV'10*, pages 41–48, 2010.

[Bou02] J.Y. Bouguet. Complete camera calibration toolbox for Matlab. *Retrieved from the World Wide Web: http://www. vision. caltech. edu/bouguetj/calib doc/index. html*, 2002.

[Car01] Alfred S. Carasso. Direct blind deconvolution. *SIAM Journal on Applied Mathematics*, 61(6):pp. 1980–2007, 2001.

[CV98] D. Chetverikov and J. Verestoy. Tracking feature points: a new algorithm. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 2, pages 1436–1438. IEEE, 1998.

[dIMH02] D.L. de Ipiña, P.R.S. Mendonça, and A. Hopper. Trip: A low-cost vision-based location system for ubiquitous computing. *Personal and Ubiquitous Computing*, 6(3):206–219, 2002.

[FSH+06] R. Fergus, B. Singh, A. Hertzmann, S.T. Roweis, and W.T. Freeman. Removing camera shake from a single photograph. *ACM Transactions on Graphics (TOG)*, 25(3):787–794, 2006.

[GGSC96] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54. ACM, 1996.

[HS88] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[HZ10] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision 2nd ed.* Cambridge Univ Press, 2010.

[Jai89] A.K. Jain. *Fundamentals of digital image processing*. Prentice-Hall, Inc., 1989.

[JSK08] Neel Joshi, Richard Szeliski, and David J. Kriegman. PSF estimation using sharp edge prediction. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2008.

[KPCW11] E. Kee, S. Paris, S. Chen, and J. Wang. Modeling and removing spatially-varying optical blur. In *Computational Photography (ICCP), 2011 IEEE International Conference on*, pages 1–8. IEEE, 2011.

[NC06] Laurent Najman and Michel Couprie. Building the Component Tree in Quasi-Linear Time. *Transactions on Image Processing*, 15(11):3531–3539, 20006.

[Ric72] W.H. Richardson. Bayesian-based iterative method of image restoration. *JOSA*, 62(1):55–59, 1972.

[SJA08] Q. Shan, J. Jia, and A. Agarwala. High-quality motion deblurring from a single image. In *ACM SIGGRAPH 2008*, pages 1–10. ACM, 2008.

[VBC+02] P. Vivirito, S. Battiato, S. Curti, M. La Cascia, and R. Pirrone. Restoration of out of focus images based on circle of confusion estimate. In *Proceedings of SPIE 47th Annual Meeting*, volume 4790, pages 408–416, 2002.

[Wie49] N Wiener. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*, volume 47. Wiley, 1949.

[ZFN02] X. Zhang, S. Fronz, and N. Navab. Visual marker detection and decoding in AR systems: A comparative study. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, page 97. IEEE Computer Society, 2002.

[Zha00] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.