Advances in computational imaging: Benchmarking Deblurring Algorithms, Deep Neural Inpainting, Depth Estimation from Light Fields

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Dipl.-Math. Rolf Michael Köhler
aus Traunstein

Tübingen
2016

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation: 01.02.2016
Dekan: Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter: Prof. Dr. Hendrik Lensch
2. Berichterstatter: Prof. Dr. Bernhard Schölkopf

Zusammenfassung

Diese Dissertation umfasst drei Teilbereiche aus dem Gebiet der Bildverarbeitung.

Bewertung von blinden Bilddekonvolutionsalgorithmen: Ein Datensatz wurde generiert, der es ermöglichte, sieben moderne Bewegungsunschärfe-Entfernungs-Algorithmen zu vergleichen. Zur Generierung des Datensatzes wurden die 6D Bewegungstrajektorien der Kamera von verschiedenen Probanden aufgezeichnet und diese auf einem Hexapod unter gleichbleibenden Laborbedingungen wieder abgespielt. Der Benchmark Datensatz umfasst 48 Bilder: vier Szenen, die mit einer Spiegelreflexkamera aufgenommen wurde, welche während der Aufnahme jeweils mit den gleichen zwölf 6D Trajektorien bewegt wurde.

Die Resultate wurden statistisch analysiert, unter Verwendung von vier verschiedenen Bild Qualitäts Metriken, und es war möglich zu zeigen, dass der Algorithmus von Xu et al. [139], im Vergleich aller 48 Bilder und unter allen Bild Qualitäts Metriken, im Durchschnitt die besten Ergebnisse auf dem Datensatz erzielen konnte.

Bild-inpainting mit einem mehrlagigen Perzeptron: Inpainting ist das Problem fehlende Pixel in einem Bild sinnvoll zu ergänzen. Wir zeigen, dass es möglich ist, einen reinen Lernansatz zu verwenden, der die Inpainting Aufgabe lösen kann. Als Methode wurde ein mehrlagiges Perzeptron verwendet, welches als Eingabe einen korrumpierten Bildausschnitt und eine Maske desselben Bildausschnittes bekommt. Die Maske gibt an, welche Pixel fehlen. Das Perzeptron wurde auf Bildausschnitten trainiert, die auf Basis von Bildern aus dem ImageNet Datensatz [27], erzeugt wurden. Wir zeigen, dass die erzielten Ergebnisse, verglichen in der PSNR-Metrik, besser sind als moderne Inpainting Algorithmen. Wir zeigen zudem, dass es auch möglich ist ein mehrlagiges Perzeptron ohne die Eingabe der Maske zu trainieren: erzielte Ergebnisse sind, wie erwartet, nicht so gut, jedoch visuell immer noch ansprechend.

Tiefenschätzung aus Lichtfeld Bildern: Lichtfeld Photographie kann als Verallgemeinerung der Stereo Photographie aufgefasst werden. Ein besonderes Merkmal der Lichtfeldbilder ist die Sichtbarkeit von Linien in den sogenannten Epipolar plane images (EPI). Die Steigung dieser Linien ist umgekehrt proportional zu dem Abstand Objekt-Kamera. Der vorgestellte Tiefenschätzungsalgorithmus löst ein Optimierungsproblem um die Steigung dieser Linien zu schätzen. Da das Resultat der Optimierung verrauscht ist, verwenden wir zusätzlich einen Regularisierungsterm, der erzwingt dass Pixel mit ähnlicher Struktur und Farbe im RGB Bild einen ähnlichen Tiefenwert erhalten sollen. Durch Verwendung dieses Regularisierungsterms haben die Tiefenbilder schärfere Objektkanten, die mit den Objektkanten im RGB Bild übereinstimmen. Ein Vergleich mit anderen modernen Tiefenschätzungsalgorithmen aus der Literatur zeigt, dass unsere Ergebnisse vergleichbar sind und zudem feinere Strukturen in dem Tiefenbild zu Tage fördern.

Summary

This thesis contains three topics from the field of computational imaging.

Benchmarking blind deconvolution algorithms: We have built a dataset, that made it possible to compare seven state-of-the-art deblurring algorithms. We have recorded the 6D motion trajectories of the camera from several subjects and played it back on a Hexapod under the same laboratory conditions. The benchmark dataset contains 48 images: four scenes, which were captured with a SLR camera, which, for each scene, was moved with the same twelve motion trajectories.

The results were statistically analyzed, using four different image quality metrics. It was possible to show that the algorithm by Xu et al. [139] was on average able to output the best results on the dataset, comparing all 48 images and considering all image quality metrics.

Inpainting using a multi-layer perceptron: Inpainting is the task of completing missing pixels in an image with in a reasonable way. We show that a pure learning based approach is able to learn the inpainting task. The method we used was a multi-layer perceptron, where the input was a corrupted image patch and the corresponding mask of the same image location. The mask specifies which pixels are missing. The MLP was trained on image patches, which were generated by using images from the ImageNet dataset [27]. We show that the achieved results are better, compared in the PSNR metric, than state-of-the-art inpainting algorithms. In addition we show that it is also possible to train a multi-layer perceptron without the mask as input. The achieved results are, as expected, not as good, but still visually appealing.

Depth estimation from light field images: Light field photography can be regarded as a generalization of stereo photography. A salient feature of the light field images is the emerging of lines in the so-called epipolar plane images (EPI). The slope of those lines is inversely proportional to the distance object - camera. The presented depth estimation algorithm solves an optimization problem to estimate the slopes of those lines. As the result of the optimization is noisy, we use an additional regularization term, which enforces the pixels with similar structure and color in the RGB image to have a similar depth value. By using this regularization term the depth maps have more sharp object boundaries, which coincide with the object boundaries in the RGB image. A comparison with state-of-the-art depth estimation algorithms shows that our results are comparable and additionally show finer structures in the depth map.

Nomenclature

ANOVA	Analysis of variance
BD	Blind Deconvolution
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DMOS	Differential Mean Opinion Score
DSLR	Digital Single Lens Reflect
EPI	Epipolar Plane Image
FR-IQM	Full Reference Image Quality Metrics
GPU	Graphics Processing Unit
HVS	Human Visual System
IFC	Information Fidelity Criterion (image quality metric)
IQM	Image Quality Metric
JPEG	Joint Photographic Experts Group (image format)
L-BFGS-B	Limited-memory Bounded Broyden-Fletcher-Goldfarb-Shanno (optimization algorithm)
LED	Light-Emitting Diode
MAP	Maximum a Posteriori
MLP	Multi-Layer Perceptron
MRF	Markov Random Field
MS-SSIM	Multi-Scale Structural Similarity Index (image quality metric)
MSE	Mean Squared Error
NLM	Non Local Means (regularization)
NU	Non-Uniformness
PDE	Partial differential equation
PSF	Point Spread Function
PSNR	Peak Signal To Noise Ratio (image quality metric)
RF	Random Field
RGB	Red Green Blue (color image)
VIF	Visual Information Fidelity (image quality metric)

Acknowledgment

First of all I want to thank Prof. Dr. Bernhard Schölkopf for accepting me as a PhD student, and for the freedom to work on the topics I was interested in and in the way I wanted to do it.

Furthermore I thank Prof. Dr. Hendrik Lensch for instantly accepting the role as the university supervisor of my thesis and for interesting discussions.

Special thanks deserve Prof. Dr. Stefan Harmeling. He taught me how to write scientific papers and his open door policy triggered a lot of discussions.

I am also deeply grateful to Dr. Michael Hirsch for his co-authorship and especially for his role as a supervisor who had valuable tips for my last project.

Of great help during my dissertation was the IT service group: Thanks to Dr. Raffi Enficiaud, Joan Piles, Jojumon Kavalan and especially to Sebastian Stark, who were able to solve all technical problems I encountered in a fast and excellent way.

I also owe thanks for invaluable support to our secretaries Julia Braun, Andrea Odermatt and especially Sabrina Rehbaum. They made life at the institute easier.

I would also like to thank my fellow PhD students for sharing good times, especially Malte Kuhlmann and Philipp Geiger for their support and for having great discussions over lunch; Martin Kiefel, Alexander Loktyushin, Christian Schuler, Maren Mahsereci, Edgar Klenske and Behzad Tabibian for helping me out in areas where my expertise was not enough.

I also thank my brother Jan for reading over my thesis and giving great advice for improvement.

An unmeasurable amount of thanks goes to my parents. They have given me freedom and huge support to go to university and pursue the academic career I wanted to. Many thanks for the opportunities, which I now can enjoy, because you decided to invest in the education of your children.

The last and sincerest words of thanks deserve my wife Jorinde for her unconditional love, constant support and understanding for my flaws and deserve my two wonderful sons Ole and Ruben for reminding me everyday what really matters in life.

Contents

Papers included in this thesis

[64] R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf and S. Harmeling. Recording and Playback of Camera Shake: Benchmarking Blind Deconvolution with a Real-World Database. European Conference on Computer Vision (ECCV) 2012.

[67] R.Köhler, C. Schuler, B. Schölkopf and S. Harmeling. Mask-specific inpainting with deep neural networks. German Conference on Pattern Recognition (GCPR). 2014.

[66] (submitted) R.Köhler, B. Schölkopf and M. Hirsch. Precise Depth Estimation from Light Field Images. International Conference on Computer Vision (ICCV) Workshop on Inverse Rendering. 2015.

1

Introduction

In this chapter an introduction to each of the three contained topics from the field of computational imaging will be given.

1.1 Introduction to Image Deblurring

Image deblurring is the task of recovering a sharp image given a blurry image as input. If additionally the blurring kernel is given as input, it is called *non-blind deblurring* or *non-blind deconvolution* . If the only input is the blurry image it is called *blind deblurring* or *blind deconvolution*. Fig. 1.1 shows a blurry image and the deblurring result of the algorithm by [134].

According to Hansen et al. [49] various factors can lead to a blurry image, among them are:

- Misfocusing, i.e. a camera lens that is not focused to the desired object in the scene.
- Optical aberrations, mainly due to flaws in the manufacturing of a lens, e.g. photos are sharper in the center and more blurry and darker in the edges due to lens imperfections.
- Atmospheric and air turbulences altering the way of light rays hitting the camera. The result is a wobbling or flickering of the scene which can also be noticed when observing a scene through the hot air coming out of a chimney or rising from a hot street.

(a) Blurry image (b) Deblurred by [134]

Figure 1.1: A blurry image and a deblurred version of it. Note that both images were gamma corrected with $\gamma = 2.2$.

- Motion blur, caused either by the movement of an object (object motion) or by the movement of the camera (ego motion) while the shutter is open. Also a mixture of both object motion and ego motion occurs.

If not stated otherwise all blurred images in this chapter were the result of camera motion, while taking an image of a static flat 2D scene.

Commonly it is assumed that the blurring process is linear and hence can be written as (e.g. [49])

$$B = KI + n \tag{1.1}$$

with B being the blurry image, K being the blur kernel matrix and n being Gaussian noise.

Additionally most often the blur is assumed to be *stationary* (aka *uniform*), that strictly means that the camera only underwent translational motion. Eq. (1.1) can then be written as a convolution

$$B = k * I + n$$

with k being the so-called *blur kernel* .

Fig. 2.4 visualizes a non-stationary blur (aka *non-uniform blur*), Fig. 1.2 a stationary blur. The figures depict how a point grid would be recorded if the camera was shaken undergoing a stationary or non-stationary motion.

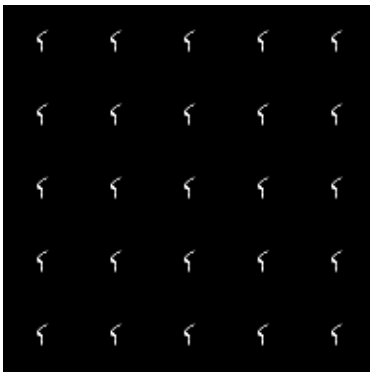


Figure 1.2: Example of a stationary point spread function. The blur kernel is translational invariant and looks the same in each position of the image. Fig. 2.4 shows an example of a non-stationary point spread function.

A stationary blur results in exactly equal looking blur kernels for every point in the image, assuming that the picture taking process is perfect. A non-stationary blur results in a point grid image where the blur kernels may look very different (compare the four corners in Fig. 2.4), but are smoothly varying.

The point spread function

The *point spread function* (PSF) expresses the effect of an imaging system on a point source. The name derives from the fact that a point spread function describes how much a point source is spread (or blurred). For a perfect imaging system the point spread function would just be a delta peak. The above mentioned factors of blurriness each lead to different looking PSFs. In Fig. 1.2 and Fig. 2.4 PSFs of a motion blur due to camera shake is depicted, assuming a perfect imaging system. Note that the brightness of the images were scaled in order to increase the visibility of the individual blur kernels.

In the case of a stationary camera motion (Fig. 1.2), neglecting all other sources of blurriness, a point spread function is shift invariant. Then a single *blur kernel* suffices to describe the effect of the camera motion on the whole image.

Assuming camera shake as the only blur factor, a point spread function can be regarded as a 2D visualization of the 6D camera trajectory.

For a more detailed introduction to point spread functions and imaging models we refer to [55].

From the benchmarked algorithms, only the algorithms by Hirsch et al. [56] and Whyte et al. [134] incorporate the non-stationarity of the blur into their model, while all other

algorithms assume stationary blur. The recorded PSFs by the author indicate that many camera shakes indeed are almost stationary, as discussed in chapter 2.5.1.

The algorithm by Hirsch et al. [56] approximates the 6D camera trajectory by the two translations along the x and z axis and the rotation about the y axis, whereas the algorithm by Whyte et al. [134] uses all three rotations about the x, y and z axis as an approximation, see Fig. 2.1.

Gupta et al. [47] discuss that three dimensions suffices in most cases to approximate the 6D camera trajectory: rotation about the x axis can well be approximated by translation along the z axis, and rotation about the z axis by translation along the x axis. On their project web page [46] they show examples, which visually confirm their claim.

1.2 Overview of blind deconvolution algorithms

Early publications in blind deconvolution (BD) were inspired by shortcomings in astronomical imaging, including e.g. [102] and [81], which were published in the early 1970s. The article by [70] gives an overview of related methods.

One of the first works to apply BD to the problem of removing camera shake from a single photograph was [36], combining the variational approach of [90] with natural image statistics [38]. Subsequent work refined the approach [109], introduced new inference strategies and fast optimization techniques [18], and proposed methods for robust kernel estimation [139, 69, 74]. See [73] for a comprehensive overview of these and related approaches.

Recent work [116, 117, 134, 47, 50, 56, 133] focuses on deriving new imaging models that are better capable of capturing real motion blur that often violates the uniform blur assumption of previous methods.

Hardware-based approaches to obtain sharper images are based on manipulating the way images are taken. For instance, [142] reconstruct a single sharp image from a pair of blurred and noisy images. While [101] encodes the movement of objects by "fluttering" the shutter, [19] is able to remove linear object motion by capturing two images of the scene with a parabolic motion in two orthogonal directions. [60] exploit inertial measurement sensor data to recover the true trajectory of the camera during exposure.

After having given an overview of blind deconvolution algorithms, we will now introduce each of the benchmarked algorithms in detail.

1.2.1 Fergus et al.: Removing Camera Shake from a Single Photograph [36]

In their work a Bayesian approach to infer the blur kernel k is presented. It was to our knowledge the first paper to introduce image statistics of the gradients of natural images as a prior.

It assumes stationary blur, i.e. blur that is the same for all pixels in the image.

Blur Kernel Estimation

In this paper a coarse-to-fine multiscale approach is applied for kernel estimation.

As additional input the user

- selects a rectangular patch B_p in the image that is "rich in edge structure" [1]. On this region the blur kernel is estimated.
- an initial guess of the blur kernel orientation (horizontal or vertical),
- maximum size of the blur kernel.

The inference of the kernel is done using image gradients (and not image intensity values). A Bayesian approach with the following posterior is applied

$$p(k, \nabla I_p | \nabla B_p) \propto p(\nabla B_p | k, \nabla I_p) p(\nabla I_p) p(k) \tag{1.2}$$

with likelihood

$$p(\nabla B_p | k, \nabla I_p) = \prod_i \mathcal{N}(\nabla B_p(i) | k \otimes \nabla I_p(i), \sigma^2) \tag{1.3}$$

(where i denotes a pixel in the patch I_p and $\otimes$ denotes the convolution operator), with a prior $p(\nabla I_p)$ on the image gradients of the user selected patch I_p, a sparsity prior $p(k)$ on the kernel k. For $p(\nabla I_p)$ a heavy-tailed distribution is used (a mixture of four zero-mean Gaussians). Figure 1.3 shows this distribution of the image gradients. For $p(k)$ a mixture of 4 exponential distributions is used. To solve this posterior a maximum-a-posteriori (MAP) solution failed. Fergus et al. hence followed an approach by [90], in which the posterior distribution is approximated and the blur kernel k is then computed using the maximum marginal probability. Code for the approach by [90] can be found at [89].

To prevent getting caught in a local minimum a coarse-to-fine multiscale approach is applied. They start with an initial coarse 3×3 kernel (user supplied, either a horizontal or vertical line). In each scale s the estimated kernel k_s and the estimated latent image gradients ∇I_p are upsampled and used as initialization for the next scale.

[1] Sec. 4.1.2 in [36]

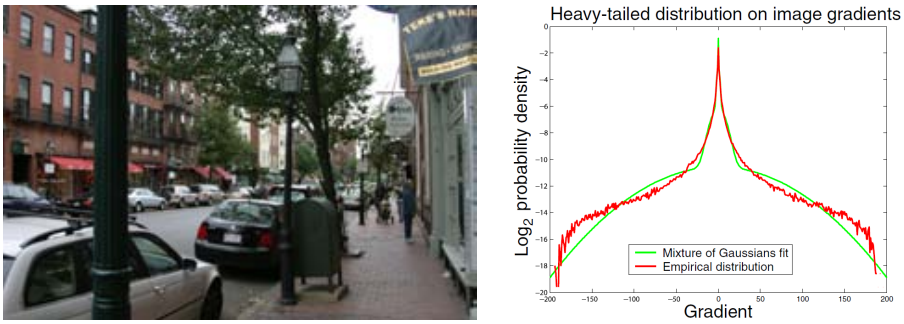


Figure 1.3: This figure is taken from [36]. **Left:** *natural image,* **Right:** *empirical and modeled distribution of the image gradients. The heavy-tailed distribution represents the gradients found in a natural image better than a normal distribution, as latter has no heavy tails. Most gradients in a natural image are small (due to smooth regions, like a sky or a street), but there is also mass on large gradients, which are mostly due to edges or boundaries between objects.*

Image Reconstruction

After estimating the kernel k at the finest scale, the kernel is thresholded to reduce noise: values smaller than $\max(k)/15$ are set to 0. In the last step the Lucy-Richardson (LR) algorithm [102, 81] is applied for deconvolution. As the deconvolution is done in the Fourier domain, the image is edge-tapered [2] before applying the LR algorithm.

1.2.2 Shan et al.: High-quality Motion Deblurring from a Single Image [109]

Shan et al. use a Bayesian approach to infer the deblurred image and the blur kernel. They contribute a new likelihood term and a refined image prior $p(I)$, which they split into a global and a local image prior.

The posterior is

$$p(k, I|B) \propto p(B|k, I)p(I)p(k) \tag{1.4}$$

where the likelihood $p(B|k, I)$ incorporates the spatial randomness of image noise. This is achieved by using the first and second derivatives of the blurred image. This helps to better estimate noise and reduce ringing artifacts. The likelihood is given by:

$$p(B|I, k) = \prod_{\partial^* \in \theta} \prod_i \mathcal{N}(\partial^*_{I_i}|0, \zeta_{\kappa(\partial^*)}) \tag{1.5}$$

[2] using MATLAB's edgetaper function

where i denotes a pixel in the image, $\theta = \{\partial^0, \partial_x, \partial_y, \partial_{xx}, \partial_{xy}, \partial_{yy}\}$ the set of derivative operators, $\kappa(\partial^*) = \{0, 1, 2\}$ the order of the derivative operator and $\zeta_1 = \sqrt(\zeta_0), \zeta_2 = \sqrt(\zeta_1)$ the standard deviations of the normal distribution.

The prior $p(k)$ is an exponentially distributed sparsity prior on the kernel k:

$$p(k) = \prod_j e^{-\tau k_j} \qquad k_j \geq 0$$

with a rate parameter τ and k_j being one element of the blur kernel.

The prior on the sharp image $p(I) = p_g(I)p_l(I)$ is decomposed into a local ($p_l(I)$) and a global ($p_g(I)$) prior. The **local prior** is designed to suppress ringing artifacts. It is based on the observation that smooth regions in the blurred image B highly likely are also smooth in the deblurred image I. Ringing artifacts destroy the smoothness in originally smooth regions by introducing high frequencies. The local prior is expressed as:

$$p_l(I) = \prod_{i \in \Omega} \mathcal{N}(\partial_x I_i - \partial_x B_i | 0, \sigma_1) \mathcal{N}(\partial_y I_i - \partial_y B_i | 0, \sigma_1) \tag{1.6}$$

with Ω being the set of locally smooth pixels. Each pixel $i \in \Omega$ is found by computing the standard deviation of a window centered in i. If the standard deviation is smaller than a threshold $t = 5$, the pixel i is said to be smooth and put into the set Ω. This local prior enforces that pixels in the gradient images of B and I are similar.

The **global prior** $p_g(I)$ alleviates the solution of the ill-posed deconvolution problem. It is defined on the image gradients and is a approximation of a logarithmic density (a heavy-tailed distribution of the image gradients)

$$p_g(I) \propto \prod_i e^{\Phi(\partial I_i)} \tag{1.7}$$

with $\Phi(x) = \begin{cases} -k|x| & x \leq l_t \\ -(ax^2 + b) & x > l_t \end{cases}$ and $k = 2.7, a = 6.1 \cdot 10^{-4}, b = 5.0$. l_t is not specified in the paper. The advantage of using a prior with the concatenated function $\Phi(x)$ over a mixture of Gaussians (as used by [36]) is the easier ability to optimize the energy function.

Optimization

To be able to solve the MAP approach, Shan et al. iteratively optimize between the kernel k and the sharp image I, keeping one value fixed while optimizing the other value. Additionally in each optimization step the influence of the prior terms and the likelihood term are reweighted. At the beginning both the global prior is emphasized to

boost edge reconstruction and the local prior to suppress ringing artifacts. In the course of the optimization the influence of the likelihood term is increased. This enables the recovery of fine details in the latent image I.

1.2.3 Xu, Jia: Two-Phase Kernel Estimation for Robust Motion Deblurring [139]

Xu and Jia's main contribution is the proposal of a new kernel estimation scheme, which contains two phases. It is based on the so called *rmap*. They observed that not every edge information facilitates kernel estimation. When the blur kernel is larger than the size of an object, the edges of that object are not useful for blur kernel estimation, e.g. in a panorama image of many small houses with a blur kernel that is larger than the houses.

The **first phase** is the kernel initialization phase and consists of following steps:

- *shock-filtering B:* first a shock-filtered image $\tilde{I}$ is generated (by first applying a Gaussian filter to the blurry image B and then using the shock filtering procedure as described in [94]).
- *computing the r-map:* following metric is applied to the blurry image B. It is a measure of the importance of an edge in the image.

$$r(x) = \frac{\left\| \sum\limits_{y \in N_h(x)} \nabla B(y) \right\|}{\sum\limits_{y \in N_h(x)} \|\nabla B(y)\| + 0.5} \tag{1.8}$$

  where $N_h(x)$ is a $h \times h$ window with center pixel x and $\nabla B(y)$ is the image gradient at pixel y. For small/narrow objects the differently signed gradients of the object's boundaries are contained in $\left\| \sum\limits_{y \in N_h(x)} \nabla B(y) \right\|$ and almost cancel out, whereas the sum $\sum\limits_{y \in N_h(x)} \|\nabla B(y)\|$ gives an estimation of the image structure in the $h \times h$ window. Small values of r indicate that the $h \times h$ window has either many small objects or consists of a flat region. Pixels where the product $M \left\|\nabla \tilde{I}\right\|_2$ is below a certain threshold are masked out, with M being a binary mask that is 1 for large enough r-values and 0 otherwise. Hence pixels which have a small r-value or where the shock filtered image $\tilde{I}$ has small gradients $\left\|\nabla \tilde{I}\right\|_2$ are masked out in $\nabla \tilde{I}$. The remaining edges in $\nabla \tilde{I}$ are denoted by ∇I^s. That is the support of the r-map. The r-map indicates image edges which are useful for the deblurring task.

- *Fast initial closed-form kernel estimation.* To get a rough estimate of the kernel k using the r-map ∇I^s following objective is used:

$$E(k) = \|\nabla I^s \otimes k - \nabla B\|^2 + \gamma \|k\|^2 \tag{1.9}$$

  with $\otimes$ being the convolution operator. It is possible to derive a closed-form solution for k, see the paper [139] for details.

- *Coarse Image Estimation:* To get a rough estimate of the latent image I following objective is used

$$E(I) = \left\|I \otimes k - B\right\|^2 + \lambda \|\nabla I - \nabla I^s\|^2 \tag{1.10}$$

  where the computed r-map ∇I^s serves as a prior on the gradients ∇I. Here as well, it is possible to find a closed-form solution for I.

The **second phase** is the kernel refinement phase: For iterative estimation of the kernel, hard thresholding the kernel can result in loosing fine structures of the kernel. Xu and Jia hence apply an iterative support detection (ISD) method by [124]. In each iteration the regularization penalty is reduced for pixels of the PSF with large values. This ensures that those pixels will not be heavily affected by the regularization in the next iteration, see [124] for details.

Deconvolution

Having estimated the kernel a TV-l_1 deconvolution model is used:

$$E(I) = \|I \otimes k - B\| + \lambda \|\nabla I\| . \tag{1.11}$$

It is solved with a variable substitution scheme, which is based on the work of [123, 141].

1.2.4 Whyte et al.: Non-uniform Deblurring for Shaken Images [135]

In this paper Whyte et al. propose a new model to describe non-uniform camera shake. This model is then applied to three [90, 36, 18] uniform deblurring algorithms.

The model is a geometric model, in which only the three rotations are used to model the ego-motion of the camera, see Fig. 2.1. It is claimed that rotation of the camera has a larger effect on the image blur than translation. [3]

[3] e.g. for a 45° field of view, 2000 × 2000 px image, to get a blur of approximately $22px \approx \frac{0.5}{45} 2000px$ pixels, a camera has to be rotated only by $0.5° = \frac{0.5}{45} 45°$, whereas the translation of the camera for a distant object of e.g. 5m has to be about $5.5cm \approx \frac{22}{2000} \cdot 5m$ (assuming that the focal length is about the same as the sensor width).

To describe the motion caused by the rotation, homographies are used:

$$H = KRK^{-1} \tag{1.12}$$

with K being the internal calibration matrix of the camera and R being the Rotation matrix. R is given by

$$R(\theta) = e^{[\theta]_x} \qquad \text{with} \quad [\theta]_x = \begin{bmatrix} 0 & -\theta_Z & \theta_Y \\ \theta_Z & 0 & -\theta_X \\ -\theta_Y & \theta_X & 0 \end{bmatrix}. \tag{1.13}$$

The observed blurry image B can thus be described as the integral over time of all homographically transformed versions of the latent image I

$$B(x) = \int_0^T I(H_t x)dt + n, \tag{1.14}$$

with $B : \mathbb{R}^2 \to \mathbb{R}$ being the blurred image and $I : \mathbb{R}^2 \to \mathbb{R}$ the static scene and x being a point in the blurred image B.

As a blurry image contains no direct temporal information Eq. (1.14) is replaced by

$$B(x) = \int_{\theta \in \mathcal{R}} I(H_\theta x)\omega(\theta)d\theta + n \tag{1.15}$$

where $\omega(\theta)$ is a weight function corresponding to the time spent at orientation θ. $\mathcal{R}$ is the set of all rotations.

Discretizing Eq. (1.15) leads to

$$B_i = \sum_k \omega_k \sum_j C_{ijk} I_j + n \tag{1.16}$$

where C_k is the matrix which applies homography H_k to the latent image I, C_{ijk} is an element of it, B_i is an observed pixel in the blurry image, k indexes the set of homographies/camera orientations, j the latent image I and i the blurry image B. The sum $\sum_j C_{ijk} I_j$ is an interpolation of the point $H_k x_i$ in the sharp image I. The weights ω_k can be regarded as a measure for the time spent at a certain camera orientation θ_k.

This model is then incorporated into the uniform deblurring algorithms by [90, 36] and [18], using the respective regularization terms of each algorithm.

1.2.5 Hirsch et al.: Fast Removal of Non-uniform Camera Shake [56]

This is one of the two (besides Whyte et al. [135]) non-uniform deblurring approaches used in the benchmark. Whyte et al. [135] use three rotations (θ_X, θ_Y, and θ_Z) to approximate and parametrize the $6D$ trajectory of the camera. Hirsch et al. use two translations T_X and T_Y (parallel to the sensor plane) and rotations θ_Z about the optical axis to describe the camera motion, see Fig. 2.1.

Non-uniform blur is approximated as a sum of blurry overlapping patches, each uniformly blurred with a possibly different kernel:

$$B = \sum_r a^{(r)} * \left(\omega^{(r)} \odot I\right) \tag{1.17}$$

with $a^{(r)}$ being the blur kernel of the r-th patch, $*$ the convolution operator, $\odot$ the Hadamard product and $\omega^{(r)}$ the weighting windowing, so that $\omega^{(r)} \odot I$ represents the r-th patch.

To constrain the blur to physically feasible camera blur, a basis of possible blurs is computed. The basis consists of a point grid p (=grid of single pixel dots) to which different homographies, denoted by H_θ, are applied. Each basis element $p_\theta = H_\theta(p)$ is then cut into local blur kernels $b_\theta^{(r)}$.

Note that the basis elements p_θ can be precomputed, with which the 6D trajectory of the camera is approximated by two translations T_X and T_Y and the rotation θ_Z.

The blur kernels $a^{(r)}$ are then restricted to be a weighted sum of those local blur kernels $b_\theta^{(r)}$

$$a^{(r)} = \sum_\theta \mu_\theta b_\theta^{(r)} \tag{1.18}$$

with μ_θ being a weight vector.

To get the final model, Eq. (1.18) is plugged into Eq. (1.17):

$$B = \sum_r \left(\sum_\theta \mu_\theta b_\theta^{(r)}\right) * \left(\omega^{(r)} \odot I\right) =: \mu \diamond I \tag{1.19}$$

Deblurring is done, as in the other described methods, in two steps: (I) determining the non-uniform blur kernel, (II) recovering the latent image by non-blind deconvolution.

(I) To estimate the kernel an iterative coarse-to-fine procedure is applied: in its (i) prediction step a rough blur kernel is estimated. In the (ii) blur parameter update step the weights μ_θ for deeming the blur kernel $a^{(r)}$ are updated by minimizing

$$\left\|\partial B - m_S \odot \partial(\mu \diamond \tilde{I})\right\|_2^2 + \frac{1}{10}\|\mu\|_2^2 + \frac{1}{2}\|\partial\mu\|_2^2 \tag{1.20}$$

where ∂B denote the gradient image of B in the horizontal and vertical direction, $\tilde{I}$ is the result of the prediction step and m_S is a weighting mask, computed using the r-map approach by Xu et al. [139].

In the (iii) sharp image update step a rough estimate of the sharp image I is computed by minimizing

$$\|B - \mu \diamond I\|_2^2 + \frac{1}{2} \|\partial I\|_2^2 . \tag{1.21}$$

The steps (i), (ii), (iii) are iteratively repeated to estimate the parameters μ_θ.

(II) The latent sharp image I is then recovered by minimizing

$$\|B - \mu \diamond I\|_2^2 + \nu \|\partial I\|_\alpha^\alpha \tag{1.22}$$

where a natural image prior $\|\partial I\|_\alpha^\alpha$ is applied, e.g. as used in [36].

1.2.6 Cho and Lee: Fast Motion Deblurring [18]

Cho and Lee propose an iterative coarse to fine approach for the kernel estimation. The blur kernel is estimated by iteratively performing following three steps

(I) prediction step

(II) kernel estimation

(III) deconvolution

(I) In the prediction step image gradient maps $\{P_x, P_y\}$ are computed by (i) applying bilateral filtering [120] to the estimation of the latent image I in order to get rid of small image structure (including noise). (ii) Applying a shock filter to enhance salient edges of I. (iii) Thresholding the gradients $\partial_x I', \partial_y I'$ of the shock filtered image I', in order to suppress noise, which was intensified during the shock filtering step.

(II) In the kernel estimation step following energy function is minimized

$$f_k(k) = \sum_{(P^*, B^*)} \omega_* || k * P_* - B_* \big\|^2 + \beta \big\| k||^2 \tag{1.23}$$

with a Tikhonov regularization term $\beta \|k\|^2$ on the kernel and with

$$(P_*, B_*) \in \{(P_x, \partial_x B), (P_y, \partial_y B), (\partial_x P_x, \partial_{xx} B), (\partial_y P_y, \partial_{yy} B), ((\partial_x P_y + \partial_y P_x)/2, \partial_{xy} B)\}. \tag{1.24}$$

Eq. (1.23) enforces that the blurred image gradient map $(k * P_*)$ and the gradient image B_* of the blurry image B should be similar. Eq. (1.23) is minimized using a conjugate gradient method.

(III) In the deconvolution step the latent image I is estimated by minimizing

$$f_I(I) = \sum_{\partial_*} \omega_* ||k * \partial_* L - \partial_* B \|^2 + \alpha \| \nabla L||^2 \tag{1.25}$$

with $\partial_* \in \{\partial_0, \partial_x, \partial_y, \partial_{xx}, \partial_{xy}, \partial_{yy}\}$ and $\omega_* \in \{\omega_0, \omega_1, \omega_2\}$ being a weight for each partial derivative ∂_*. Eq. (1.25) can be efficiently minimized by pixel-wise division in the frequency domain, needing only two Fast-Fourier-Transforms.

This estimated latent image serves in the next iteration as an input for better estimation of the blur kernel.

After iteratively estimating the blur kernel, the smallest elements (with values smaller than 1/20 of the maximal value) are set to 0 and the resulting kernel is normalized. With this kernel non-blind deconvolution is performed using the method proposed by [109].

1.2.7 Krishnan et al.: Blind Deconvolution Using a Normalized Sparsity Measure [69]

The main contribution of this paper is the introduction of a new image regularization term: the ratio between the l_1 and l_2 norm applied to a high frequency image

$$\frac{\|x\|_1}{\|x\|_2} \tag{1.26}$$

where x is the high frequency image, e.g. the gradients $(\nabla_x I, \nabla_y I)$ of the latent image I.

Only applying the l_1 norm on the high frequencies of an image would favor a blurry image, as the blur reduces the l_1 norm. The ratio between l_1 and l_2 norm can be interpreted as a normalized version of the l_1 norm. If an image is blurred, the l_1 norm is reduced less than the l_2 norm, hence the ratio l_1/l_2 increases. This property of the l_1/l_2 ratio makes it a good regularization term for deblurring.

The l_1/l_2 regularization must be applied to the high frequency image, e.g. the stacked gradient image $(\nabla_x I, \nabla_y I)$. If applied to an image containing all frequency bands, the change in the l_1/l_2-function of a blurred image would be rather small compared to the l_1/l_2-function value of the original unblurred image.

Blur Kernel estimation:
The blur kernel estimation is done in a coarse-to-fine approach, similarly to [36], with a size ratio of $\sqrt{2}$ between each scale-level.

Following model is used to estimate the blur kernel:

$$\min_{x,k} \lambda \|x \otimes k - y\|_2^2 + \frac{\|x\|_1}{\|x\|_2} + \psi \|k\|_1 \tag{1.27}$$

with x being the sharp image in the high-frequency space, e.g. generated by the gradient operators ∇_x and ∇_y, k is the blur kernel, $y = [\nabla_x B, \nabla_y B]$ the gradients of the blurred image and λ and ψ scalars. The l_1 prior on $\|k\|_1$ helps to reduce noise in the kernel.

Eq. (1.27) as being non-convex is optimized by alternately optimizing and updating x and k.

Image Recovery:
The latent sharp image is recovered using the non-blind deconvolution method by [68], in which following cost function is minimized w.r.t I:

$$\min_I \lambda \|I \otimes k - B\|_2^2 + ||\nabla_x B\|_\alpha + \|\nabla_y B||_\alpha \tag{1.28}$$

with $\alpha = 0.8, \lambda = 3000$.

1.3 Introduction to image inpainting

Image inpainting tries to fill-in missing parts of an image. The term *inpainting* was adopted from art restorers and first used by [3] in his seminal inpainting paper. Before that it was known as *image interpolation*. Fig. 1.4 shows a photography of an easel painting with missing parts due to cracks and the restored version of it. The restoration was done by an art restorer.

Figure 1.4: The art of image restoration: Manually inpainted and restored easel painting "Birth of the Milky Way" by an unknown artist. The restoration was done as a graduation project by the student Michal Mišáni [88]. It is housed in Cerveny Kamen Museum, Casta, Slovakia.

Inpainting is an ill-posed problem, it has no unique solution. The goal is to modify an existing image in a way that the resulting image looks plausible, natural and pleasing to the observer and that the image modifications are non-detectable.

Commonly, one can distinguish two settings, where pixels in an image are missing, *image completion* and *hole removal*. In the following both settings will be shortly described.

Image completion

Image completion : In the first setting, the goal is to manipulate an existing image. Usually, some image details or larger regions of a given image should be removed, see Fig. 1.6. The resulting hole must be filled in to create a *plausible* complete image. For instance, consider an image with two persons, where one person should disappear, e.g. in Fig. 1.5. Then the task of image inpainting is to fill-in the resulting (possibly large) hole with some background textures or patterns. The goal is not to recover a *true* image but one that looks realistic. Many successful methods (some based on the seminal paper on *texture synthesis* by [30]) have been proposed in the past, for instance [29, 98, 21, 140, 80] and references therein. These ideas can be also generalized to video inpainting [132]. Further below we will shortly introduce the main principle of the exemplar-based inpainting method, which is based on texture synthesis.

Figure 1.5: Image completion / object removal in censorship: The left image shows Lenin with a group of men, including Alexander Marchenko, who felt in disgrace, was executed and removed from further reproductions of the image. The photography was altered by Nadezhda Konstantinovna Krupskaya. The images were taken from [136] and also are featured in the book [63].

Above methods fill an undesired region of an image by using only the image information available in the image itself. There also exists methods for image completion, which copies pixel information from external images. Hays and Efros [51] for example, use an external dataset of millions of images. To inpaint an image, images from the dataset depicting a similar scene are found by using the gist scene descriptor developed by [93]. The area, which needs to be inpainted, is cut from one of the found image and seamlessly blended by using the technique of Poisson blending [97]. Fig. 1.6 shows a result of this algorithm.

Besides removing distracting or unwanted objects in an image, image completion can also be used in the area of privacy protection, e.g. to create visual appealing images,

while protecting the privacy of individuals [16]. In google street view, e.g. faces or license plates are blurred out. Instead of blurring these objects, they could be inpainted to produce a visual more plausible image while keeping the privacy. Bitouk et al. [7] present a work on automatic face replacement, which can be used for face de-identifying.

Figure 1.6: Image completion: Result of the algorithm by [51], which uses a dataset of million of images to inpaint a whole region. Images are taken from [51].

Exemplar-based image inpainting

Exemplar-based image inpainting is a method that is suitable for inpainting large holes in an image, e.g. for object removal. It is based on the texture synthesis problem and exploits the self-similarity in an image. In the following we will explain the main idea of the algorithm proposed in the seminal paper [21].

An unknown region Ω is inpainted by selecting one pixel p from the boundary $\delta\Omega$, see Fig. 1.7. In the next step the best matching patch $\Psi_{p'}$ is found, for which the known part of Ψ_p and the corresponding part in $\Psi_{p'}$ match best, according to a metric. The metric used in [21] is the sum of squared differences of the corresponding regions of the two patches. From the patch $\Psi_{p'}$ pixels are copied into the missing region of Ψ_p.

The order of selecting the pixel p from the boundary $\delta\Omega$ is crucial. Selecting the pixels in the wrong order can lead to image artifacts. [21] proposes to give higher priority to boundary pixels that lie on a region of continuing image structure, e.g. a line.

Hole removal

Hole Removal: The second setting of image inpainting algorithms considers an image that is locally corrupted, see e.g. Fig. 1.4. In this case, the missing regions are small but possibly all over the image. Fields of application comprises scratch removal, text or logo removal, red eye removal, wire removal holding movie actors and disocclusion when rendering new views. The goal is to recover an image which is as close as possible to the *true* image. Also for this setting many good approaches exist, which can be categorized into two groups:

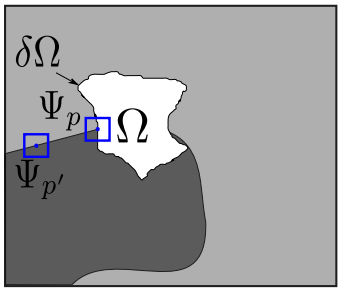


Figure 1.7: Exemplar based inpainting: For a patch Ψ_p, centered at the boundary $\delta\Omega$ similar patches $\Psi_{p'}$ are searched. The information from the best matching patch $\Psi_{p'}$ is used for the inpainting.

(i) Many classical approaches are *diffusion-based* methods that propagate the local information, such as edges and gradients, from the boundary to the missing pixels, see e.g. [87, 3, 14]. Further below we will sketch the ideas of the PDE-based inpainting method by [3].

(ii) The second class is based on *sparse representations* using dictionaries [31, 85, 34]. General purpose image priors based on Markov random fields (MRF) can be learned on image databases [103, 105]. These have been also successfully applied to inpainting.

Furthermore inpainting techniques are also used in image compression, see [79]. In the original image appropriate pixels are deliberately not saved and later reconstructed from the image.

Inpainting with PDEs

In the following we will sketch the idea of the partial differential equation (PDE)-based inpainting method of the seminal paper by [3]. The goal is to fill the unknown region Ω with suitable color by propagating information from the outside into the unknown region Ω by following the iosphotes,[4] see Fig. 1.8.

This is done by solving the following PDE

$$\frac{\partial I}{\partial t} = \nabla(\Delta I) \cdot \nabla^{\perp} I, \tag{1.29}$$

with the steady state solution being given by setting $\frac{\partial I}{\partial t} = 0$. The parameter t is an artificial time marching parameter. If the steady state equation is fulfilled, the change in

[4] Isophotes are the lines of constant gray value/brightness in an image.

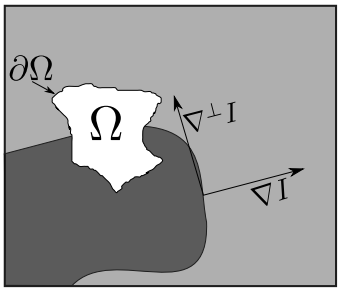


Figure 1.8: PDE based inpainting: The pixel information is propagated into the unknown region Ω by following the appropriate isophotes direction $\nabla^{\perp} I$.

the Laplacian $\nabla(\Delta I)$ is 0 along the isophotes direction $\nabla^{\perp} I$, meaning that the Laplacian is constant along that direction. The Laplacian is a way to measure smoothness in the image. If the change in the Laplacian is 0, the continuation of the image into the unknown region Ω is smooth. In [3] the numerical implementation of this PDE is described.

This PDE-based approach has difficulties with propagating texture into the unknown region, the major downside of this approach. Especially for larger unknown regions Ω a PDE-based approach results in a blurry filling of the unknown region Ω.

1.4 Introduction to light field photography

Research on light fields has increasingly gained in popularity, driven by technological developments and especially the launch of the First Generation Lytro consumer light field camera [91] in 2012. In 2014, Lytro launched the "Illum" follow-up model. While these cameras are targeted to the end-consumer market, the company Raytrix [99] manufactures high-end light field cameras, some of which are also capable of video recording, but aimed for the industrial sector. Both Lytro and Raytrix use an array of microlenses to capture a light field with a single camera.

There are also trends to include the lightfield technique into smartphones, e.g. the "HTC One" enables refocusing by using a stereo camera and post processing algorithms, or the company "Pelican Imaging" which is working on a microarray of cameras [122], that can be built into a smartphone, making it capable of true light field photography.

Prior to the first commercially available light field cameras other practical methods have been proposed to capture light fields such as a camera array [121] or a gantry robot mounted with a DSLR camera that was used to produce the Stanford Light Field dataset [113].

Light Field parametrization

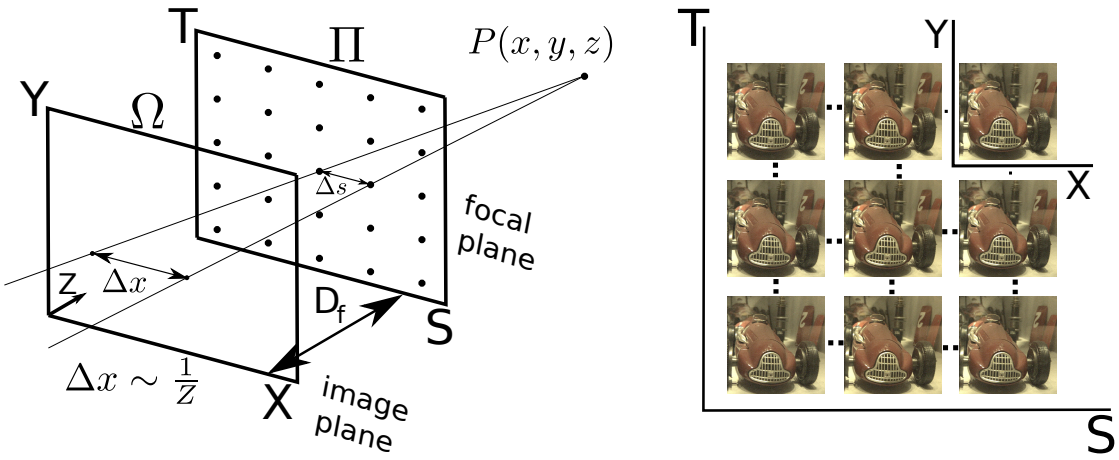
The term *light field* was introduced by [42]. The light field, aka the *plenoptic function*, is a function that specifies the amount of light and its direction passing through each point in space. According to [44] four dimensions suffice to describe the plenoptic function. In the next section the four dimensional parametrization will be explained.

The 4D light field, aka the *Lumigraph* [44], is mostly parametrized using the two plane parametrization, see Fig. 1.9. It was introduced in the seminal paper by [75]. It is a mapping

$$L : \Pi \times \Omega \to \mathbb{R}, \qquad (s, t, x, y) \mapsto L(s, t, x, y).$$

$\Omega \subset \mathbb{R}^2$ denotes the image plane and $\Pi \subset \mathbb{R}^2$ denotes the focal plane, which contains the focal points of the different virtual cameras. (x, y) is a point in the image plane, (s, t) is a point in the camera plane. In a discretely sampled light field, (x, y) can be regarded as a pixel in an image and (s, t) can be regarded as the position of the camera in the grid of cameras, see Fig. 1.9. For the algorithm described in Ch. 4 the light field was stored as a 4D object with $dim(L) = (S, T, X, Y)$.

For the discrete case, aka light field photography, it is convenient to regard the light field as a collection of images of the same scene, taken by several cameras at different positions. Each camera is placed on a cross-section of an equi-spaced $n \times n$ grid. Hence each image shows the same scene from a slightly different perspective.



(a) Two plane parametrization (b) Sub-aperture visualization

Figure 1.9: **Left:** *Two plane parametrization. The distance Δx of the projection of the 3D point $P(x, y, z)$ onto two different images is inversely proportional to the distance Z.* **Right:** *Sub-aperture visualization. The 4D light field is visualized by showing three sub-aperture images in each dimension S and T. For a fixed pair (s, t) a pixel position in a sub-aperture image is described by its (x, y) coordinates.*

Visualizations of the light field

Two descriptive visualizations of the light field arise if different parameters are fixed. If $s = s^*$ and $t = t^*$ are kept fixed, so-called *sub-aperture images* arise, see Fig. 1.9 and Fig. 1.12 for an explanation of the term "sub-aperture image". We denote them with I_{s^*,t^*}. A sub-aperture image hence is the image of one camera looking at the scene from a fixed viewpoint (s^*, t^*), and looks like a normal image. If ($y = y^*$, $t = t^*$) or ($x = x^*$, $s = s^*$) are kept fixed, so-called *epipolar plane images (EPI) epipolar plane images (EPI)* [8] arise, see Fig. 1.10. We will denote it with E_{y^*,t^*} or E_{x^*,s^*}.

A more intuitive explanation of an EPI is the following: an EPI E_{y^*,t^*} is the result of extracting the row with fixed coordinate y^* from each sub-aperture I_{s,t^*} image and stacking those rows on top of each other, see Fig. 1.10. Note that the sub-aperture images I_{s,t^*} have the same angular row t^* and vary in s. In the same way an EPI E_{x^*,s^*} is the result of stacking together columns with a fixed coordinate x^* from sub-aperture images $I_{s^*,t}$ with the same angular column s^* and varying t.

A very interesting feature of an EPI is that a point P that is visible in all sub-aperture images is mapped to a line in the EPI, see Fig. 1.10.This characteristic property has been employed for a number of tasks incl. denoising [43, 24], in-painting [43], segmentation [131, 76], matting [17], super-resolution [130, 128, 5] and depth estimation (see related work Sec. 4.1). Our depth estimation algorithm also makes use of this property.

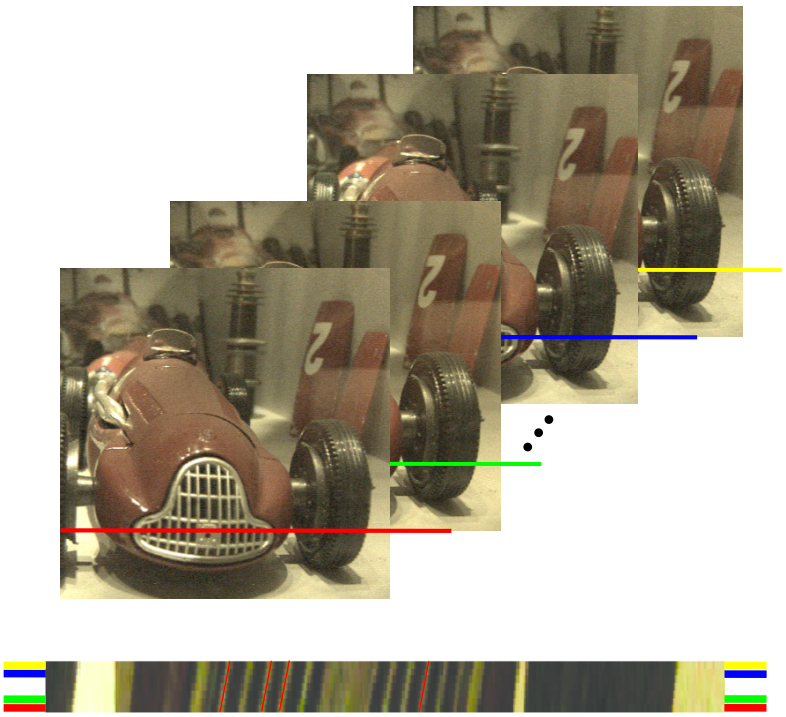
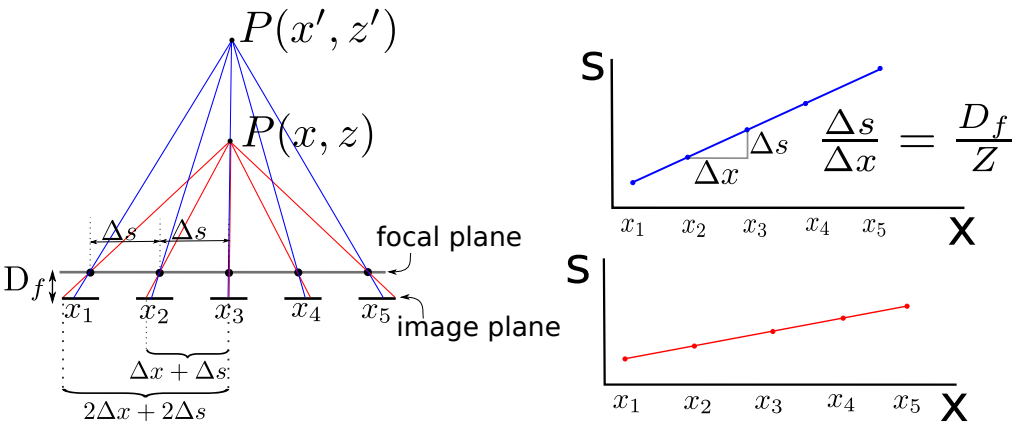


Figure 1.10: An intuitive explanation of an EPI E_{y^*,t^*}. Four sub-aperture images with the same angular row t^* are shown. From each of the images the colored row y^* is taken and stacked together, resulting in an EPI shown at the bottom. The pixels highlighted with a red line pixels are placed at the bottom, with a green line pixels are placed on top of it, etc. Note that the images shown in this figure are γ-compressed with $\gamma = 2.2$.

The emergence of lines with different slopes in an EPI is visualized in Fig. 1.11: in a 2D world (X,Z) with cameras equidistantly placed (same Δs) on a line, a point $P(x, z)$ will be projected to different locations x_i at each camera sensor. For a further away point $P(x', z')$, the difference Δx will be smaller. If the point is infinitely away ($Z = \infty$), e.g. a star, the light rays come in parallel, hence the point will be projected onto the same location at each sensor, resulting in a vertical slope in the EPI. Therefore, the slope of a line in an EPI image is directly related to its distance and hence to its depth within the scene.

Light field Camera

In this section we will explain shortly how one possible realization of a light field camera, aka *plenoptic camera 1.0*, works. This setup is used by the Lytro camera, which was the first consumer camera that is able to capture a light field. The company Lytro was



(a) Two plane parametrization (b) Resulting EPI

Figure 1.11: Two-plane parametrization in a 2D world. Applying the intercept theorem one can see that the slope of a line in the EPI (epipolar plane image) $\frac{\Delta s}{\Delta x}$ *is inversely proportional to the distance* Z*. (To be precise* $\frac{\Delta s}{\Delta x + \Delta s} = \frac{D_f}{Z}$ *in the left image. Since the origin in each sub-aperture image has coordinates* $(0|0)$*, meaning it is shifted by* Δs *already, the ratio is reduced to* $\frac{\Delta s}{\Delta x} = \frac{D_f}{Z}$ *in the right image.) A nearer point* $P(x, z)$ *leads to a flatter line in the EPI than a further away point* $P(x', z')$*.*

founded by Andrew Ng, who built a Lytro like camera during his PhD [91]. Georgiev [40] sketches a short history from the first idea of a light field capturing system by the Nobel laureate Lippmann [78] to the research by Andrew Ng [92] that enabled the first consumer light field camera Lytro.

In [41] Georgiev describes the setup of the first generation Lytro camera. It can be regarded as a normal digital camera with an additional microlens grid. This grid is placed between sensor and main lens, at a distance $f_{\text{microlens}}$ away from the sensor, with $f_{\text{microlens}}$ being the focal length of each of the lenses of the micro lens array, see Fig. 1.13. Hence each microlens is focused at infinity.

The main lens is focused on the microlens array. At each microlens the focused incoming light ray is split up into rays. Ideally each pixel would record a single light ray, coming from a different direction, which is not possible in practice. Instead the light rays, emitted by the object, are discretely sampled. Hence each pixel of the sensor records a bunch of light rays.

Note that the f-number of the microlens and the f-number of the main lens are adapted [92] to ensure that the images of the microlens optimally fit on the sensor. If the f-number of a microlens is lower, sensor pixels are not used, if it is higher, the microlens images overlap.

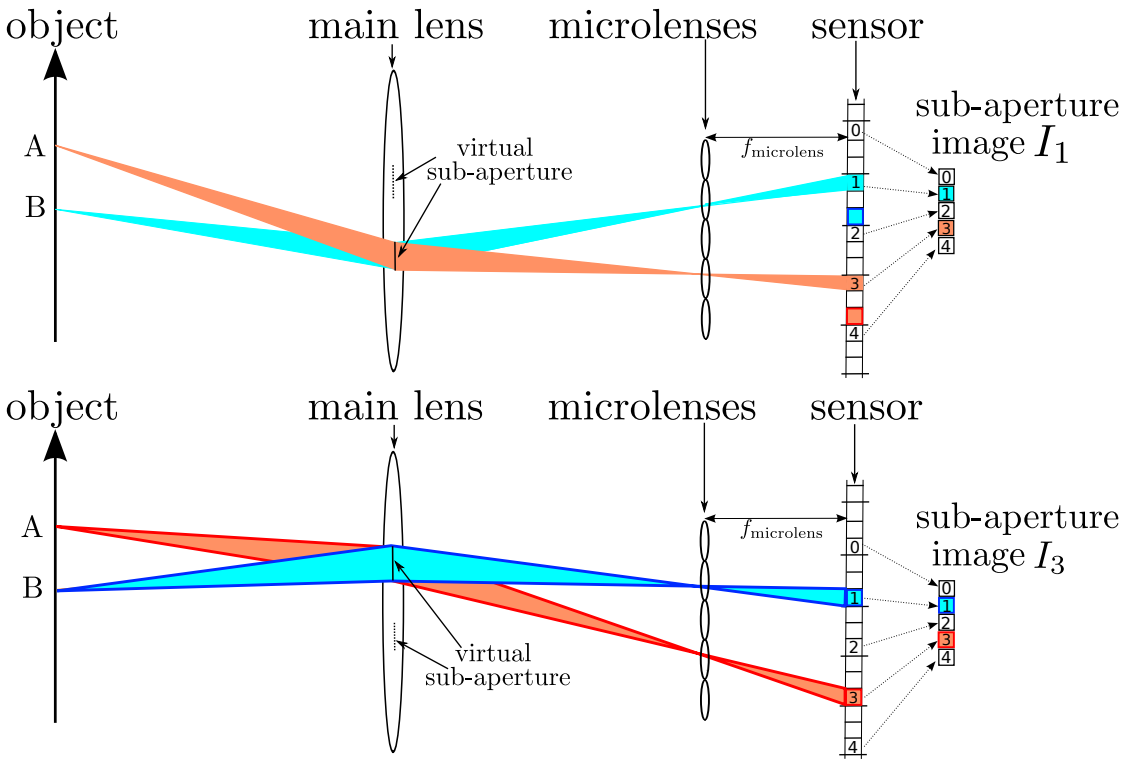


Figure 1.12: Explanation of the term "sub-aperture image": In this 2D visualization each microlens covers 3 pixels. If the main lens is partitioned into virtual sub-apertures, the rays from the whole object passing through a virtual sub-aperture are recorded at each third pixel. Each of the three sub-aperture images can be constructed by reading out each third pixel of the sensor.

The image below each microlens covers a certain area of the sensor. The larger that area the higher the angular resolution.

Fig. 1.15 shows a demosaiced raw image taken with a first generation Lytro camera. At first glance it looks like a normal image, that's due to the main lens. At a finer scale the comb like structure of the image is visible. Each comb is the image below one microlens.

The number of pixels in a microlens image determines the angular resolution. The larger a comb in Fig. 1.15 the higher the angular resolution, but the lower the spatial resolution, i.e. for a fixed sensor and pixel size the more microlenses there are in the grid of microlenses the higher the angular resolution, but the lower the spatial resolution. The extreme case for 0 microlenses would be a normal camera, having no angular resolution, but a single high resolution sub-aperture image. The other extreme having one microlens per pixel would result in sub-aperture images which are just one pixel large, but each of which having recorded light from a different direction.

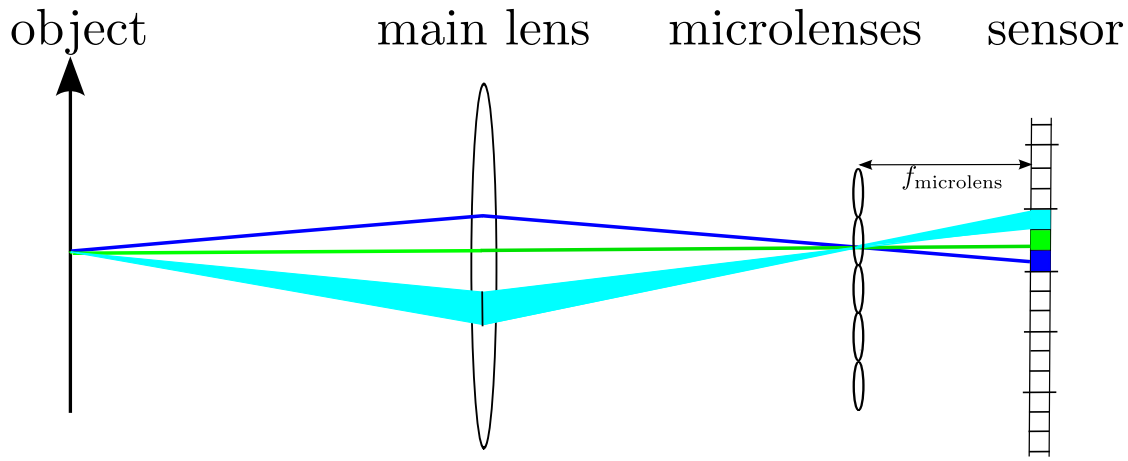


Figure 1.13: 2D visualization of a plenoptic camera 1.0 setup. The main lens focuses the rays, which are reflected by the object, onto the microlens array. There the rays get split up and each pixel of the sensor records the light rays from a different direction. Of course not only a single ray is recorded at each pixel, but a bundle of rays, visualized in the light-blue bundle.

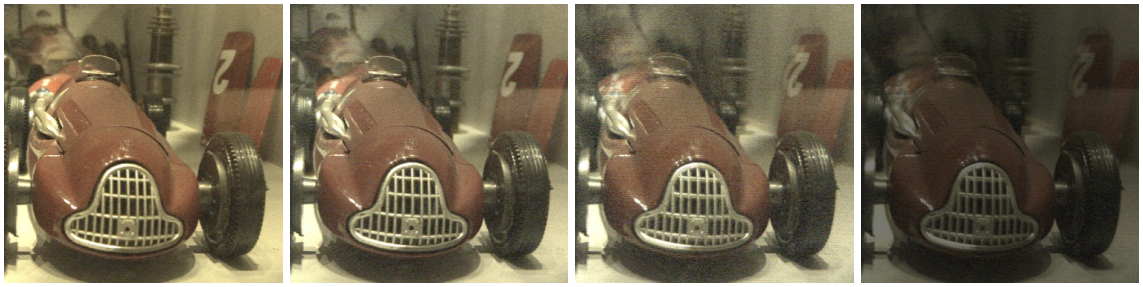
This trade-off between angular and spatial resolution is a current drawback of the plenoptic camera 1.0 design. There are some solutions to this trade-off: One obvious solution is to do superresolution [130, 128, 6, 5], as the sub-aperture images are very similar (see Fig. 1.14), if quality differences due to microlens aberrations are not taken into account. The company Lytro, for example, has its own superresolution algorithm, built into its desktop software [83] increasing the image size from about 0.1MB to about 1MB pixels.

Another solution is using different hardware: the company Sony has filed a patent [54] about a sensor with two layers of pixels that seems to be able to alleviate the low-resolution problem.

Another proposed solutions by [82] is to use a different camera setup, the so called *plenoptic camera 2.0* setup. There the microlens array is positioned in a different way. Perwas and Wietzke [100] proposed to use a microlens array with differently focused microlenses. Both approaches make it possible to render the light field in a way such that a higher spatial resolution can be obtained. The company Raytrix, founded by Perwas and Wietzke, uses their approach [100].

A note on taking pictures with the Lytro camera : as mentioned on lightfield-forum.com[5] taking images with the Lytro camera is a different task than with a usual photo camera. It is advisable having a part of the scene close to the camera (about 10-15cm). If "normal images" are taken, the advertised refocusing performance is not visible and it will also

[5] `http://bit.ly/1IQUxR4`,accessed 24-April-2015



(a) central sub-aperture image $(s = 0, t = 0)$ (b) $(s = 3, t = 3)$ (c) $(s = 4, t = 4)$ (d) $(s = 5, t = 4)$

Figure 1.14: Sub-aperture images $I_{s,t}$ from the Lytro camera. The quality of the sub-aperture images deteriorates. **(a)** *The central sub-aperture has the best visual quality. Due to lens aberrations in the microlenses further away sub-aperture images have worse quality.* **(b)** *Besides being noisier, the sub-aperture image ($s = 3, t = 3$) shows no apparent visual artifacts.* **(c) & (d)** *The number 2 on the door in the background is duplicated.* **(d)** *The image intensity is lower than for the central sub-aperture image. Note that the four images shown here were γ-compressed with $\gamma = 2.2$.*

be not possible to infer any reasonable depth map from the Lytro image. If publicly available Lytro images[6] are examined, it is possible to almost always notice a very close foreground object.

Reading out sub-aperture images from the raw file

Fig. 1.12 explains how sub-aperture images can be read out from the raw image. Each microlens splits the focused light beam into bundles of light rays. In Fig. 1.12 there are three pixels under each microlens. The angular resolution would be 3×1. Each of those three pixels records another direction of the incoming light. To create the sub-aperture image I_i it is necessary to read out every third pixel, starting with the i-th pixel.

For the First Generation Lytro camera the toolbox by [26] returns an $11 \times 11 \times 375 \times 375 \times 3$ dimensional light field.[7] Due to lens aberrations, which is the weakest in the center of the lens, the quality of sub-aperture images further away from the center sub-aperture image gets worse. Hence for computing depth maps from Lytro light field images, see Ch. 4, we decided to only use the central 7×7 sub-aperture images, as outer sub-aperture images show major artifacts, see Fig. 1.14.

[6] e.g. `https://pictures.lytro.com/lytro/albums/4`, accessed 24-April-2015

[7] To be precise it returns a $11 \times 11 \times 377 \times 377 \times 3$ light field. But we crop the first and last row and column, as they show pixel artifacts.

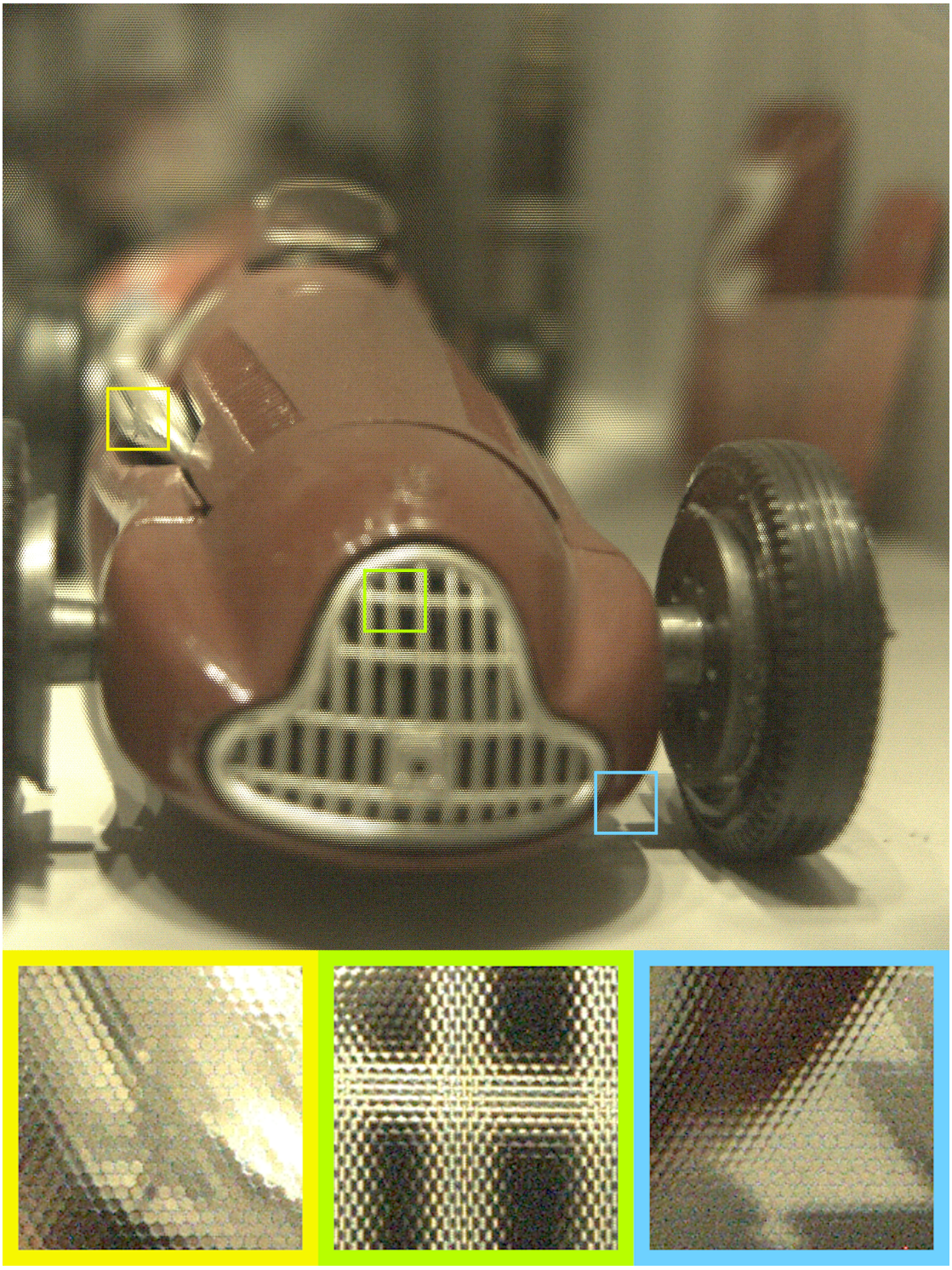


Figure 1.15: Lenslet image of the Lytro camera, extracted using the toolbox by [26]. Debayering of the raw image was done with Matlab's build in function "demosaic". The image was gamma corrected with $\gamma = 2.2$. The comb like structure is visible in the close-ups. Each comb is the image below one microlens.

2

Benchmarking blind deconvolution algorithms

Chapter abstract Motion blur due to camera shake is one of the predominant sources of degradation in handheld photography. Single image blind deconvolution (BD) or motion deblurring aims at restoring a sharp latent image from the blurred recorded picture without knowing the camera motion that took place during the exposure. The only input is the single blurred image. BD is a long-standing problem, but has attracted much attention recently, cumulating in several algorithms able to restore photos degraded by real camera motion in high quality. In this chapter, we present a *benchmark dataset* for motion deblurring that allows quantitative performance evaluation and comparison of recent approaches featuring non-uniform blur models. To this end, we *record and analyse real camera motion*, which is played back on a robot platform such that we can record a sequence of sharp images sampling the six dimensional camera motion trajectory. The goal of deblurring is to recover one of these sharp images, and our dataset contains all information to assess how closely various algorithms approximate that goal. In a comprehensive comparison, we evaluate state-of-the-art single image BD algorithms incorporating uniform and non-uniform blur models.

This chapter is based on the following publication:

[64] R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf and S. Harmeling. Recording and Playback of Camera Shake: Benchmarking Blind Deconvolution with a Real-World Database. European Conference on Computer Vision (ECCV) 2012.

2.1 Introduction and Contributions

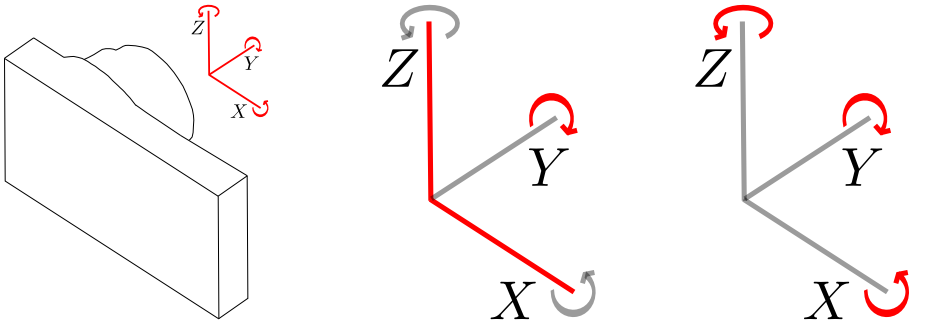
Camera motion during exposure is a major problem in handheld photography, as it causes image blur that destroys details in the recorded photo. Especially photos taken in low light without flash suffer from motion blur due to the necessity of longer exposure times. Recently, single image blind deconvolution (BD) has attracted significant attention. Considerable progress was made not only by conceiving more efficient inference strategies but also by proposing more realistic imaging models, better able to capture real camera shake. Levin at al. [73] put together a benchmark dataset for the case of uniform (stationary) blur, allowing objective evaluation and comparison of single image BD algorithms. The lack of a proper benchmark for the case of non-uniform blur, however, makes it difficult to evaluate the performance of recent blind deblurring methods that feature non-uniform blur models and go beyond the traditional invariant convolution model.

To close this gap, this chapter presents a new benchmark dataset that is not restricted to translational motion, but mainly consists of example images with non-uniform blur originating from real camera trajectories. To this end we designed an experimental setup that allows the recording of unconstrained camera motion with full six-dimensional degree of freedom at sub-millimeter precision. Furthermore, we recorded and analysed real camera shake by humans who were asked to take photos with relatively long exposure times. We confirm the finding of Levin et al. [73] that real camera motion does indeed often involve rotational motion, rendering image blur non-uniform across the image plane, however, we also find that the amount of spatial variance of the point spread function (PSF) is often small. In this context, we also investigate and validate recently proposed imaging models [134, 47], which consider only three (out of a possible six) degrees of freedom to capture real camera motion.

For playing back real camera shake, we use a Stewart platform (Hexapod) with six degrees of freedom, featuring repeatability at micrometer accuracy. This allows the recording of a sequence of sharp ground truth images, taken along the played-back camera trajectory.

In a comprehensive comparison, we evaluate state-of-the-art single image BD algorithms with both uniform and non-uniform blur models.

Before describing the benchmarked algorithms in greater detail, we will give a short introduction to image deblurring, give an overview of blind deconvolution (BD) algorithms and introduce the benchmarked algorithms in greater detail.



(a) camera and the 6D space (b) approximation by Hirsch et al. [56] (c) approximation by Whyte et al. [134]

Figure 2.1: **(a)** *Visualization of the six dimensions needed to record a camera trajectory (translation along the three axis and rotation about each axis).* **(b)** *Approximation of the six dimensions by one rotation and two translations (shown in red) as used by Hirsch et al. [56].* **(c)** *Approximation by three rotations as used by Whyte et al. [134].*

Contributions of this chapter:

1. A new setup for recording and playing back camera shake.
2. An extensible benchmark dataset with ground truth data.
3. A comprehensive comparison of state-of-the-art single image BD algorithms.

2.2 Recording trajectories of human camera shake

The motion of a camera can be parametrized by its six dimensional trajectory in space, with three translational and three rotational coordinates changing through time, see Fig. 2.1. Our subjects were holding a compact camera, the Samsung WB600. Compact cameras are widely used cameras, as they are easy to carry around. Most models usually only allow for taking images with a small aperture and photographs hence get easily blurred. In low light situations where the use of a flash is not desired, the incoming light with a small aperture is not sufficient and the exposure time has to be increased.

For all our experiments we used an exposure time of 1/3 sec, a realistic value for a low light scenario.

The camera trajectory was recorded with a Vicon tracking system with 16 high-speed Vicon MX-13 cameras running at a frame rate of 500 Hz. The cameras are mounted at about 4m from the ground on the walls of a hall with an approximate size of 11.13m $\times$ 12.60m $\times$ 8.4m (length $\times$ width $\times$ height). As a small cube was sufficient for

the motion recording experiment, we calibrated the Vicon cameras to a cube of roughly the size 2.5m × 2.5m × 2.5m in the center of the hall, which all 16 cameras could see.

The error of calibration was less than 0.15 pixel for each Vicon camera. This error cannot be translated easily into an error in measurement, as it depends on the distance of the object to the camera. Furthermore each camera had a different pixel error and in the end the signals of the 16 cameras are combined. Therefore we measured the error of the measurement support frame, see the left image of Fig. 2.2, standing still on the floor, in the middle of the calibrated cube. For a frame rate of 500 Hz, we got a mean of $\mu_{(\theta_x,\theta_y,\theta_z,x,y,z)} = (-0.0065°, -0.0034°, -0.0065°, -0.1610mm, -0.0180mm, -0.1071mm)$ and a standard deviation of $(0.0086°, 0.0058°, 0.0076°, 0.1067mm, 0.0484mm, 0.1115mm)$.

To locate the exact position of the camera, a light-weight but rigid gadget with attached markers, i.e. reflective balls, was used, see Fig. 2.2. The Vicon cameras send out near infrared light and records the rays as they are reflected by the markers. We used the usual markers that are normally shipped with the Vicon tracking system. We chose the largest available markers with a diameter of 35mm.[1] The larger a marker the more pixels it fills in the field of view of each tracking camera and the more exact the position of the marker can be determined.

For synchronizing the camera shutter with the trajectory data the flash was used. The flash light had to be converted into infrared light first, as the Vicon cameras are only sensitive to the infrared spectrum. Therefore a photodiode was attached to the flash. In the moment in which the flash discharges, the photodiode sends out a current, which lits an infrared lamp for about 0.07sec. This extra point can be seen in the recorded trajectory data with the exact time stamp of its appearance.

We asked six subjects to take photographs in a natural way. Each of the subjects was aware that the exposure time was set to 1/3 sec. In total 40 trajectories were recorded with this setup (three subjects with ten trajectories, three subjects with five trajectories; five trajectories had to be excluded due to temporary recording problems).

In the next section, we will describe how the recorded camera trajectories were used to build a benchmark dataset of blurry images.

2.3 Playing camera shake on a picture-taking robot

The raw recorded trajectory data was noisy in frequency ranges, that cannot be due to human motion. To reduce the noise, we used a moving average filter of size 41, where the filtered output $y(t)$ at time t consists of the average of the 20 data points $x(s)$ before

[1] For comparison: Since the year 2000 the diameter of an official table tennis ball is 40mm.

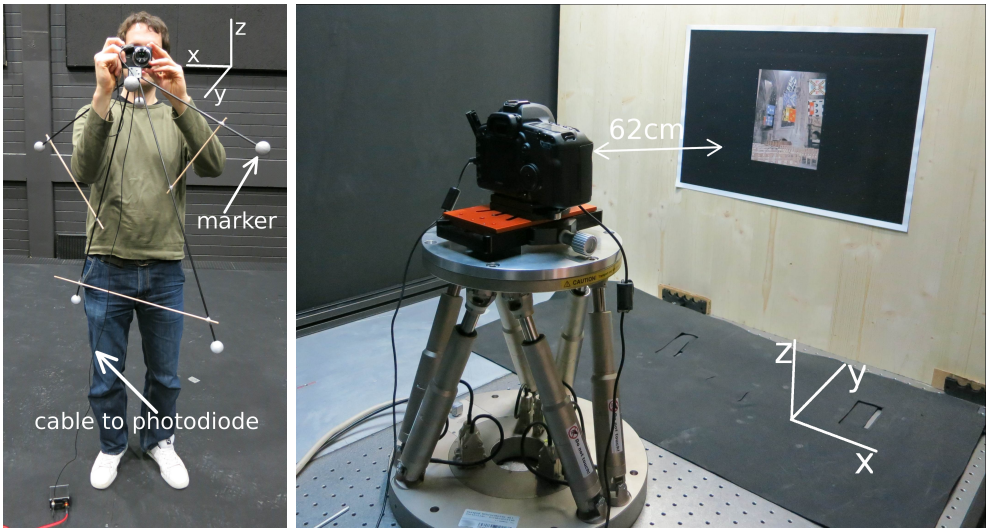


Figure 2.2: **(Left)** *Setup for recording the camera shake with a Vicon system: Lightweight gadget with attached reflective ball-shaped markers.* **(Right)** *Setup for taking blurred photographies using the recorded camera shake: A DSLR camera mounted on a high-precision hexapod robot.*

and after time t.

$$y(t) = \frac{1}{41} \sum x(t-20) + \ldots + x(t) + \ldots + x(t+20)$$

The filtered output is of the same size as the unfiltered, as we used additional 2 x 20 data points at each of the two ends of the recorded trajectory. In total, a trajectory of 1/3 sec exposure time, recorded at a frame rate of 500 Hz, contains 167 frames. Fig. 2.3 shows one smoothed camera trajectory for the rotation and translation.

The filtered trajectories were then played back on a Stewart platform. We used the model PI M-840.5PD, a very precise, high-speed hexapod for small motions. This robot has (according to the manufacturer's specification) following translational travel range from its zero position: ±50mm (x and y axis), ±25mm (z axis) and following rotational travel range: ±15° (x and y axis) and ±30° (z axis). The minimal incremental motion is: $3\mu m$ (x and y axis), $1\mu m$ (z axis) and 5 μrad (rotations) with a repeatability of: $\pm 2\mu m$ (x and y axis), $\pm 1\mu m$ (z axis) and $\pm 20\mu rad$ (rotations). The maximal velocity is: 50 mm/s for translation along the (x, y, z) axis and 600 mrad/s for the rotation.

A DSLR camera (Canon Eos 5D Mark II) that can be remote controlled was mounted to the Stewart platform to allow synchronization of the camera trigger with the platform (see Fig. 2.2 for the complete setup). For the benchmark dataset (detailed in the following) a printed image was mounted at a distance of 62cm from the camera. A printed image was used, as all benchmarked algorithms assume that the scene is flat.

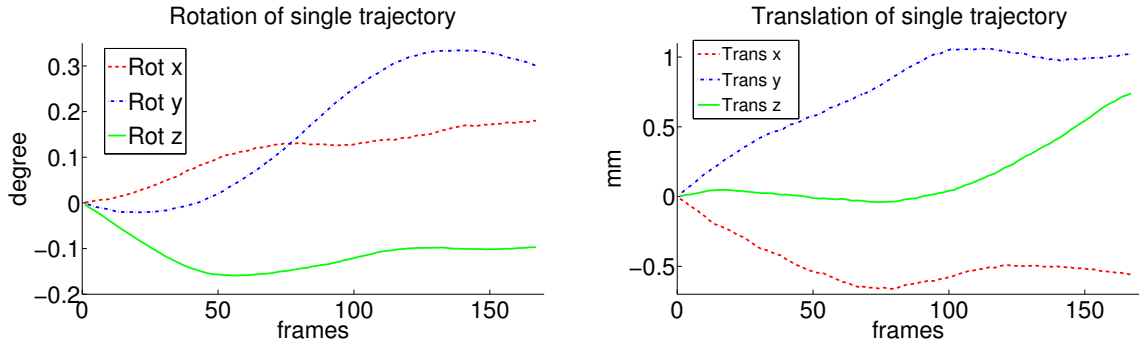


Figure 2.3: Single trajectory of the recorded and smoothed 6D camera shake: **(left)** *of the rotation angles and* **(right)** *translations. Fig. 2.4 displays the corresponding PSF.*

To qualitatively assess whether a real camera shake movement can be recorded and played back by our setup, we took long exposure images of a point grid of LEDs with a somewhat exaggerated camera shake (to obtain large visible blurs). Simultaneously, we recorded the six dimensional camera trajectory. The recorded trajectory was played back on the Stewart platform and the attached camera recorded a long exposure of the same point grid. Fig. 2.5 shows that the real camera shake (left images) is correctly simulated by the Stewart platform (right images). The existing difference in the images is caused by the fact that different cameras were used for recording and playing back the trajectory and that the distance camera to point-grid could not guaranteed to be exactly 62cm at recording. Nonetheless, in both examples (upper and lower images) the true image blurred by camera shake and the image generated by playing back the camera trajectory are reasonably similar. This shows that our setup is able to capture the movement of real camera shake and to play it back again.

Note that the playback on the Stewart platform was performed in slower motion (duration 1sec instead of 1/3sec) to increase the smoothness of the movement — stretching time does not change the obtained PSF.

Note that we decided against using the point grid of LEDs during the recording of the human camera trajectories for the benchmark. We found that the small motions of real camera shake (where the photographer tries to keep the camera steady) can not be seen well with the LED point grid. The grid would have to be positioned rather close to the camera, resulting in an unnatural feeling of tightness, normally not present while taking a photograph.

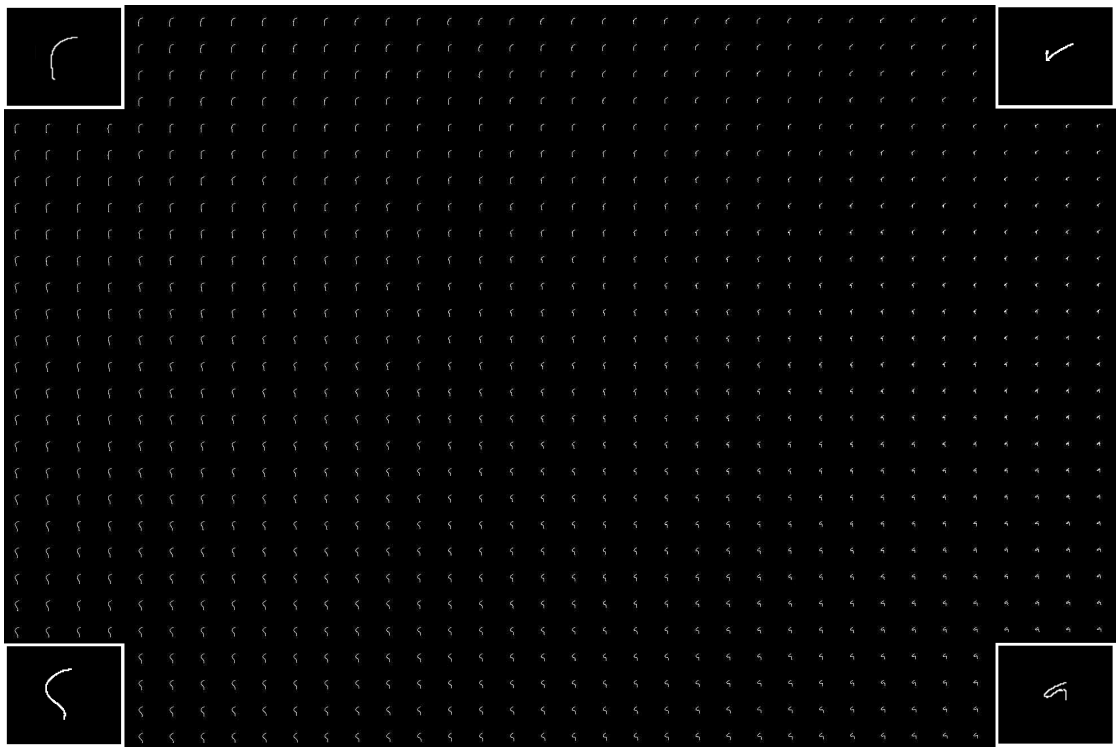


Figure 2.4: Example of a point spread function (PSF) which was generated with the 6D trajectory shown in Fig. 2.3.

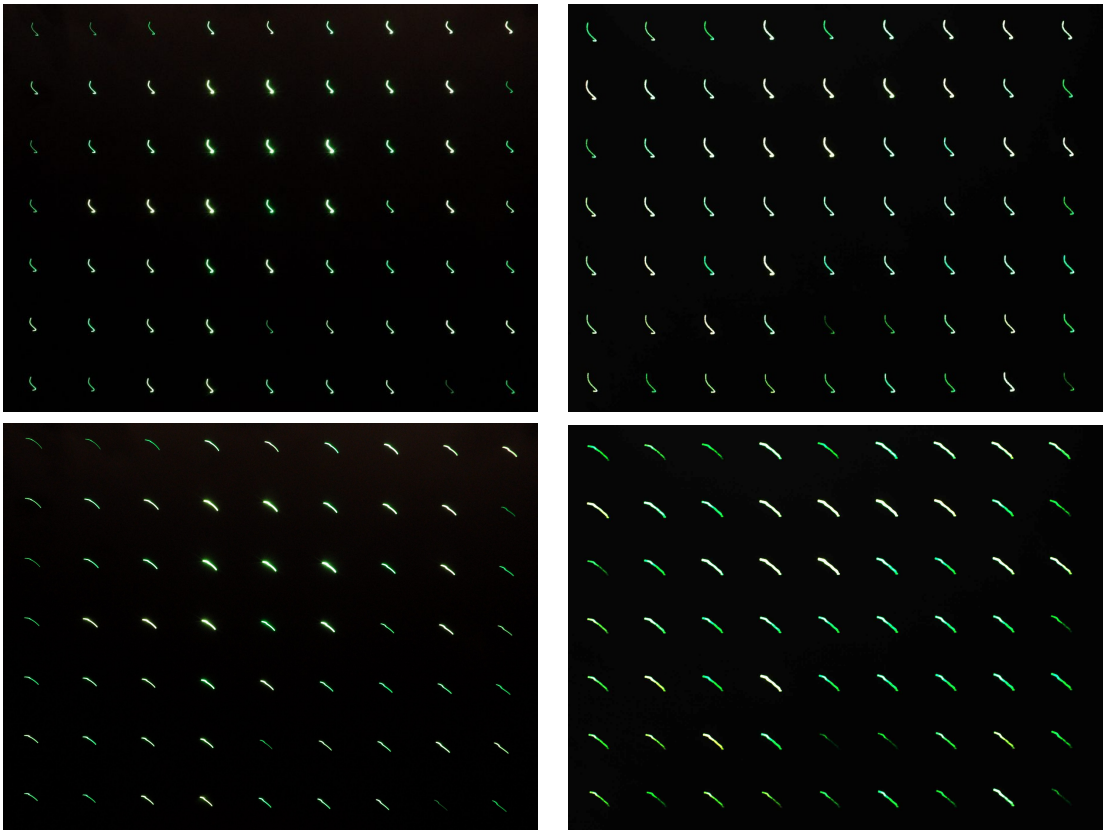
2.4 Benchmark dataset for real-world camera shakes

2.4.1 Recording images with played back camera shake

As described in Section 2.3 the image blurs created by the Stewart platform are good approximations to real image blurs. So we create a benchmark dataset of images blurred by camera shake by playing back human camera shakes on the hexapod for different images, see Fig. 2.6.

We randomly selected two blur trajectories of each of the six subjects applying them to four images (Fig. 2.6) resulting in 48 blurry images. Some local blur kernels turned out rather large. We decided to leave those blur kernels in the benchmark, as they reflect natural camera shake, probably caused by holding the camera too relaxed and not paying attention to hold it still, as could happen in real imaging situations. Even though such images are often erased, it is interesting whether current or future deblurring algorithms can cope with such large blur kernel sizes.

As discussed in Section 2.5.1 some of the blurs are approximately uniform and some are



(a) recorded image (b) played back

Figure 2.5: Two examples (first and second row) of an image with camera shake and its counterpart generated by playing back the camera trajectory on a Stewart platform.

not. Eyeballing the generated PSFs (they are visualized on the project website [65]), the blur kernels numbered 1,3,4,5,8,9,11 (see Table 6.1) tend to be approximately uniform, while blur kernels 2,6,7,10,12 appear non-uniform, in accordance with the NU criterion, defined in Sec. 2.5. Kernels 8 and 11, although having a high NU value still visually appear to be uniform. The non-uniformness detected by the NU value is not clearly visible due to the large size of the kernels.

For recording the image database, the Steward platform was placed inside a light-tight box. The SLR camera was set to ISO 100, aperture f/9.0, exposure time 1sec, taking images in the Canon raw format SRAW2. A Canon EF 50mm f/1.4 lens was used. The illumination inside the box was adjusted to the camera parameters and kept constant for all images taken. The true sharp image was mounted as a poster at a distance of 62cm to the camera. The complete setup is shown in the right image of Fig. 2.2.

church clock backyard roof

Figure 2.6: The four original images used in the benchmark.

The recorded SRAW2 images (16bit) were processed by DCRAW[2], generating 8bit images without correcting gamma. From the center of the image, a 800×800 patch was cropped. The blurry images are named (i, j) for the i-th image blurred with the j-th camera trajectory.

2.4.2 Recording ground truth images

Even though the original images are available since we printed them onto posters, comparing these images with deblurred images from some algorithm is not easy, because of different image resolution, different lighting, and possible color differences due to the printing process. Instead our robotic setup allows us to generate ground truth images *along* the trajectory by playing the camera trajectory step by step and taking an image per step.

Using this strategy, we recorded for each of the 48 blurry images (12 camera shakes each blurring 4 images) 167 images along the trajectory. Additionally, we created 30 images at intermediate positions calculated by the hexapod during playback. We denote the ground truth images by $u_1^*, \ldots, u_N^*$.

2.5 Analyzing camera shake trajectories

2.5.1 Is camera shake stationary or non-stationary?

Often surprisingly, blind deconvolution algorithms for stationary blur work quite well on real world camera shake, even though theoretically, the resulting blur should be non-stationary, i.e. varying across the image. To study this effect with our trajectories we generate a blurred point grid for a given camera trajectory, see Fig. 2.4 for an example.

[2]DCRAW was called with parameters -W -g 1 1 -r 1.802933 1.0 2.050412 1.0 -b 4.

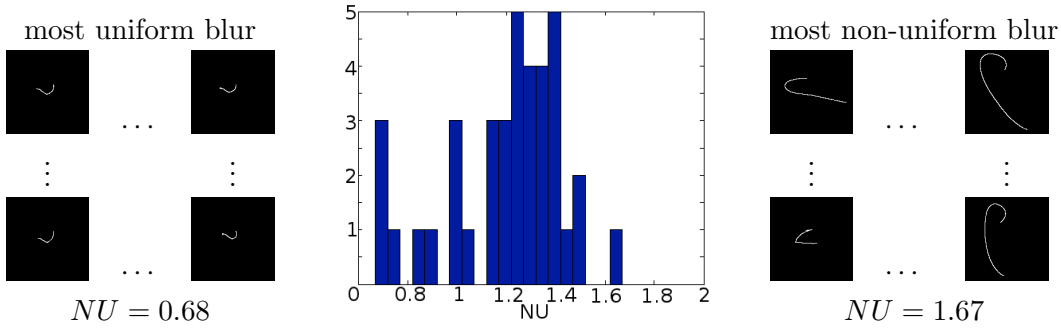


Figure 2.7: PSF images of recorded camera trajectories being most uniform (left) and most non-uniform (right) according to the NU criterion. Histogram of all NU values (middle).

We denote the camera trajectory by a point sequence $p_1, \ldots, p_T$ where each point has six dimensions,

$$p_t = [\theta_x, \theta_y, \theta_z, x, y, z]^{\mathsf{T}} \tag{2.1}$$

with the last three coordinates being the shifts in mm, the axis positioned indicated in Fig. 2.2, the first three coordinates being the rotation angle around the indicated axis.

Given an image u showing a point grid (14×10 equispaced points, size 2808×1872), we can visualize the point spread function (PSF) of the camera trajectory by converting each point p_t in time into its corresponding homography H_t (assuming distance of 2m for the point grid), assigning p_1 the identity transformation. Then the resulting blurry image v is the superposition of the transformed point grids:

$$v = \sum_{t=1}^{T} h_t(u) \tag{2.2}$$

where $h_t(u)$ is the image u transformed by H_t. The local blur kernels of the resulting PSF ξ can be read off the blurry image v. To quantify the non-uniformness (abbreviated NU) of the PSF ξ, we introduce the following measure: given four local blur kernels $\xi_{lr}, \xi_{ur}, \xi_{ll}, \xi_{ul}$ of the lower right, upper right, lower left and upper left corner, we can calculate how similar they are by comparing the lower left with the upper right and the upper left with the lower right blur kernel:

$$\mathrm{NU}(\xi) = \frac{\|\xi_{ll} - \xi_{ur}\|^2 + \|\xi_{ul} - \xi_{lr}\|^2}{2}, \tag{2.3}$$

where each blur kernel is normalized to have L2-norm one. Note that the difference between two blur kernels has to be minimized over all possible shifts (omitted in the

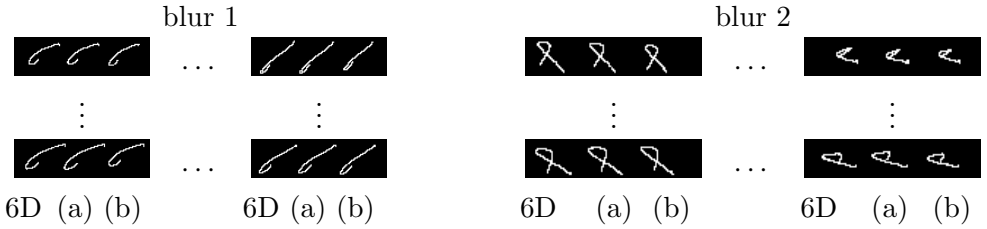


Figure 2.8: Mapping the 6D camera trajectory to 3D using (a) Whyte et al.'s [134] approach and (b) Hirsch et al.'s [56] approach. Each image patch compares the blur due to the original 6D representation to (a) and (b). We set the focal length to 50mm and the object distance to 2m.

formula for clarity). This can be achieved efficiently by noting that $\|\xi_1 - \xi_2\|^2 = \xi_1^\mathsf{T}\xi_1 + \xi_2^\mathsf{T}\xi_2 - 2\xi_1^\mathsf{T}\xi_2$, where the first summands are one due to normalization and the inner products for different shifts are the cross correlations between ξ_1 and ξ_2. Minimizing $\|\xi_1 - \xi_2\|^2$ for all possible shifts is thus done by maximizing the cross-correlation between ξ_1 and ξ_2.

Note that for a perfectly uniform blur (pure shift trajectory along x and z axes), i.e. a camera that has no rotations and no y component, the non-uniformness index NU is equal to zero. Fig. 2.7 shows the extreme corners of two PSF images generated from recorded camera trajectories. The left example is the most uniform blur according to the NU criterion, the right example is the most non-uniform blur. Also qualitatively, we observe that the NU criterion captures non-uniformness. In the middle is the histogram of the NU values of all 40 PSFs.

2.5.2 How well can we approximate the 6D trajectory with 3D?

To decrease the complexity of possible PSFs, the six dimensional real camera trajectory is often approximated with three dimensions. There are two main approaches:

(a) ignore the shifts and model only three rotations (θ_x, θ_y, θ_z) [134].

(b) model only shifts (x and z) and rotation around the visual axis (θ_y) [47, 56],

Both approximations achieve good results, as can be seen in Fig. 2.8, so we would like to analyze whether this is in correspondence with our recorded trajectories. To this end, we transform the trajectories to three dimensions following the two approaches. Note that the influence of the angles on the shifts and vice versa depends on the distance d

of the object to the camera:

$$\text{(a) } p_t = \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \\ x \\ y \\ z \end{bmatrix} \mapsto \begin{bmatrix} 0 \\ \theta_y \\ 0 \\ x - d\sin(\theta_z) \\ 0 \\ z + d\sin(\theta_x) \end{bmatrix} \qquad \text{(b) } p_t = \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \\ x \\ y \\ z \end{bmatrix} \mapsto \begin{bmatrix} \theta_x - \arcsin(x/d) \\ \theta_y \\ \theta_z + \arcsin(z/d) \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Assuming an object distance of $d = 2m$ we transform all 40 camera trajectories to three dimensions using mapping (a) and mapping (b). Fig. 2.8 shows, based on two examples, that both 3D transformations are qualitatively valid. For future research it might be interesting whether there exists another *canonical* transformation from 6D to 3D which preserves the PSFs.

Before evaluating the results of the benchmarked algorithms, we will discuss several image quality metrics, which we used to measure the performance of the various algorithms

2.6 Full reference image quality metrics

A metric is needed to compare the improvement in image quality among the various algorithms. This is not a trivial task and there is a whole scientific community developing metrics that are able to judge the image quality of degraded images, mostly due to compression artifacts (like jpeg) or due to noise.

Every developer of a computational imaging algorithm needs to evaluate the performance of his algorithm, which differs for different design choices. Different artifacts are introduced by different design choices (e.g. oversmoothing with a Gauss Prior, but less noise) and it should be known what choices lead to an image quality that is regarded as the best by the majority of humans.

A good image metric would highly correlated with the evaluation of the majority of test candidates. In this thesis we will only use so-called *full reference image quality metrics (FR-IQM).*

FR-IQM are metrics to evaluate the quality of a degraded image (e.g. through compression artifacts or noise) compared to the original undistorted ground truth image. *No-reference image quality metrics* in contrast try to evaluate the quality of a degraded image without the knowledge of the ground truth image.

In the following sections, we introduce four FR-IQMs, which we used for the evaluation of the deblurring algorithms. For surveys on image quality metrics, see [77, 32, 96].

	JP2K#1	JP2K#2	JPEG#1	JPEG#2	WN	GBlur	FF	All data
PSNR	0.9332	0.8740	0.8856	0.9167	0.9859	0.7834	0.8895	0.8709
MS-SSIM	0.9702	0.9711	0.9699	0.9879	0.9737	0.9487	0.9304	0.9393
IFC	0.9421	0.9626	0.9209	0.9744	0.9766	0.9691	0.9631	0.9441
VIF	0.9791	0.9787	0.9714	0.9885	0.9877	0.9762	0.9704	0.9533

Table 2.1: Table taken from [112]. Linear correlation coefficient after nonlinear regression between scores of the four chosen image quality metrics and perceived image quality scores by human subjects. The correlation coefficients are shown for different image distortion types and for the combined data.

We decided to use the widely used metric PSNR and besides that, the three FR-IQMs MS-SSIM [127], VIF [110] and IFC [111], as they showed a high correlation with perceived image quality by humans as shown in the paper by [112]. Table 2.1 summarize the linear correlation coefficient of those four metrices compared with an assigned human score:

In the work by [112] around 111-174 images[3] were distorted each with different distortion types, (JPEG and JPEG2000 of different compression rates, added white noise (WN), Gaussian blur (GBlur) and transmission errors (bit errors) due to a fast-fading wireless channel (FF)). About 20-29 subjects[3] were asked to evaluate the image quality of the distorted test image compared to the undistorted reference image. From that subjective evaluations a score was computed, called DMOS score, for each distorted image, representing the average perceived image quality of the distorted image by the human subjects. Each FR-IQM score was mapped to this DMOS score via a non-linear five-parameter logistic function

$$\begin{aligned} \text{Quality}(x) &= \beta_1 \text{logistic}(\beta_2, (x - \beta_3)) + \beta_4 x + \beta_5 \\ \text{logistic}(\tau, x) &= \frac{1}{2} - \frac{1}{1 + \exp(\tau x)}, \end{aligned}$$

with x being the score of a chosen IQM. The parameters for β_i were determined using regression.

In the following sections, we will shortly present each of the four FR-IQMs.

2.6.1 PSNR

A widely used metric is the *peak signal-to-noise ratio* (PSNR). Given two images of equal size $X \times Y$ and equal maximum signal extent $I_{MAX} = I_{1,MAX} = I_{2,MAX}$, [4] PSNR is

[3] Exact number depended on the distortion type.

[4] $I_{MAX} = 2^8 - 1 = 255$ for 8-bit images, $I_{MAX} = 2^{16} - 1$ for 16-bit images.

defined (e.g. by [115]) as

$$PSNR(I_1, I_2) = 10 \log_{10} \frac{I_{MAX}^2}{MSE(I_1, I_2)}$$

where MSE is the *mean squared error*, defined as the mean of squared pixel differences

$$MSE(I_1, I_2) = \frac{1}{XY} \sum_p \left(I_1(p) - I_2(p)\right)^2 .$$

PSNR is actually almost only the sum of squared pixel differences between two given images I_1, I_2. To account for different image sizes, the mean of the sum is considered and to account for different signal extents, the maximum signal extent is also considered. To shift it in easier to use range the transformation by $10 \log_{10}$ is used. The higher the PSNR score, the more similar the distorted and the ground truth reference image are. If $I_1 = I_2$ PSNR is defined to be ∞.

Despite being simple to understand and widely used, it is not clear if the PSNR is a good metric to evaluate the quality of an image compared to a reference image. A whole paper [125] is dedicated to the deficiencies of the PSNR metric. Fig 2.9 shows images taken from that paper, which all have about the same PSNR value, but differ a lot in perceived image quality.

2.6.2 Multi-scale SSIM

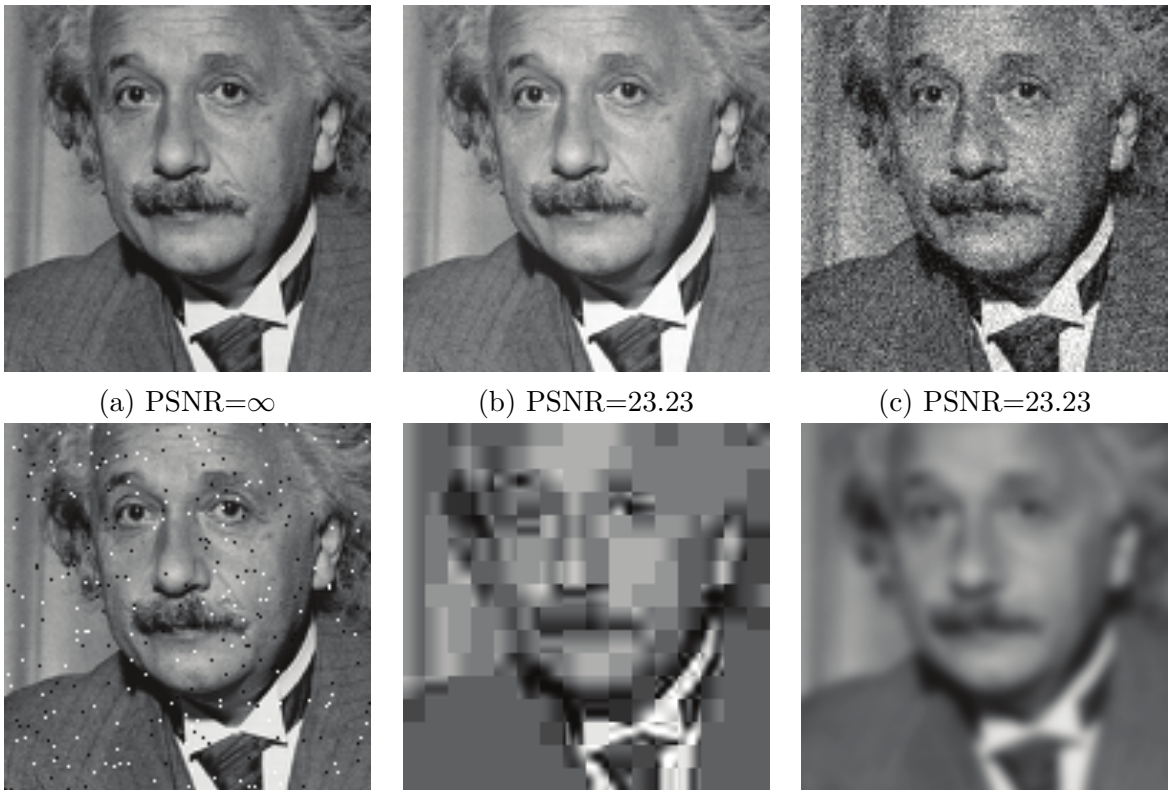
The *Multi-scale SSIM* was introduced by [127] and is a multi-scale version of the SSIM index [126]. The underlying assumption is that the human visual system is very sensitive to structural information and hence the difference in structural information would be a good way to quantify image quality.

It tries to evaluate the damage to an image combining luminance-, contrast- and structure-comparison-measures on multi-scale windows. It is defined as

$$\text{MS-SSIM}(I_1, I_2) = [l_M(I_1, I_2)]^{\alpha_M} \cdot \prod_{j=1}^{M} [c_j(I_1, I_2)]^{\beta_j} \cdot [s_j(I_1, I_2)]^{\gamma_j} .$$

For easier notation we explain the three functions $l(\cdot), c(\cdot), s(\cdot)$ without its indices, assuming for now $M = 1$.

$l(I_1, I_2) = \frac{2\mu_{I_1}\mu_{I_2}+C_1}{\mu_{I_1}^2+\mu_{I_2}^2+C_1}$ is the luminance comparison function, measuring the intensity difference in the mean intensities μ_{I_i}.



(a) PSNR=∞ (b) PSNR=23.23 (c) PSNR=23.23

(d) PSNR=23.18 (e) PSNR=23.23 PSNR=23.25

Figure 2.9: The images are taken from [125]. Despite having almost identical PSNR values, the visual quality of the images differ a lot. (a) reference image, the other images were distorted by (b) Luminance shift, (c) Gaussian noise, (d) Salt and Pepper noise, (e) Jpeg compression and (f) Blurring.

$c(I_1, I_2) = \frac{2\sigma_{I_1}\sigma_{I_2}+C_2}{\sigma_{I_1}^2+\sigma_{I_2}^2+C_2}$ is the contrast comparison function, measuring the difference in the contrast by using the standard deviation of the intensity values σ_{I_i}.

$s(I_1, I_2) = \frac{\sigma_{I_1 I_2}+C_3}{\sigma_{I_1}\sigma_{I_2}+C_3}$ is the structure comparison function, measuring the correlation between I_1 and I_2 using the correlation coefficient $\sigma_{I_1 I_2}$.

The constants $C_1 > 0, C_2 > 0$ and $C_3 > 0$ are introduced to prevent the denominator to be close to 0.

The reference and distorted images are filtered with a low-pass filter and downsampled by a factor of 2. This is done M times, with M being the last iteration. Note that the luminance component l_M is only computed at the coarsest scale M. The MS-SSIM not only compares two images I_1 and I_2 on one scale but on multiple scales. In the paper, the authors set $M = 5$ and propose possible values for α_j, β_j and γ_j, which are deduced

from an experiment done with 8 subjects on 10 sets of test images. For evaluating the performance of the deblurring algorithms, we also set $M = 5$ and the proposed values for α_j, β_j and γ_j.

MS-SSIM ranges in $[0, 1]$. It is exactly 1 if two images are identical.

2.6.3 IFC

The *IFC* metric was proposed by [111]. It is derived in the wavelet domain and based on natural scene statistics. The underlying distortion model describes the distorted image D and the reference image C as Gaussian scale mixtures random fields in the wavelet domain. The distortion model in each subband is given by

$$\mathcal{D} = \mathcal{GC} + \mathcal{V} = \{g_i C_i + V_i : i \in I\}$$

with I being the set of spatial indices for the RF, $\mathcal{C}$ being the RF from one subband in the reference image and $\mathcal{D}$ in the distorted image. $\mathcal{G}$ is a scalar attenuation field and $\mathcal{V}$ is a zero mean Gaussian noise RF with variance σ_V^2.

The IFC for one subband k is defined as the mutual information $I(C^{N,k}, D^{N,k})$ between the k-th subband of the model of the reference image $C^{N,k}$ and the distorted image $D^{N,k}$, with $C^{N,k} = (C_1, \ldots, C_N)$ denoting N elements from $\mathcal{C}$ for the fixed subband k.

The overall IFC is the summation over all subbands

$$IFC = \sum_{k \in \text{subbands}} I(C^{N,k}, D^{N,k}) \tag{2.4}$$

In the paper by [111] a numerical expression for the IFC is derived based on equation Eq. (2.4).

The IFC metric takes values in $(0, \infty)$ (as the PSNR metric), where the value ∞ is assigned if the two images are identical. The lower the VIF value the more different the two images are.

2.6.4 VIF

The VIF image quality metric, proposed by [110] is similar to the IFC metric. It incorporates a Human Visual System (HVS) model in the wavelet domain. The HVS is regarded as a "distortion channel", which is not capable of extracting all the information

from the given reference image $\mathcal{C}$ or $\mathcal{D}$,[5] but adds noise to the signals flowing through it:

$$\mathcal{E} = \mathcal{C} + \mathcal{N} \qquad \text{(reference image)}$$
$$\mathcal{F} = \mathcal{D} + \mathcal{N}' \qquad \text{(test image)}$$

where $\mathcal{E}$ and $\mathcal{F}$ denote the visual signals in a single subband at the output of the HVS model. $\mathcal{N}$ and $\mathcal{N}'$ are assumed to be uncorrelated, zero mean multivariate Gaussian, having both the same covariance $\Sigma_{\mathcal{N}} = \Sigma_{\mathcal{N}'} = \sigma_n^2 I$.

The mutual information $I(\overrightarrow{C}^{N_j}; \overrightarrow{E}^{N_j})$ resp. $I(\overrightarrow{C}^{N_j}; \overrightarrow{F}^{N_j})$ is the amount of information the HVS system is capable of extracting from the j-th subband of the reference image resp. the test image (where $\overrightarrow{C}^{N_j} = \overrightarrow{C}_{1,j}, \ldots, \overrightarrow{C}_{N,j}$ denote N elements from $\mathcal{C}$ in one fixed subband j).

The VIF is then defined as

$$VIF = \frac{\sum\limits_{j \in \text{subbands}} I(\overrightarrow{C}^{N_j}; \overrightarrow{F}^{N_j})}{\sum\limits_{j \in \text{subbands}} I(\overrightarrow{C}^{N_j}; \overrightarrow{E}^{N_j})}. \tag{2.5}$$

It is the ratio of the information contained in the test image divided by the information in the reference image. The VIF metric is in $[0, 1]$. It is exactly 1 if the test and distorted images are identical.

2.7 The design of the benchmark

The experimental settings are briefly summarized in the following: Four ground-truth-images (in the following called "scene") are blurred using twelve PSFs which results in 48 blurred images. On each of these blurred images seven state-of-the-art deblurring algorithms (Cho et al. [18], Xu et al. [139], Shan et al. [109], Fergus et al. [36], Krishnan et al. [69], Whyte et al. [134], Hirsch et al. [56]) have been tested. The image quality of the blurry and deblurred images has been measured using four different image quality metrics (*IQM*): PSNR, MS-SIM [127], IFC [111] and VIF [110].

All deblurred images including the estimated kernels are shown on the project website [65].

The seven deblurring algorithms can be divided into two groups:

- Algorithms which assume a uniform blur model and account for translational motion only [36, 109, 18, 139, 69].

[5] As in the IFC metric a subband of the reference resp. test image is modeled as a GSM RF (denoted by $\mathcal{C}$ resp. $\mathcal{D}$).

- Algorithms which assume a non-uniform blur model. In particular, [133] assumes rotational motion (yaw, pitch and roll), while [56] considers translations and in-plane rotations only.

For fair comparison, we asked the authors of the tested algorithms to run their code on our benchmark dataset and to provide us with their best deblurring results. The results reported for [109, 139, 56, 133] were provided by the authors, while for [36, 18, 69] we used provided code to yield the deblurring results ourselves. Regarding parameter settings, we followed the provided instructions or the personal advice of the authors and did our best to optimally adjust the free deblurring parameters incl. kernel size (and image patch in the case of [36]).

Although runtime is a critical factor and a discriminantive feature of deblurring algorithms, we do not report any runtimes here as the results were obtained by heterogenous implementations (e.g. Matlab vs. C) and different hardware systems (e.g. CPU vs. GPU).

To compare similarity between two images a and b (represented as vectors), we first estimate the optimal scaling $\hat{\alpha}$ and translation $\hat{T}$ such that the L2 norm between a and b becomes minimal,[6] i.e. $\hat{\alpha}, \hat{T} = \min_{\alpha,T} \|a - T(\alpha b)\|^2$. We then calculate the different image metrics. Given a sequence of ground truth images $u_1^*, \ldots, u_N^*$ along the trajectory, we define the IQM similarity between an estimated image $\hat{u}$ and the ground truth as the maximum IQM between $\hat{u}$ and any of the images along the trajectory,

$$\mathrm{SIM}_{IQM} = \max_n \mathrm{IQM}(u_n^*, \hat{u}). \tag{2.6}$$

Tables 6.1-6.4 report the SIM_{IQM} for all of the 48 benchmark images and the four used image quality metrics, where we computed the IQM score according to Eq. (2.6). For better visual assessment we color coded the results from blue (low IQM score, poor performance), green (intermediate IQM score) to red (high IQM score, good deblurring performance).

Twelve blurry and deblurred images are shown in Fig. 2.10. We chose the images, so that each of the twelve 6D trajectories is represented once and that each image is represented three times. For each blurry image, the deblurring algorithm, that performed the best w.r.t the PSNR score was chosen.

[6] We allow for integer pixel translations only, which we estimate with the Matlab function `dftregistration` by [45].

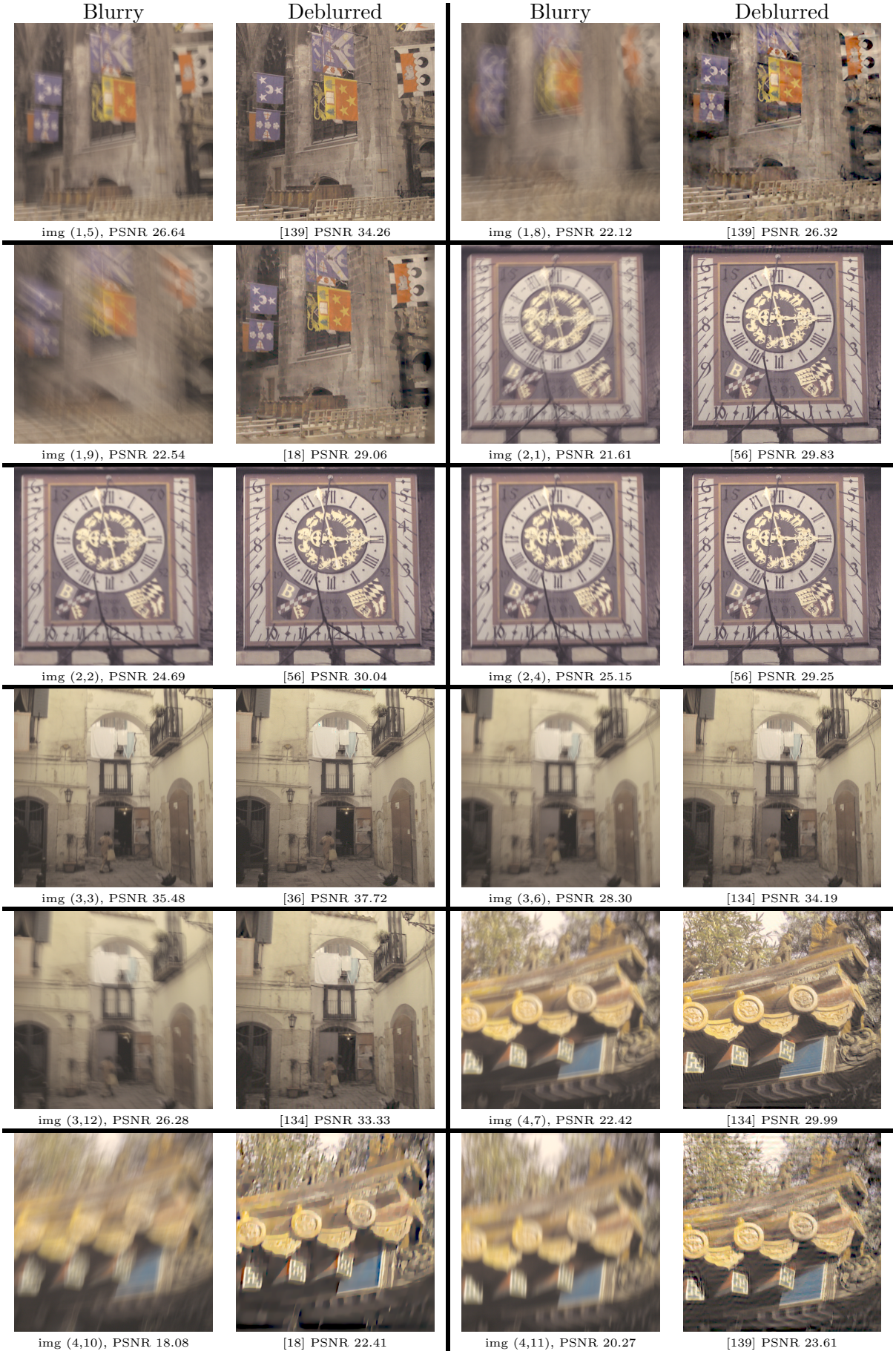


Figure 2.10: Twelve blurry images, each blurred by a different 6D trajectory. The deblurred images were chosen to be the best performing algorithms w.r.t the PSNR score for each particular image. Note that the images were gamma corrected with gamma = 2.2.

2.8 Statistical evaluation of the performance of the deblurring algorithms

In this section the results of the seven deblurring algorithms on the 48 blurry images are statistically evaluated.

The results of the four IQMs are differently scaled and not linearly dependent, so it is not straightforwardly possible to combine the IQMs in the statistical evaluation. Hence, we did the statistical evaluation for all four IQMs individually.

In the following subsections the influence of the image quality metric, the scene, the PSF and the deblurring algorithm on the deblurring result is illustrated. Following results will be shown:

- The four image quality metrics tend to rank the quality of the 48 pictures in a similar way with PSNR differing the most from the other three metrics.
- The PSFs $8, 10, 11, 9$ are the most difficult PSFs for the seven tested algorithms, the PSF 3 is the easiest one. This conforms to the average kernel size (kernel three has the smallest size, whereas kernels eight to eleven have the largest sizes). The PSFs can be seen on the project website [65].
- All deblurring algorithms except for the methods of Cho et al. [18] and Xu et al. [139] have difficulties with the large PSFs 8-11.
- The two scenes 'clock' and 'roof' are harder to deblur, whereas the other two scenes 'backyard' and 'church' are easier to deblur. We reckon that the images 'church' and 'backyard' have less high frequency content, i.e. less edges, which are destroyed during the blurrying process. Those two images have large smooth areas, like the walls or the flags.
- The strongest deblurring algorithm on the benchmark dataset is Xu et al. [139], followed by Cho et al. [18]. The weakest one - at the same time the oldest one benchmarked - is by Fergus et al. [36], which sometimes results in deblurred images which have a lower IQM score than the blurred input image.

2.8.1 How similar are the scores of the four image quality metrics?

In this section we show that the scores of the four used IQMs are correlated, i.e. that high scores for a specific IQM normally result in high scores for another IQM. The correlation is not perfect for all four IQMs. The IQMs IFC and VIF have the highest correlation, see Fig. 2.11 and Table 2.2.

	PSNR - MS-SSIM[127]	PSNR - IFC [111]	PSNR - VIF [110]	MS-SSIM - IFC	MS-SSIM - VIF	IFC - VIF
Blurred	0.91	0.59	0.67	0.77	0.85	0.97
Cho [18]	0.83	0.69	0.87	0.91	0.91	0.88
Xu [139]	0.81	0.70	0.87	0.95	0.93	0.90
Shan [109]	0.87	0.75	0.83	0.92	0.92	0.95
Fergus [36]	0.87	0.70	0.73	0.91	0.93	0.98
Krishnan [69]	0.89	0.83	0.89	0.95	0.95	0.97
Whyte [134]	0.91	0.82	0.90	0.95	0.97	0.98
Hirsch [56]	0.87	0.83	0.90	0.96	0.96	0.97

Table 2.2: Spearman's rank correlation coefficients comparing the relationship between image quality metrics for the blurred images and for each deblurring algorithm.

The IQMs PSNR and IFC have the lowest correlation. Images with a higher score on IFC tend to score higher on PSNR but there are some deviations, as the regression line is not always strictly monotone increasing.

In Table 2.2 the Spearman's rank correlation coefficients between the different IQMs on the blurred images and the deblurred images of each algorithm are summarized. We chose Spearman's rank correlation as the IQMs are not linearly correlated, as can be seen in Fig. 2.11.

The Spearman's rank correlation coefficients are similar for each pairwise comparison of IQMs (each column in Table 2.2). Only the pair PSNR vs. IFC seems to differ with some coefficients being around 0.7 and some around 0.8. The coefficient for the blurred values is lower for the pairs PSNR-IFC, PSNR-VIF and MS-SSIM-IFC.

Overall the metric PSNR differs the most from the other metrics. Especially the comparison between PSNR and IFC, see Fig. 2.11, results in the lowest correlation coefficients.

2.8.2 Influence of the PSF on the image quality after deblurring

In this section we show, as expected, that the PSF has an influence on the deblurring result.

In Fig. 2.12 the IQM scores against the twelve PSFs are plotted.

To answer the question which of the PSFs is the most difficult, we rank the IQM scores of the PSFs for each combination out of the seven deblurring algorithms and four scenes, see Table 2.3. The PSF with the highest score for all IQMs is PSF number three, meaning it is the easiest PSF to deblur. PSFs $8, 10, 11, 9$ are the most difficult PSFs, in this order. PSFs $1, 2$ and 4 form a group of easier PSFs.

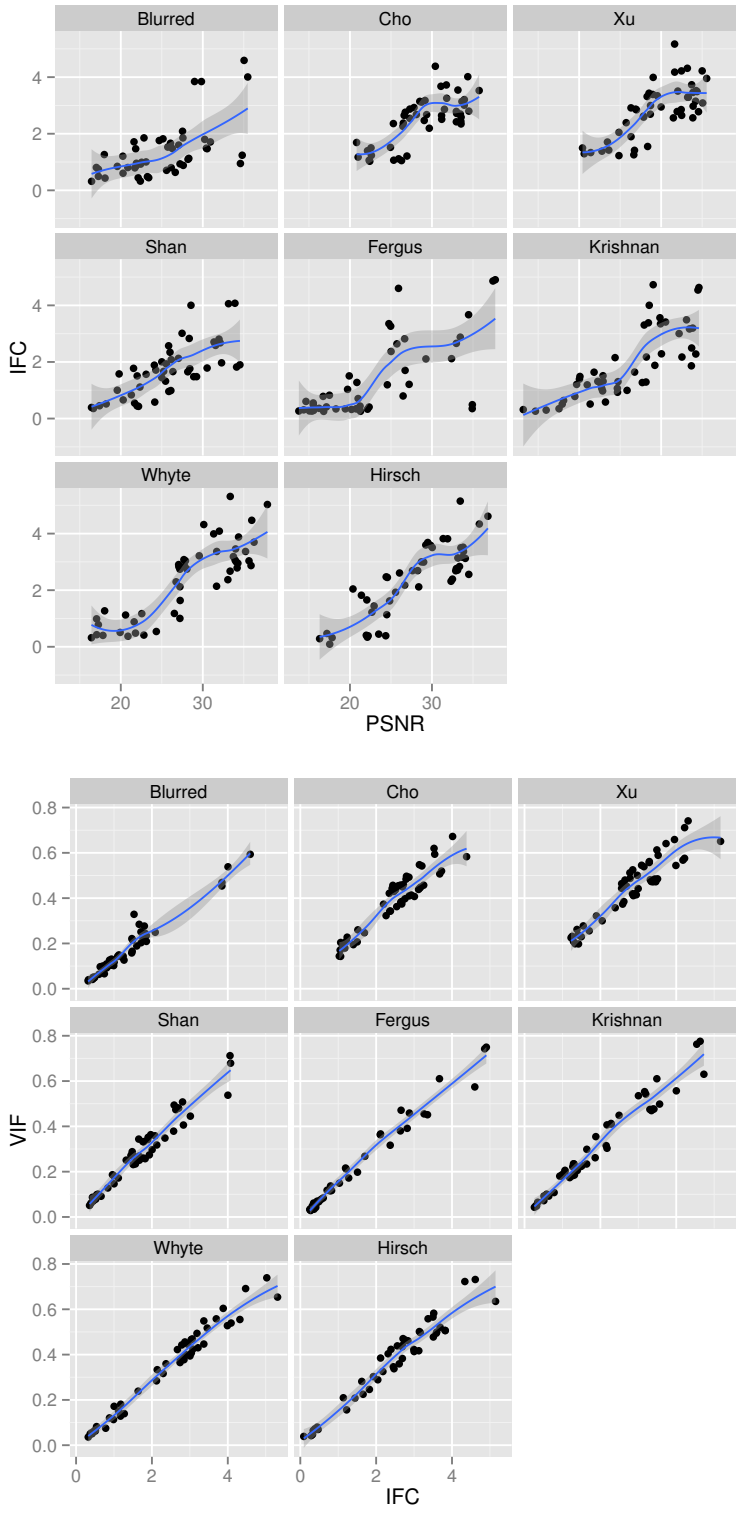


Figure 2.11: Image quality metric values for metrics PSNR vs. IFC and IFC vs. VIF. Each dot represents one of the 48 deblurred images. PSNR and IFC do not correlate highly, wherease IFC and VIF correlate quite high. The blue regression line is fitted using the loess smoothing method, polynomial regression fitting is used, see [20]. For the fit at a point x, 75 % of the points in the neighborhood are used and these are weighted. The gray area is the 95% confidence intervall of the regression line.

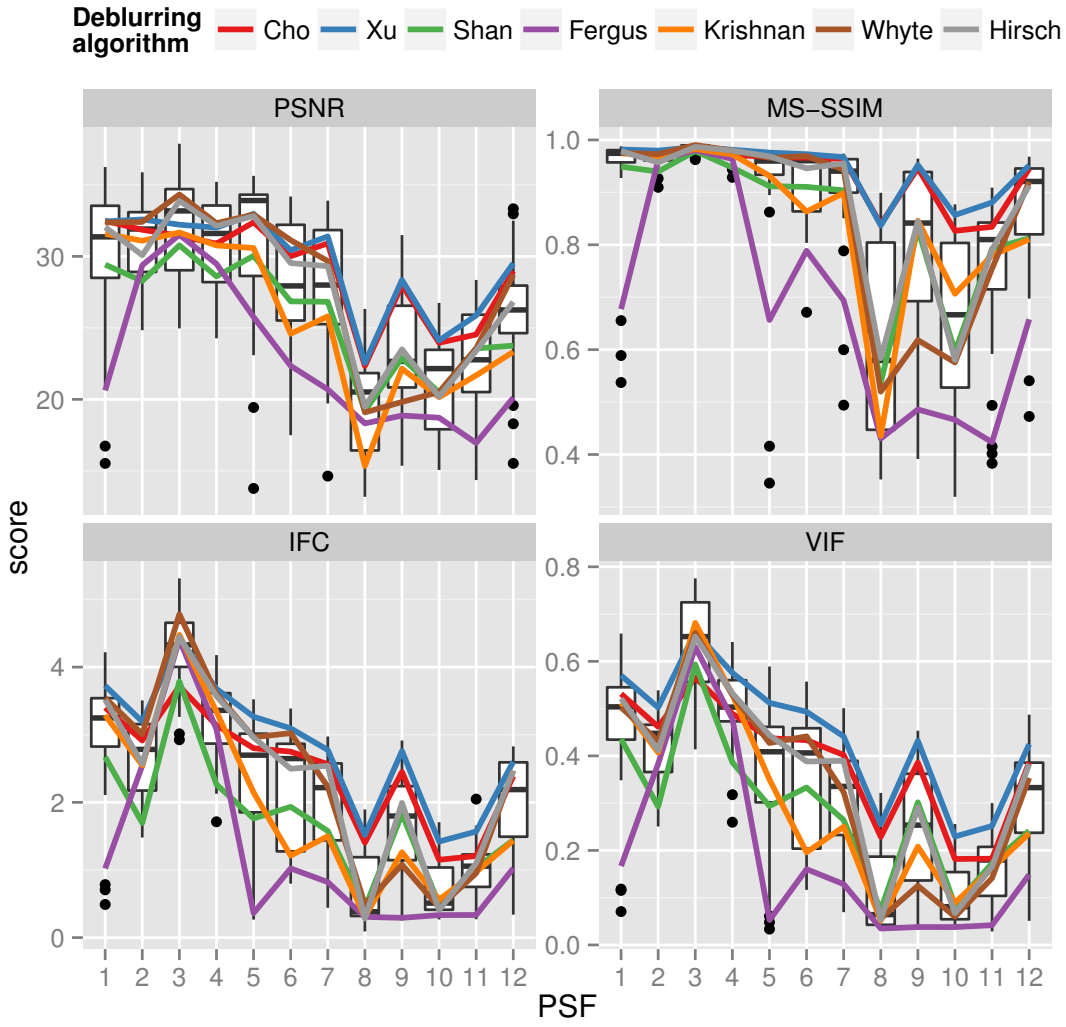


Figure 2.12: IQM scores depend on the PSF. Each boxplot is build from 28 data points from four scenes and seven deblurring algorithms. The mean over the four scenes for each deblurring algorithm is plotted as a line. The PSFs 8-11 are the most difficult to deblur. All deblurring algorithms except for Cho et al. [18] and Xu et al. [139] have difficulties with those large PSFs.

IQM	1	2	3	4	5	6	7	8	9	10	11	12
PSNR	3.68	3.50	2.25	3.89	3.36	6.32	6.39	11.50	9.00	10.46	9.75	7.89
MS-SSIM	3.21	3.79	1.32	2.89	5.46	5.61	6.50	11.46	8.79	10.96	9.82	8.18
IFC	2.93	4.71	1.07	2.79	5.64	5.96	7.04	11.14	8.39	11.04	9.93	7.36
VIF	2.93	4.64	1.07	2.68	5.50	5.89	7.11	11.32	8.43	11.07	10.07	7.29

Table 2.3: For each of the 4×7 *combinations of scene and deblurring algorithm, the twelve PSFs were ranked. The average rank over the* 28 *values has been calculated separately for each IQM. An average rank of 1.0 for a certain PSF means that for all* 28 *combinations of scene and deblurring algorithm the PSF has always the highest IQM score compared to the other eleven PSFs.*

These results conform to the average size of the blur kernels. As can be seen on our project website [65], PSFs eight to eleven are the largest PSFs, PSF three is the smallest and PSF two and four are also rather small. In Fig. 2.12 it can be seen that for the large kernels 8 - 11, all benchmarked algorithms except for the methods of Cho et al. [18] and Xu et al. [139] have difficulties.

2.8.3 Influence of the scene on the image quality after deblurring

In this section we show that the four scenes influence the image quality of the deblurred image, i.e. that there are scenes which are more difficult to deblur.

In Fig. 2.13 the IQM scores against the four scenes are plotted.

In Table 2.4 the ranks of the IQM scores of the 84 combinations of the twelve PSFs and seven deblurring algorithms have been averaged. One can see that the 'clock' and 'roof scene are the most difficult, having the lowest average rank for all IQMs, except IFC. For IFC all scenes rank about the same. The 'clock' and 'roof' scenes are the more difficult as they contain more edges than the 'backyard' or 'church' scenes.

2.8.4 The ranking of the deblurring algorithms

In this section the performance of the deblurring algorithms is compared. First we show that the deblurring algorithms perform statistically different on the benchmark dataset, then we try to find the best performing algorithm.

Is there a significant difference between deblurring algorithms?

To test if there is a significant difference between the deblurring algorithms, a one-way within subjects analysis of variance (Anova), see [35], is conducted for each image quality

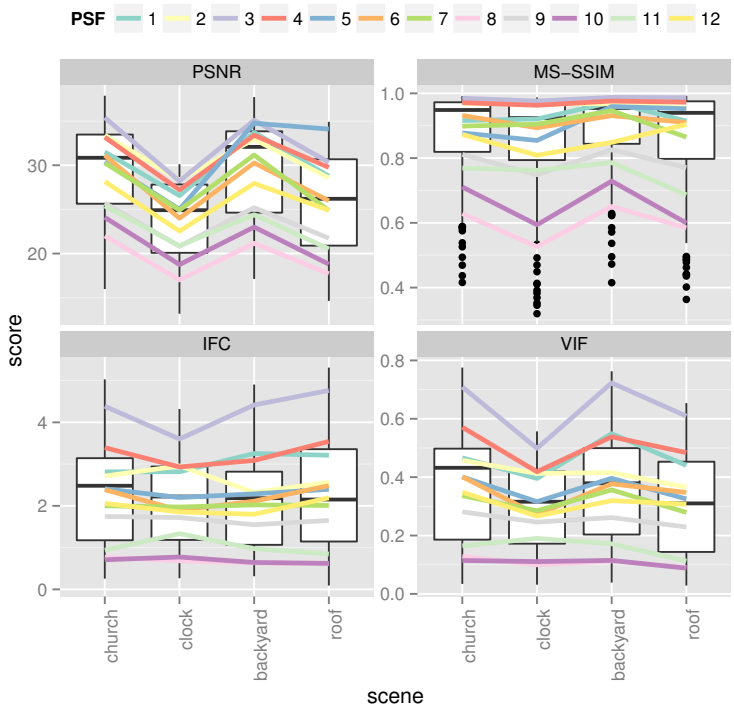


Figure 2.13: IQM scores depend on the scene. Each boxplot is build from 84 data points from twelve PSFs and seven deblurring algorithms. The mean over the four scenes for each PSF is plotted as a line. The scene has an influence on the deblurring score, as the lines are not parallel to the x-axis but have kinks. Scenes 'clock' and 'roof' seem to be the most difficult to deblur, which is especially visible in the IQMs PSNR and VIF.

IQM	church	clock	backyard	roof
PSNR	1.60	3.80	1.50	3.11
MS-SSIM	2.18	3.31	1.82	2.69
IFC	2.15	2.68	2.80	2.37
VIF	1.67	3.29	1.85	3.20

Table 2.4: For each of the 12×7 combinations of PSF and deblurring algorithm, the four scenes were ranked. The average rank over the 84 values has been calculated separately for each IQM. An average rank of 1.0 for a certain scene means that for all 84 combinations of PSF and deblurring algorithm the scene has always the highest IQM score compared to the other three scenes.

metric to compare the effect of the *deblurring algorithms* on the deblurring improvement. The deblurring improvement was measured as the difference between the IQM score after deblurring and the blurred score.

The effect of the *deblurring algorithms* on the deblurring improvement is significant at the level of $\alpha = 0.05$ for the seven deblurring algorithms on all 48 images for all IQMs after correcting for multiple testing with the Bonferroni correction (see [35]), because it was tested on all four IQMs. The result of the one-way Anova for each IQM is the following. The test statistic follows the F-distribution with degrees of freedom $7 - 1 = 6$ and $(48 - 1) \cdot (7 - 1) = 282$:

PSNR	$\left(F(6, 282) = 50.58, p = 5.0710^{-42}\right)$
MS-SSIM	$\left(F(6, 282) = 42.55, p = 7.7810^{-37}\right)$
IFC	$\left(F(6, 282) = 51.05, p = 2.6110^{-42}\right)$
VIF	$\left(F(6, 282) = 57.8, p = 2.4610^{-46}\right)$

Therefore, one can clearly conclude that there is a significant influence of the deblurring algorithm on the IQM score after deblurring.

Which is the best deblurring algorithm?

Fig. 2.15 shows the boxplots of the image quality improvement for different deblurring algorithms and IQMs. The algorithm by Fergus et al. [36] achieves the lowest improvement for all four IQMs and some of the deblurred images have a worse quality than the blurred image itself, resulting in a negative score. The algorithm by Xu et al. [139] tends to have the best results followed by Cho et al. [18]. The boxplots do not show the paired comparison for each of the 48 pictures, but Fig. 2.14 does. There it can be seen that the algorithm by Xu et al. [139] outperforms Cho et al. [18] in almost every blurred image.

To see which deblurring algorithm has the highest mean improvement on all 48 images, i.e. which algorithm is the best, a post-hoc test (see [59]) on the Anova is conducted. For the within-subject design, Keppel [61] suggests Student's t-test comparison with adjusted p-values for multiple testing. In the following, the Holm-Bonferroni correction is applied for multiple testing. As the image quality metric is measured for each picture on different deblurring algorithms, a paired Student's t-test is conducted.

Table 2.5 contains the average score of improvement for each of the deblurring algorithms for each of the four IQMs. Xu et al. [139] and Cho et al. [18] have the highest and second highest means for all metrics and Fergus et al. [36] the lowest.

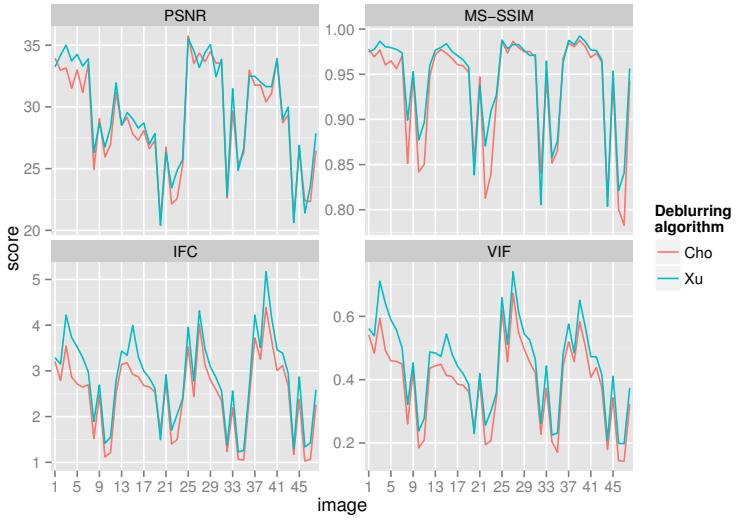


Figure 2.14: Comparing the algorithms by Cho et al. [18] and Xu et al [139]. The algorithm by Xu et al. is able to achieve better results for almost all blurred images considering each of the four image quality metrics.

Deblurring Algorithm	PSNR	MS-SSIM	IFC	VIF
Cho	4.053	0.150	1.142	0.218
Xu	4.607	0.161	1.487	0.273
Shan	0.961	0.059	0.395	0.117
Fergus	-2.200	-0.101	-0.052	0.019
Krishnan	0.796	0.065	0.572	0.134
Whyte	3.141	0.066	1.034	0.165
Hirsch	2.835	0.091	1.014	0.186

Table 2.5: Average score of improvement for each of the seven deblurring algorithms for each of the four IQMs.

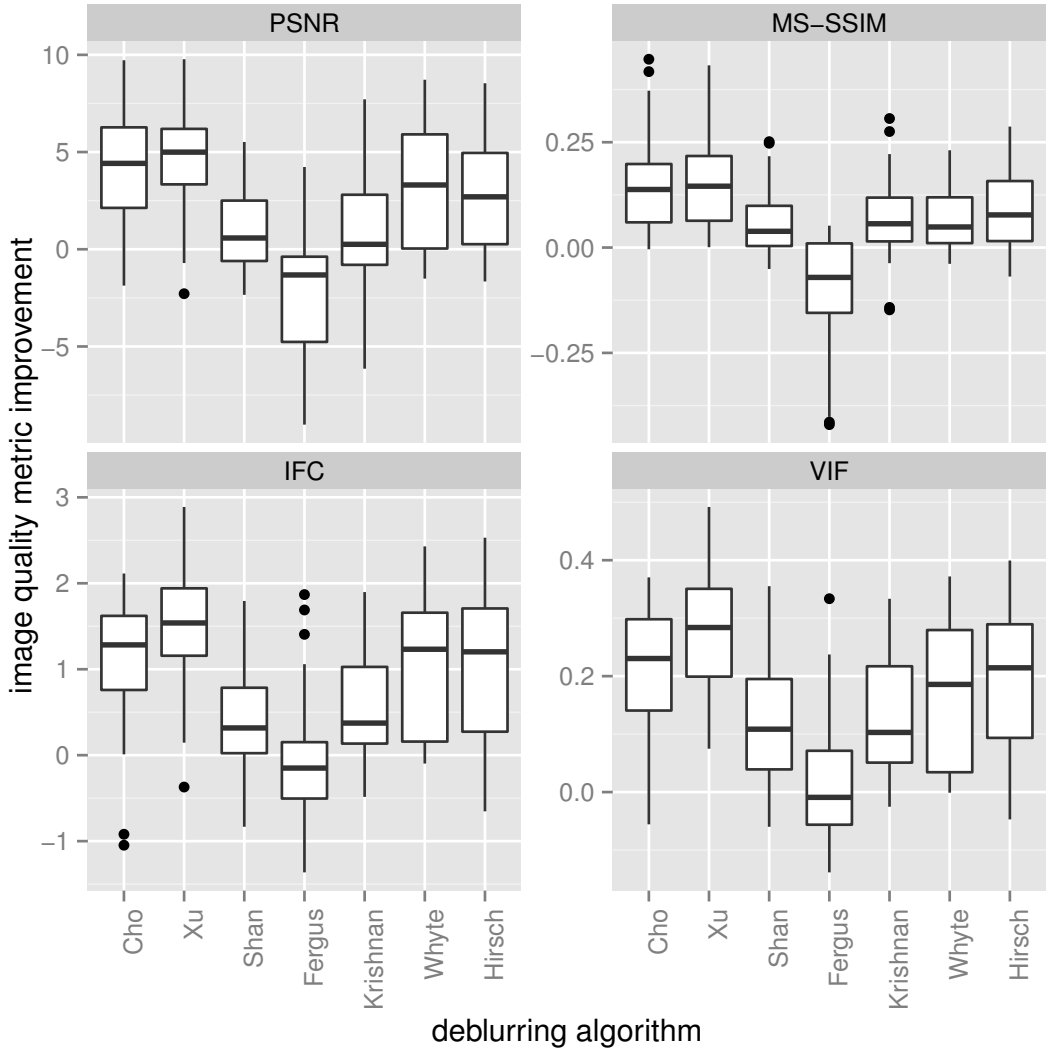


Figure 2.15: Improvement on IQM score for different IQMs and deblurring algorithms. The algorithm by Fergus et al. [36] achieves the lowest improvement for all four IQMs, the algorithm by Xu et al. [139] tends to have the best results.

ranking	PSNR	MS-SSIM	IFC	VIF
Rank 1	Xu	Xu	Xu	Xu
Rank 2	Ch/Wh/Hi	Ch	Ch/Wh/Hi	Ch
Rank 3	Sh/Kr	Sh/Kr/Wh/Hi	Sh/Kr	Sh/Kr/Wh/Hi
Rank 4	Fe	Fe	Fe	Fe

Table 2.6: Which deblurring algorithm is the best? Algorithms which share a rank are not statistically different. The algorithm by Xu et al. performs the best for all four IQMS and the algorithm by Fergus et al. the worst. The algorithms are abbreviated as follows: Ch=Cho[18], Xu=Xu [139], Sh=Shan [109], Fe=Fergus [36], Kr=Krishnan [69], Wh=Whyte [134], Hi=Hirsch [56].

Table 2.6 ranks the deblurring algorithms according to their mean value over all 48 images. Deblurring algorithms ranked the same do not have significantly different paired quality improvement values: e.g. the algorithms by Cho et al., Whyte et al. and Hirsch et al. share the second rank for the measure IFC. The mean of the IQM score improvement, shown in Table 2.5 is 1.142 for Cho et al., 1.034 for Whyte et al. and 1.014 for Hirsch et al. These average scores seem to be similar. Table 2.7 shows the p-values of the paired Student's t-test with the null hypothesis that the average scores are the same. The p-values are 0.5867 for the pair Cho-Whyte, 0.4316 for Cho-Hirsch and 0.8105 for Whyte-Hirsch, meaning that above null hypothesis cannot be rejected at a reasonable significance level of e.g. $\alpha = 0.1$. Hence we conclude that the three algorithms have to share a rank.

The algorithm by Xu et al. is ranked first as above null hypothesis can be rejected at a significance level of e.g. $\alpha = 0.01$, for all pairwise tests. Furthermore the average score of improvement shown in Table 2.5 of Xu et al. is 1.487, which is higher than for any other algorithm.

Considering all IQMs, we conclude that the "average" ranking of the algorithms is:

1. Xu
2. Cho
3. Whyte and Hirsch
4. Shan and Krishnan
5. Fergus

		Cho [18]	Xu	Shan	Fergus	Krishnan	Whyte
PSNR	Xu [139]	0.0003					
	Shan [109]	0.0000	0.0000				
	Fergus [36]	0.0000	0.0000	0.0000			
	Krishnan [69]	0.0000	0.0000	0.6863	0.0001		
	Whyte [134]	0.1374	0.0075	0.0000	0.0000	0.0000	
	Hirsch [56]	0.0075	0.0000	0.0000	0.0000	0.0000	0.6863

		Cho	Xu	Shan	Fergus	Krishnan	Whyte
MS-SSIM	Xu	0.0026					
	Shan	0.0000	0.0000				
	Fergus	0.0000	0.0000	0.0000			
	Krishnan	0.0000	0.0000	1.0000	0.0000		
	Whyte	0.0010	0.0003	1.0000	0.0000	1.0000	
	Hirsch	0.0019	0.0004	0.0036	0.0000	0.1125	0.2251

		Cho	Xu	Shan	Fergus	Krishnan	Whyte
IFC	Xu	0.0000					
	Shan	0.0000	0.0000				
	Fergus	0.0000	0.0000	0.0111			
	Krishnan	0.0000	0.0000	0.2629	0.0000		
	Whyte	0.5867	0.0000	0.0000	0.0000	0.0000	
	Hirsch	0.4316	0.0000	0.0000	0.0000	0.0000	0.8105

		Cho	Xu	Shan	Fergus	Krishnan	Whyte
VIF	Xu	0.0000					
	Shan	0.0000	0.0000				
	Fergus	0.0000	0.0000	0.0001			
	Krishnan	0.0000	0.0000	0.2471	0.0000		
	Whyte	0.0045	0.0000	0.0168	0.0000	0.0885	
	Hirsch	0.0371	0.0000	0.0000	0.0000	0.0007	0.1479

Table 2.7: P-values from the paired Student's t-test with the null hypothesis that the means of the quality metric improvement for the respective IQM is the same for each pairwise comparison. The p-values are adjusted for multiple testing using the Holm-Bonferroni correction.

3

Inpainting with Deep Neural Networks

Chapter abstract Most approaches to fill in holes in images, also called the *inpainting* problem, usually require a good image model to infer the unknown pixels. In this chapter, we directly learn a mapping that maps image patches, corrupted by missing pixels, onto complete image patches. This mapping is represented as a deep neural network that is automatically trained on a large image data set. In particular, we are interested in the question whether it is helpful to exploit the shape information of the missing regions, i.e. the masks, which is something commonly ignored by other approaches. In comprehensive experiments on various images, we demonstrate that our learning-based approach is able to use this extra information and can achieve state-of-the-art inpainting results. Furthermore, we show that training with such extra information is useful for *blind* inpainting, where the exact shape of the missing region might be uncertain, for instance due to aliasing effects.

The chapter is based on the following publication:

[67] R.Köhler, C. Schuler, B. Schölkopf and S. Harmeling. Mask-specific inpainting with deep neural networks. German Conference on Pattern Recognition (GCPR). 2014.

3.1 Contributions

While existing inpainting methods[1] lead to impressive image reconstruction results, to the best of our knowledge, none of them is exploiting the shape of the mask for inpainting. In this chapter we show how this additional information can be utilized to obtain better image reconstruction results. For this we choose a task-specific learning approach employing deep neural networks since they have recently been successfully applied to several other image restoration problems, e.g. to image denoising [10] or to image deblurring [106]. The authors of [138] apply a deep learning approach to both denoising and inpainting. They are also able to do blind inpainting (as we do in Sec. 3.3.4), but do not use the mask information.

In a nutshell, the contributions of this chapter are as follows:

- We show that a mask-specific inpainting method can be learned with neural networks. It is able to compete with the best inpainting methods and often leads to improved performance.
- We show that it is relevant to train the inpainting method with the correct masks.
- We show that by training an inpainting method for masks generated with certain fonts, it is possible to *blindly* inpaint an image, i.e. without knowing the locations of the missing pixels.

In sections 3.2 - 3.2.2, we briefly explain the deep neural networks we employed, and how information about the masks is used during the training process. Next, in sections 3.2.3, 3.2.4, we demonstrate that the performance depends on the type of mask used during training. In the results section 3.3, we further show that our learning approach is able to compete with the state-of-the-art methods in image inpainting on commonly used images. Furthermore (Sec. 3.3.4), we show that the learning based approach can be applied to images without providing the locations of the missing pixels, which further simplifies the image recovery process. Finally (Sec. 3.4), we try to understand the inner workings of the learned neural networks.

3.2 Learning mask-specific inpainting methods

The overall idea is to train a neural network to map corrupted image patches to their uncorrupted counterparts. This mapping is applied to all patches of a corrupted image.

[1] See chapter 1.3 for an introduction to inpainting.

The recovered patches are averaged at their overlapping parts to obtain the reconstructed image. In case the true mask for the whole picture is given, the known pixels from the input image are directly copied to the reconstructed image. Note that the input patches are usually larger than the output patches, which matches the intuition that the missing pixels can be recovered by considering some large enough area around them. In the following we briefly recall Multi-Layer Perceptrons (MLPs) and explain the training procedure.

3.2.1 Multi-Layer Perceptrons

A Multi-Layer Perceptron (which is an instance of a neural network) is a nonlinear function f that maps input vectors x onto output vectors y. We use the same notation as [10]. For instance, an MLP with two hidden layers can be written as

$$f(x) = b_3 + W_3 \tanh(b_2 + W_2 \tanh(b_1 + W_1 x)) \tag{3.1}$$

(from Eq. (1) in [10]) where vectors b_1, b_2, b_3 and the matrices W_1, W_2, W_3 are the parameters of the MLP. More generally we denote the architecture of an MLP by some integer tuple: e.g. an $(256, 1024, 1024, 64)$-MLP has a 256-dimensional input layer, two 1024-dimensional hidden layers and an 64-dimensional output layer.

In theory one hidden layer is enough to approximate any function, as shown in [22, 39, 57]. This result is known as the *universal approximation theorem* (see below). In practice however one hidden layer is often not sufficient, but several hidden layers may increase the representational power of the neural net [71, 114] and make the optimization task easier.

The universal approximation theorem as stated by [22]:

Universal Approximation Theorem:
Let I_n denote the n-dimensional unit cube $[0,1]^n$ and let $C(I_n)$ denote the space of continuous functions on I_n. Let $\alpha_j, b_j \in \mathbb{R}$ and $x, w_j \in \mathbb{R}^n$ and σ be any continuous sigmoidal function . Then finite sums of the form

$$f(x) = \sum_{j=1}^{N} \alpha_j \sigma(w_j^T x + b_j)$$

are dense in $C(I_n)$. In other words, given any $g \in C(I_n)$ and $\epsilon > 0$, there is a sum, $f(x)$, of the above form, for which

$$|f(x) - g(x)| < \epsilon \qquad \text{for all} \quad x \in I_n$$

The proof of the universal approximation theorem can be found in [22]. Intuitively (and slightly imprecise) the universal approximation theorem states that any continuous function can be approximated with arbitrary accuracy by a fully connected Neural Net with just a single hidden non-linear layer, as long as it has enough neurons.

3.2.2 Mask-specific training

To adjust the parameters of the MLP, we need training data that consists of pairs of input patches x and output patches y. Using these we can automatically *learn* the mapping using the backpropagation algorithm [104, 72]. To speed up convergence we use the ADADELTA method [143] that automatically adapts parameter-specific learning rates by approximating the Hessian matrix of second derivatives. We kept the batch size fixed to 128 and the conditioning constant to $\epsilon = 10^{-6}$, as recommended by [143].

Similar to [107] we found that the best parameters were a learning rate of 0.01 and a decay rate of 0.95. Fig. 3.1 shows a comparison of different parameter settings for a $(29^2, 2047, 2047, 2047, 13^2)$-MLP. The training PSNR shown in Fig. 3.1 was evaluated on 100 image patches of size 40×40, which are depicted in Fig. 3.2. Those patches were cropped from the center of each of the 100 test images of the Berkley segmentation dataset 300 [86].

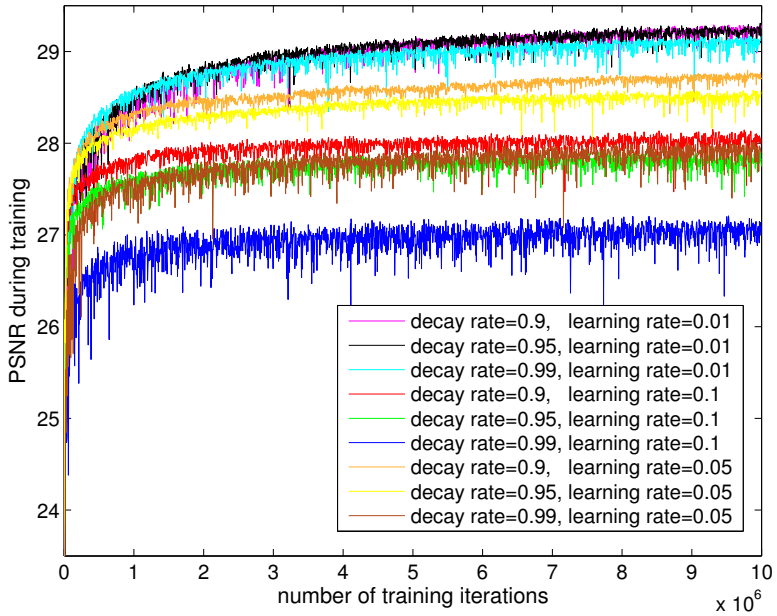


Figure 3.1: Comparison of different parameters for the ADADELTA method [143]. The training PSNR was evaluated on 100 patches of size 40×40*, which are shown in Fig. 3.2.*

Figure 3.2: 100 patches of size 40×40 *that were used to evaluate the PSNR during training. The patches are cropped from the center of each of the 100 images of the Berkley segmentation dataset 300 [86].*

To generate large amounts of training pairs we randomly select image patches from an image database (we used ImageNet [27] for training). These extracted patches are the uncorrupted output patches.

To corrupt those patches we utilize the knowledge about the masks. For instance for super-imposed text we corrupt the output patches by adding random text of the same font and size, if that information is available. This allows us to create input patches with corruptions that are similar to the corruptions in the test image.

3.2.3 Training with and w/o mask as input patch

Many inpainting algorithms require the exact locations of the missing pixels. For this reason we consider two versions of our approach: the first considers only the corrupted patch as the input. Note that the corrupted patch does not always contain the information which pixels are missing (e.g. for blind inpainting, see Sec. 3.3.4). Version two of our approach requires the corrupted patch and additionally the masking patch as the input.

To show that feeding the masking patch additionally helps we performed a comparison of the two approaches on 10 test images (randomly selected from the Berkeley segmentation dataset [86]) with super-imposed text. Fig. 3.3 shows the results of two neural nets, both with architecture $(16^2, 512, 512, 512, 8^2)$ which were trained for the same amount of time. The first MLP was trained with the mask as an additional input, the second MLP was trained without the mask, but with correctly corrupted patches, i.e. using the correct font and text size. In all cases the MLP with the additional input was better. See also Fig. 3.6 for example images.

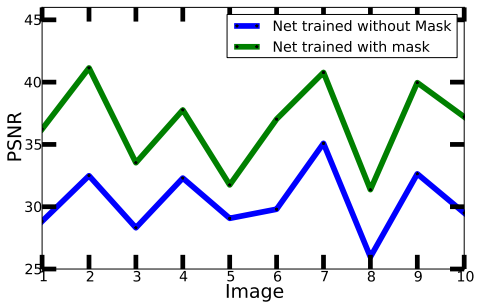


Figure 3.3: PSNR for 10 test images for an MLP that takes only the corrupted patch as input (blue line) and an MLP that additionally has the masking patch as input (green line).

3.2.4 Training with the correct and wrong masks

In this section we demonstrate the advantage that is gained through incorporating the correct mask into the learning process. For that purpose we generate several masks which are similar. We start with a mask, showing vertical bars of size 14×3 pixels. We rotate those bars repeatedly by 15 degrees. We do not allow aliasing, but only binary masks, so each pixel in the mask which is greater than 0 after rotation is set to 1.

We trained several MLPs with the architecture $(16^2, 512, 512, 512, 8^2)$ using training images generated with these different masks. For each of those angles, also 10 test images are generated (10 randomly picked images from the Berkeley segmentation dataset [86]).

Fig. 3.4 shows that the neural net performs best on the angle it was trained on, and that the performance deteriorates quickly for other angles. The methods by [105] and [85] perform about the same for all angles, no matter in what way the bars are orientated. Our approach is only better if the correct angle has been considered during training time.

A similar behavior was observed by [11] for Neural Nets that were trained for the denoising task: Images were corrupted with Gaussian-noise of different noise levels σ and for each of those noise level a Neural Net was trained on. The performance of the Neural Net was best for the noise level it was trained on and deteriorated quickly for other noise levels.

3.2.5 Inpainting a whole image

In order to inpaint a whole image and not only a single patch, the corrupted image is chopped into overlapping patches with a sliding window approach. The size of each

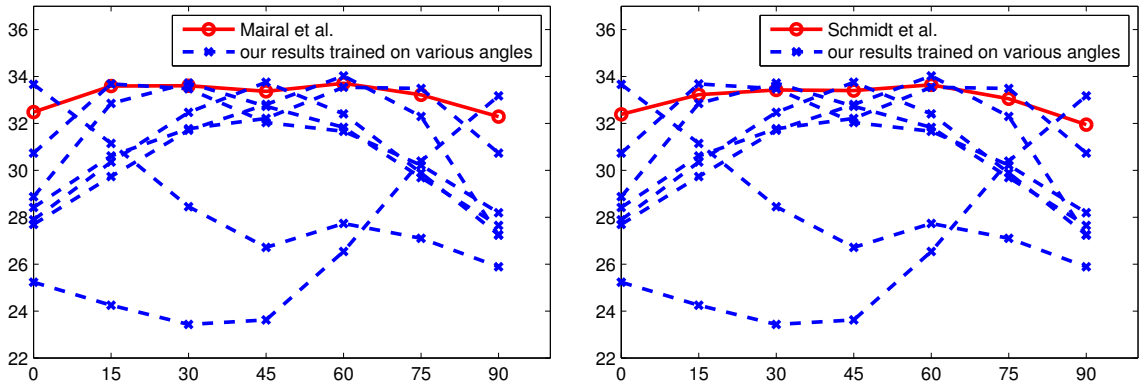


Figure 3.4: Performance of the MLP trained on different angles compared against the method by **(left)** *[85] and* **(right)[105]**. *The performance of the neural net depends on the orientation of the bars and is best for the orientation on which the neural net was trained on. The performance was measured as the average PSNR value of ten corrupted images. One corrupted image and the results of differently trained MLPs can be seen in Fig. 3.5.*

patch is the same as the input-size of the MLP. The stride of the sliding window was one.

Each extracted patch was inpainted on its own with the trained MLP. The inpainted patches were combined by taking the arithmetic average of them. We always trained the MLPs on gray-scale images. To apply them to color we applied the same MLP separately to the three color channels, see Fig. 3.7.

3.3 Experiments

To generate training patches we used 1.8 million color images from ImageNet [27], which were converted to gray-scale.

Note that we did not have access to implementations of all competing methods. For Figs. 3.8, 3.9 and 3.10 we applied all algorithms that were available to us. Furthermore, we included for Fig. 3.8 those methods which published results on the "New Orleans" image in their papers. Fig. 3.10 shows a comparison on images from [140].

3.3.1 Comparison on the New Orleans image

We compare our method against other methods on the New Orleans image, used by [3]. For this comparison a $(39^2, 2047, 2047, 2047, 2047, 13^2)$ architecture was trained. Visually

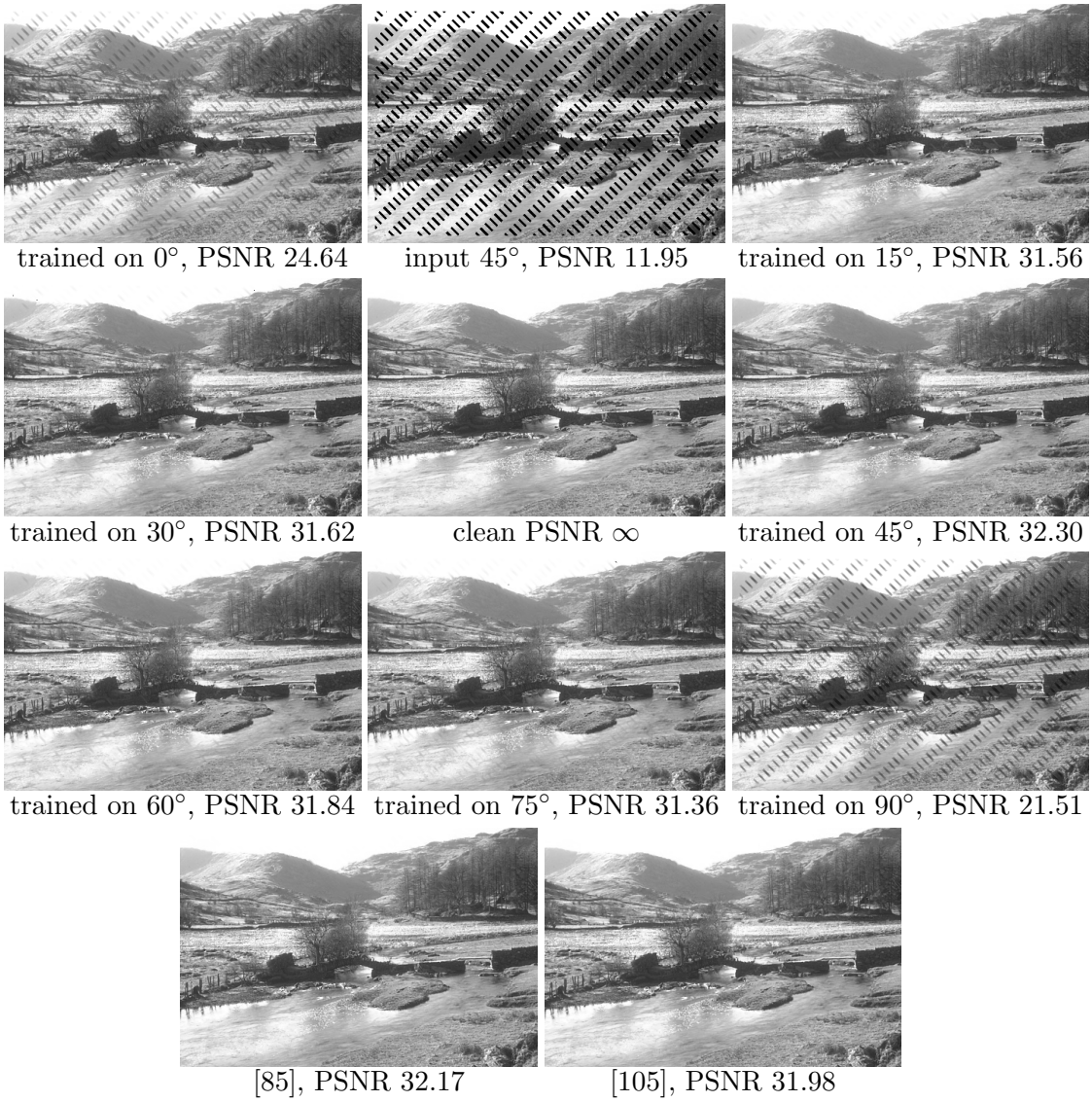
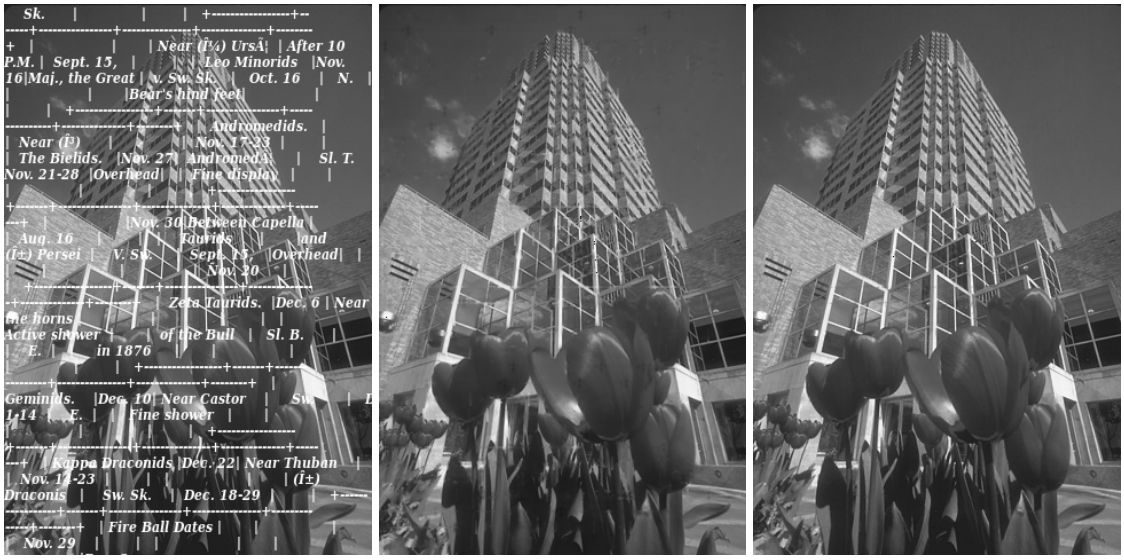


Figure 3.5: Performance of MLPs trained on different angles shown on one image. The input image was corrupted with bars rotated about 45°. The performance of the MLP which was trained on images corrupted with 45° bars is best. The results of the algorithms by [85] and [105] are also included. Fig. 3.4 shows the average PSNR value on ten corrupted images for MLPs trained on different orientations.

our approach performs better than the approaches by [21], [3], [105], [103], [119] and by [2]. Our result looks similar to the result of [85], but we are still able to recover more detailed information, e.g. the pole (marked with a green box in the corrupted image). One reason why we are better than the second best performing method [85] might be



(a) input image (b) trained w/o mask (c) trained with mask

Figure 3.6: The performance of the neural net improves if a mask is included in the training process. A mask helps the neural net to better learn the structure of the occluded pixels. It can be seen that in image (b), which was inpainted by a neural net, trained without the mask information, artifacts, especially in the sky, are visible.

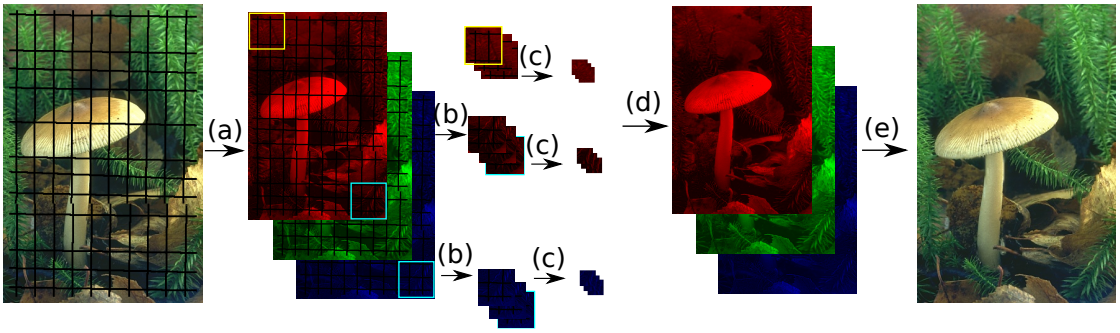


Figure 3.7: Inpainting a whole image: **(a)** *The corrupted image is separated into the three RGB color channels.* **(b)** *From each channel overlapping patches are extracted by a sliding window approach with stride size one.* **(c)** *Each patch is inpainted separately with the trained MLP.* **(d)** *The inpainted patches of each color channel are combined by adding them up and taking the arithmetic mean for each pixel.* **(e)** *The inpainted three RGB color channels are combined to form the color image.*

that we are able to use larger patch sizes. We used input patches of size 39×39 pixels, whereas [85] used a dictionary with patch size 9×9 pixels.

Note that for this example the original image was corrupted with some text of which the true mask is available. We used exactly this mask to corrupt images for the training set.

3.3.2 Comparison with horizontal/vertical lines

We perform an experiment with horizontal and vertical lines painted on an image. A $(16^2, 1024, 1024, 1024, 8^2)$ architecture was used. As can be seen in the two top rows of Fig. 3.9, we achieve a better PSNR than the methods by [105] and by [85]. This is especially due to the fact that the MLP is able to inpaint the stripes on the shirt in a much better way than the other algorithms. For the natural image below, differences are not salient, however we do achieve the highest PSNR.

3.3.3 Comparison against images from Xu and Sun [140]

Fig. 3.10 shows the results for two out of five images from [140] (Fig. 8 in that paper). In Sec. 3.3.5 we show the limitations of our proposed method on one of the other images. The neural net (with architecture $(39^2, 2047, 2047, 2047, 13^2)$) was trained with the given masks from [140]. On the Lena image we achieve better results (in terms of PSNR) than all other algorithms. On the Penguin image we are on par with [105].

3.3.4 Comparison for blind inpainting

With an MLP we are also able to inpaint images without the exact knowledge of the mask while other inpainting methods do require a binary mask as an input. Fig. 3.11 shows results for two images, on which random text was written with the same font and font-size as used for the training procedure of the MLP. The text which was written on the images was randomly selected from twelve English books downloaded at [48], concatenated comprising 97.709 lines, 1.027.960 words or 5.903.067 characters.

Note that the text written on the image is aliased, meaning it is not binary. Extracting a mask from such an image is rather tedious, as the optimal mask cannot be found by just thresholding the image. However, identifying the font and font-size is often possible and provides important information for the training process.

Optimally we would compare against [138] who also consider a deep learning approach to inpainting (but ignoring the mask based training). Unfortunately, we were not able to obtain their images nor their code, so we have to postpone this comparison. For this reason we compare only against the method of [105] which seems to be our closest competitor in terms of PSNR. Since the latter method requires the mask we additionally

Figure 3.8: Comparison on the image "New Orleans" from [3]. While [85] and our approach are close in terms of PSNR, the enlarged image detail (see green box in the corrupted image) shows that our approach is better able to recover the pole.

input, PSNR 17.36 [21], PSNR 32.26 [119], PSNR 32.66 [2], PSNR 32.91

[105], PSNR 35.31 [85], PSNR 35.32 ours, PSNR **37.34** original, PSNR ∞

input, PSNR 17.08 [21], PSNR 29.73 [105], PSNR 30.92 [119], PSNR 32.02

[2], PSNR 32.12 [85], PSNR 33.12 ours, PSNR **33.67** original, PSNR ∞

Figure 3.9: Comparison on an image with line artifacts. Note that the chest-part of the shirt shows fewer artifacts with our approach than it does with the other methods. However, for the natural image below (mushroom) we cannot see major differences, even though we achieve the highest PSNR.

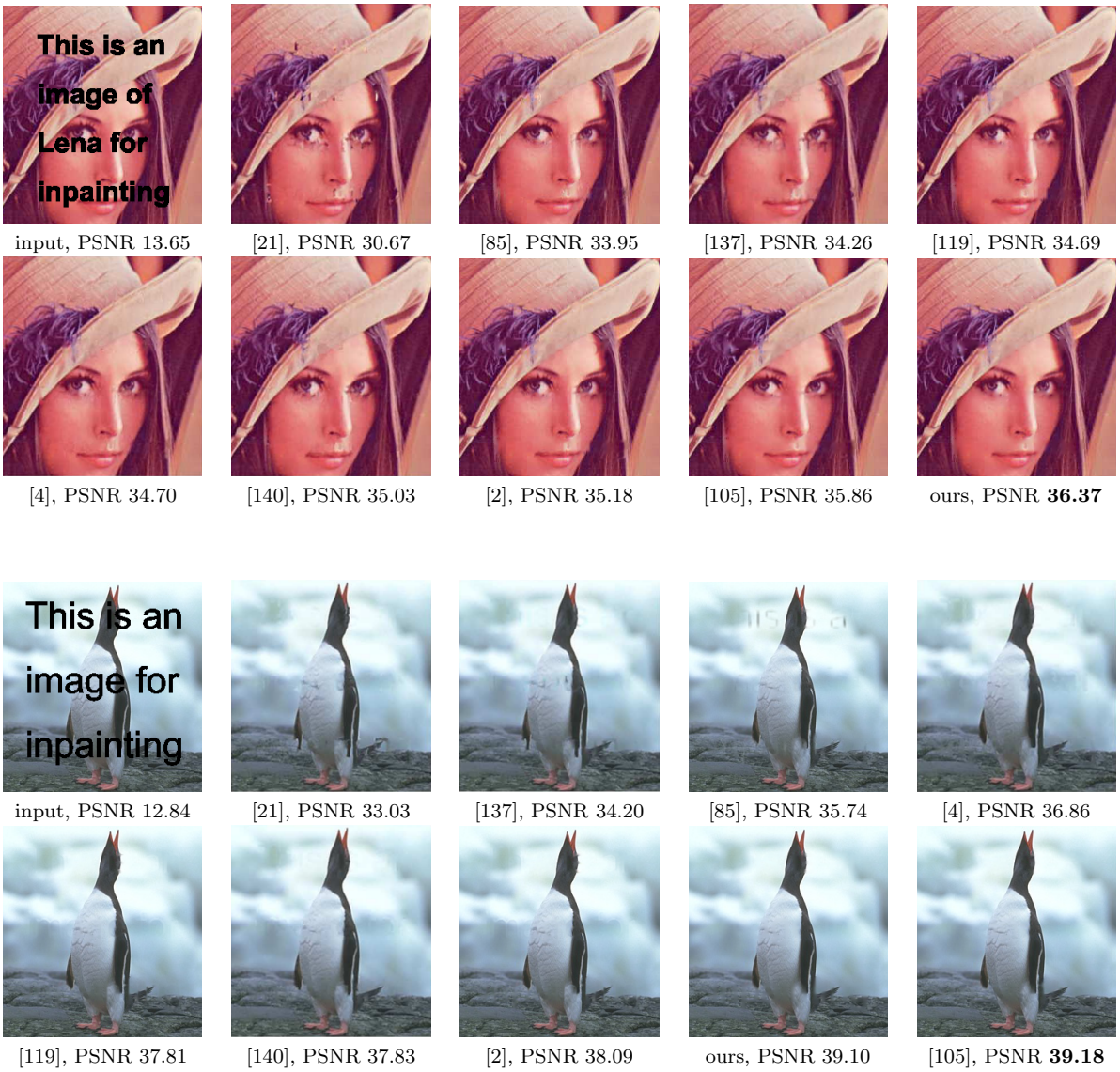
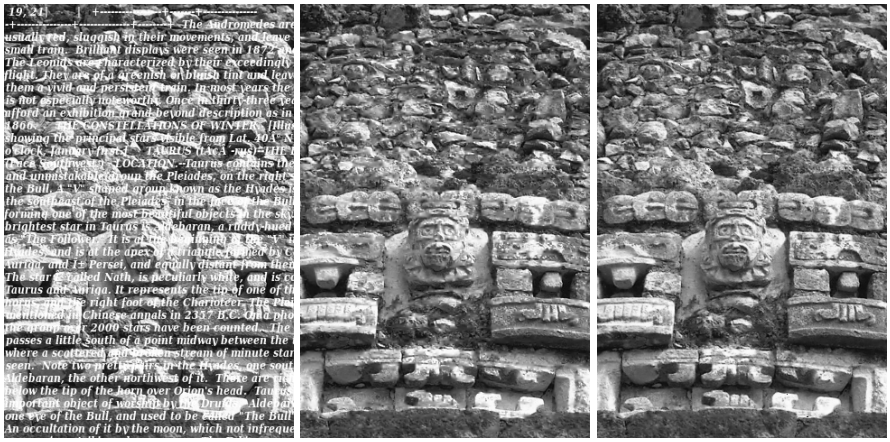
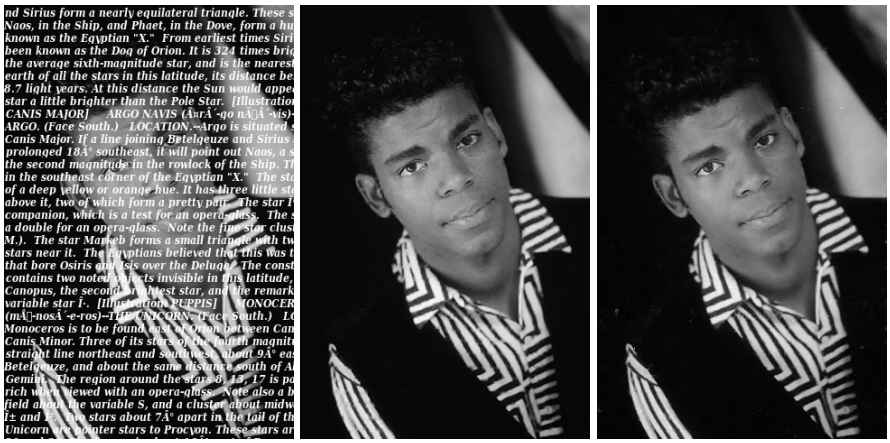


Figure 3.10: Comparison on two images from [140]. On Lena we achieve the highest PSNR, on the Penguin we are slightly worse than [105]. For [21, 137, 4, 140] we copied the results from [140]. Note that all PSNR scores refer to the full image and not only to the corrupted pixels (as done in [140]).



corrupted image [105], PSNR 23.99 ours, PSNR **26.09**



corrupted image [105] PSNR 32.13 ours, PSNR **34.22**

Figure 3.11: Inpainting an image without knowledge of the exact mask of each image. Only the font and font-size of the masks is known for training the neural net.

provided it to their method. Although our blind approach did not have access to the mask, it was able to reconstruct a better result.

We see that our method can be used to automatically inpaint images with super-imposed text or for automatic removing of watermarks or logos. It can also be used to batch inpaint several images which were damaged by the same type of corruption, without creating a mask for each corrupted image. Though this method does not perform as well as with the mask as an input (see Sec. 3.2.3), it is a good way to inpaint several images at once without the necessity of the user to determine the pixels of an image which are corrupted.

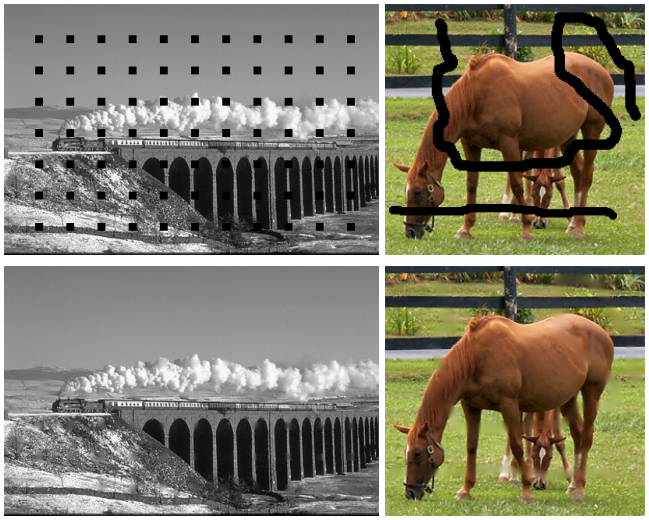


Figure 3.12: Limitations of our approach: large holes result in blurry output on textured regions, e.g. at the poles of the bridge. The inpainting in the homogeneous sky still works. Similar limitations can be seen on the horse image from [140]. On non-textured regions, the proposed method is able to inpaint the scribble, but on textured regions (e.g. the lawn) the inpainted result is blurry. Note that for the bridge the black squares are 10×10 *pixels and for the horse the line has width* 10 *pixels.*

3.3.5 Limitations

The proposed method performs well if the holes to be filled-in are small. If the holes are too large the inpainting result gets blurry, see left column of Fig. 3.12. While the sky is inpainted in an appropriate way (since it is smooth), the bridge pylons and the grass on the left part of the image are inpainted in a blurry way. Similarly, in the right column of Fig. 3.12 we see that across the grass the inpainted regions appear blurry. This is similar to the phenomenon also present with diffusion-based inpainting methods. This is probably due to the fact that the nonlinear filter learned by the MLP can only propagate a few pixels into the mask, but fails to propagate texture and fill-in larger regions. It seems that the neural net used in this approach is only able to learn to continue image information along isophotes.

3.4 Towards understanding the trained neural network

A common criticism of methods based on neural networks is that they work like a black box, i.e. even though they are able to reach state-of-the-art performance it remains unclear how the task is solved. To gain insight into how the trained neural network

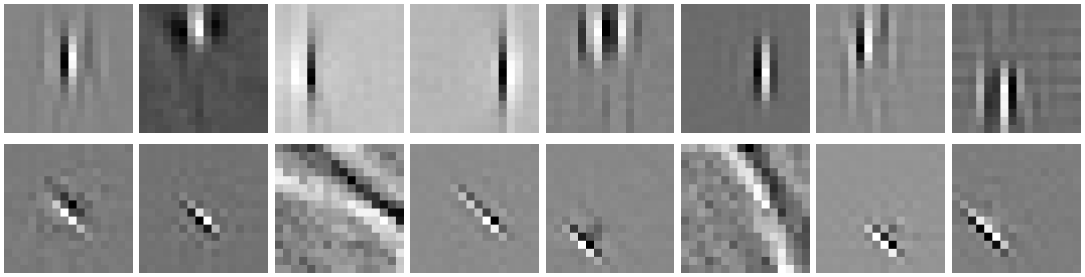


Figure 3.13: Selection of weights for the first layer of two MLPs trained on images corrupted with vertical bars (first row) or with 45 diagonal bars (second row). It can be seen that the filters learned by the MLPs are dependent on the distortion type (shape of the mask) given to the MLP.

achieves its performance, we study the feature generators for MLPs trained with different distortion types and look how the input feature depend on the shape of the masks.

3.4.1 Recognize and play back

Activation maximization [33] finds, for a given neuron in the last hidden layer, the respective input patch that maximizes that neuron's activation while constraining the input's L_2-norm. The activation is the sum of the bias and the direct input to the neuron from the previous layer. The idea behind activation maximization is that an input patch for that a given neuron has maximal activation, might be a good representation for understanding what the unit is doing.

The neurons in the last hidden layer are of interest since they are the *feature generators*, meaning the weights from these neurons to the output layer comprise the features that are linearly superimposed to reconstruct the image. We here analyze the neural net trained for Sec. 3.3.1. For Fig. 3.14, we additionally fixed the inpainting mask to be the letter 'e'. The bottom row of each block depicts typical 13x13 outputs of the feature generators, the top row the maximizing 39x39 input. For the Gabor-like features (some of them framed in red), the MLP tries to detect a continuation of the feature outside of the output size, which also motivates the importance of larger input than output patches. The NN also learned to just copy features to the output, if they are not obstructed by the mask (some of them framed in green). The feature framed in blue is impossible to reconstruct with the given mask, the result of the activation maximization is only L_2-constrained noise.

Intuitively speaking, it seems that the neural network is able to detect certain basic features which are then generated without missing pixels in the last layers. So image

features are detected even though pixels are missing, and played back including the missing information.

3.4.2 Input features depend on the masks

In Sec. 3.2.4 we showed that the inpainting performance depends on the masks used for training. To gain additional insight, we can also look at the weights for the first layer, which are shown in Fig. 3.13. The first row shows weights of a neural network trained on vertical bars, while the second rows was trained on diagonal bars. In general we see that the MLP learns mask specific feature detectors. It would be interesting to better understand the other features that appear; those currently have no obvious interpretation.

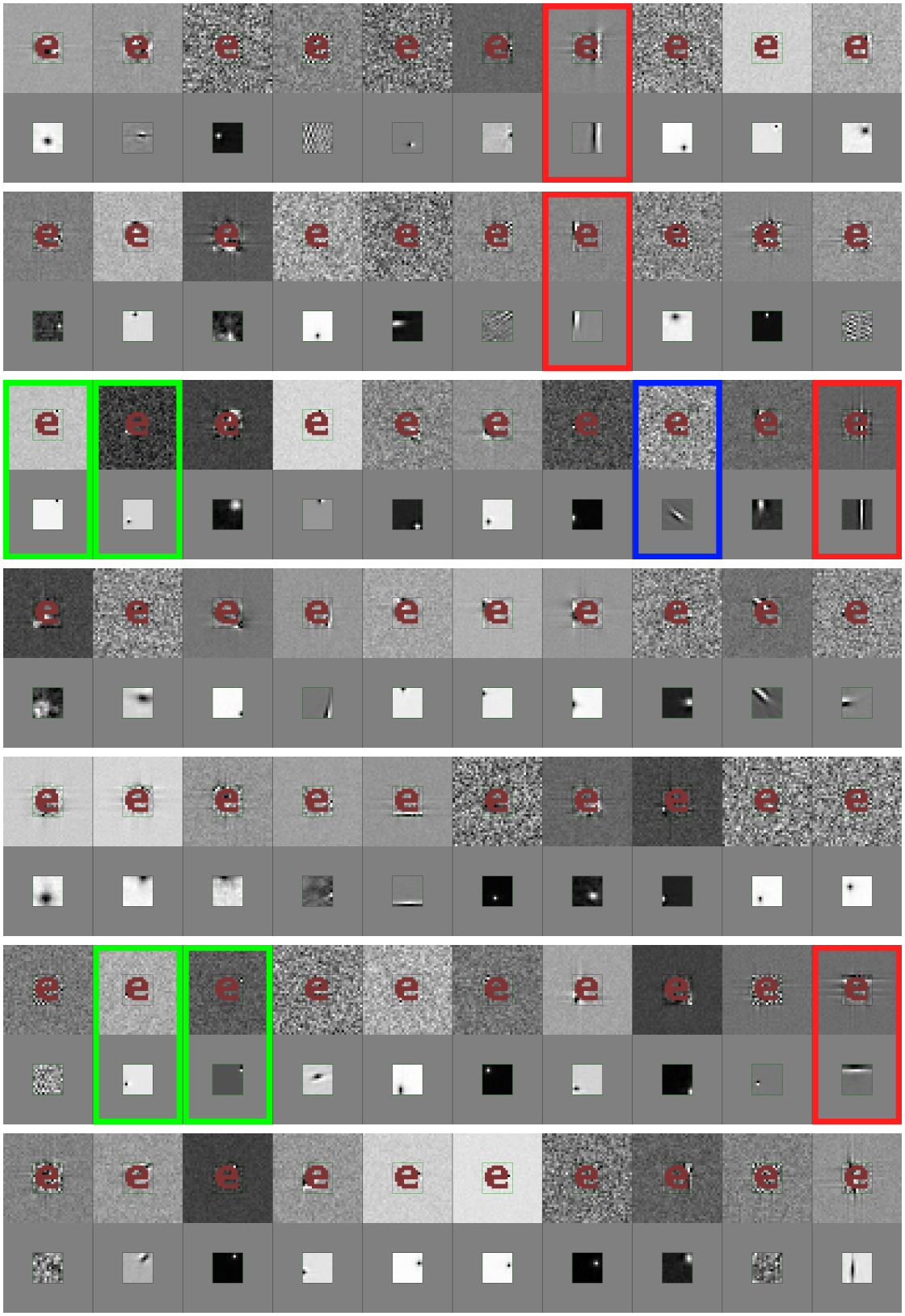


Figure 3.14: Input patterns (top row of each block) maximizing the activation of ten of the MLP's feature generators (bottom row) for the given inpainting mask 'e' (shown in red), the location of the output patch is marked with a green hairline in the bottom row. The MLP reconstructs features in the output image by trying to find a corresponding input pattern in regions not obstructed by the mask.

4

A Generative Model for Light Fields Applied to Depth Reconstruction

Chapter abstract Light field photography captures rich structural information that may facilitate a number of traditional image processing and computer vision tasks. A crucial ingredient in such endeavors is accurate depth recovery. We present a novel framework that allows the recovery of a high quality continuous depth map from light field data. To this end we propose a generative model of a light field that is fully parametrized by its corresponding depth map. The model allows for the integration of powerful regularization techniques such as a non-local means prior, facilitating accurate depth map estimation. Various priors suitable for other image processing tasks could be incorporated into our generative model.

Comparisons with previous methods show that we are able to recover faithful depth maps with much finer details. In a number of challenging real-world examples we demonstrate both the effectiveness and robustness of our approach.

The chapter is based on the following submitted publication:

[66] (submitted) R.Köhler, B. Schölkopf and M. Hirsch. Precise Depth Estimation from Light Field Images. International Conference on Computer Vision (ICCV) Workshop on Inverse Rendering. 2015.

4.1 Related Work and Contributions

Various algorithms have been proposed to estimate depth from a lightfield:

To construct the depth map of the sub-aperture image I_{s^*,t^*} a structure tensor on each EPI E_{y,t^*} $(y = 1, \ldots, Y)$ and E_{x,s^*} $(x = 1, \ldots, X)$ is used by [129] to estimate the slopes of the EPI-lines. For each pixel in the image I_{s^*,t^*} they get two slope estimations, one from each EPI. They combine both estimations by minimizing an objective function. In that objective they use a regularization on the gradients of the depth map to make the resulting depth map smooth. Additionally they encourage that object edges in the image I_{s^*,t^*} and in the depth map coincide. A coherence measure for each slope estimation is constructed and used to combine the two slope estimations. Despite not using the full light field, but only the sub-aperture images $I_{s^*,t}$ $(t = 1, \ldots, T)$ and I_{s,t^*} $(s = 1, \ldots S)$, for a given (s^*, t^*), they achieve appealing results.

In [62] a fast, GPU based, algorithm for light fields of about 100 high resolution DSLR images is presented. The algorithm also makes use of the lines appearing in an EPI. The rough idea for a 3D lightfield is as follows: to find the disparity of a pixel (x, s) in an EPI, its RGB pixel value is compared with all other pixel values in the EPI that lie on a slope that includes this pixel. They loop over a discretized range of possible slopes. If the variance in pixel color is minimal along that slope, the slope is accepted. For a 4D lightfield not a line is used, but a plane.

In [118] the refocusing equation described in [92] is used to discretely sheer the 4D lightfield in order to get correspondence and defocus clues, which are combined using a Markov Random Field. A similar approach is used by [28].

[52] suggest an optimization approach, which makes use of the fact that the sub-aperture images of a light field look very similar: sub-aperture images are warped to a reference image by minimizing the rank of the set of warped images. In their preceding work [53] the same authors match all sub-aperture images against the central view using a generalized stereo model, based on variational principles. [100] uses a multi-focus plenoptic camera and calculates the depth from the raw images using triangulation techniques. [1] uses displacement estimation between sub-aperture image pairs and combines the different estimations via a weighted sum, where the weights are proportional to the coherence of the displacement estimation.

We propose an algorithm in this chapter, which is capable of producing a continuous depth map from a recorded light field and hence provides more accurate depth labels - than algorithms which return discrete labels like [62, 118, 28], especially for fine structures.

Contributions of this chapter: While a number of different approaches for depth estimation from lightfield images exists, we derive to the best of our knowledge for the first time a fully generative model of EPI images that allows a principled approach for depth map estimation and the ready incorporation of informative priors. In particular, we show

1. a principled, fully generative model of light fields, that enables the estimation of continuous depth labels (not discrete).
2. an efficient gradient-based optimization approach that can be readily extended with additional priors. In this work we demonstrate the integration of a powerful non-local means prior term.
3. favorable results on a number of challenging real-world examples. Especially at object boundaries, significantly sharper edges can be obtained.

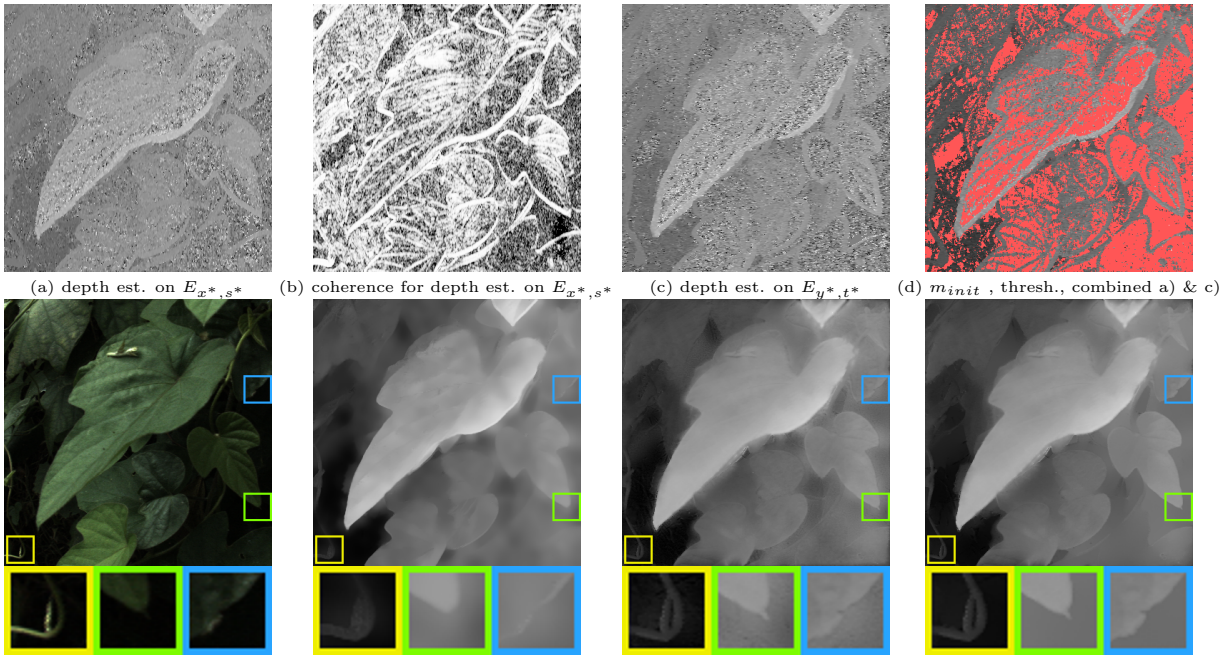
4.2 Overview

The presented algorithm computes the depth map of a selected sub-aperture image from a light field. In this chapter - without loss of generality - we always chose the central sub-aperture image. Our algorithm works in a two step procedure: As a first step a rough estimation of the depth map is computed locally (Sec. 4.3). This serves as an initialization to the second step (Sec. 4.4), in which it gets refined by using a gradient based optimization approach.

The following implicit assumptions are made by our algorithm:

- There exists no occlusions in the scene. This means that lines in the EPI images do not cross. This is a sensible assumption, especially for the Lytro camera, as the baseline is not very large, as also stated by [25, 24].
- All objects in the scene are Lambertian, a common assumption which is also used by [130, 108, 62, 13].
- Object boundaries in the RGB image coincide with (abrupt) changes in the depth image. This assumption is also explicitly made in [95, 37].

Not all of these assumptions are perfectly met in real scenes. Despite relying implicitly on those assumptions, our algorithm works well on real world images, as we will demonstrate in Sec. 4.6.



(a) depth est. on E_{x^*,s^*} (b) coherence for depth est. on E_{x^*,s^*} (c) depth est. on E_{y^*,t^*} (d) m_{init} , thresh., combined a) & c)

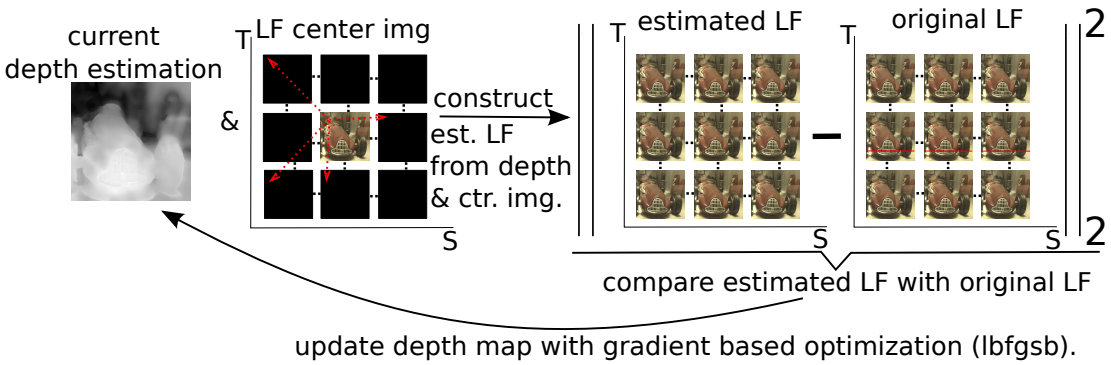
(e) central sub-aperture image (f) propagation of (d) (g) result without prior (h) result with NLM prior

Figure 4.1: Overview of our method on an image by [118], best viewed on screen. Note: each image was normalized to [0,1], hence gray values between images may vary. Images (a)&(c) show rough depth maps, computed using the first part of the local depth estimation method of [129], (a) on the EPI E_{x^*,s^*}, and (c) on the EPI E_{y^*,t^*}. (b) Coherence map for the depth estimation on the EPI E_{x^*,s^*}. (d) Those two depth maps are thresholded using the coherence map of [129] and then combined. It is visible that coherent depth values are only located at edges while non-coherent depth values (in red) are found in smoother regions. (f) To fill the red missing pixels the coherent depth values are propagated using a NLM approach. (g) Our result without prior shows more details than (f), but also introduces speckles in the depth estimation. (h) A NLM prior is able to get rid of those speckles while preserving the finer details.

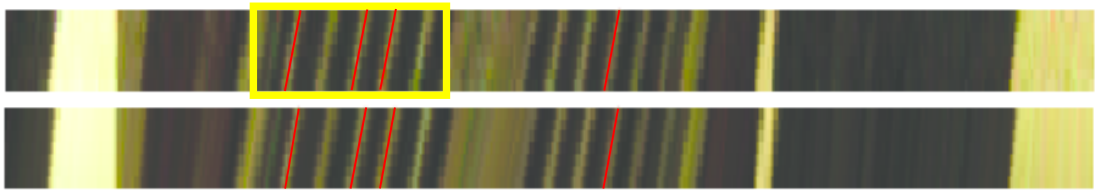
4.3 Rough depth map estimation using a Non local means regularizer

To get an initialization for the depth map, we adopt the initial part of the depth estimation algorithm of [129] [1]: the local depth estimation on EPIs. It estimates the slopes of the lines visible in the epipolar image using the structure tensor of the EPI. It only uses the epipolar images of the angular center row $\left(E_{y,t^*}\right.$ with $\left.t^* = T/2, y = \{1, \dots Y\}\right)$

[1] We use the given default values sigma=1.0, tau=0.5.



(a) Overview of the refinement step visualized with sub-aperture images, best viewed on screen: The estimated light field (LF) is constructed by shifting pixels from the central sub-aperture image to the other sub-aperture images. The shifting depends on the current depth value of the shifted pixel and the distance to the sub-aperture image. The objective is to minimize the squared L2 distance between the estimated LF and the original LF. To update the depth map, the gradient based optimizer L-BFGS-B is used.



(b) **Top**: Original EPI E_{y^*,t^*} from 7 sub-aperture images using the row in each image, which is highlighted in red in the original LF above. **Bottom**: Estimated EPI constructed just from the central row by shifting center-row pixels according to its depth value to the other rows. Note that for better visualization, each pixel (1×1) was stretched to twice its height (2×1). Four red lines are overlaid to visualize the emergence of lines in the EPI.

(c) Overview of the refinement step visualized with an EPI.

Figure 4.2: Overview of the refinement step.

and angular center column $\left(E_{x,s^*}\right.$ with $s^* = S/2, x = \{1, \ldots X\}\left.\right)$ and yields two rough depth estimates for each pixel in the central image, see Fig. 4.1 (a,c). Besides, it also returns a coherence map, which we denote by $C(\cdot)$ (Fig. 4.1 (b)) . A value within the coherence map is high if the algorithm is sure about its depth estimation.

This approach returns consistent estimates at image edges, but provides quite noisy estimates at smoother regions, see Fig. 4.1 (a,c). We take the two noisy depth estimates and threshold them by setting depth values with a too low coherence value to "non-defined" (red pixels in Fig. 4.1 (d). This gives us an initial estimate for the depth values

at the edges. After thresholding the two estimates are combined: if for a pixel in the depth map both estimates (on E_{y,t^*} and E_{x,s^*}) have coherent values, we will take the depth estimation with the higher coherence value. The result is shown in Fig. 4.1 (d). This rough depth map estimate is denoted by m_{init}. Due to an initial smoothing step in the implementation of [129] edges are not preserved at the correct location and appear smeared out.

To propagate information into regions of non-coherent pixels (red pixels in Fig. 4.1 (d)) of the depth map and to improve the localization of the edges, we solve the following optimization problem

$$\underset{m}{\operatorname{argmin}} \sum_{p} \sum_{q \in N(p)} w_{pq} \Big[\big(m(p) - m(q)\big)^2 + C(p) \big(m_{init}(p) - m(q)\big)^2 \Big]. \tag{4.1}$$

where w_{pq} is a weighting term on the RGB image, which will be defined later in Eq. (4.4): w_{pq} is close to 1 if the 3×3 window around p and q have similar color and gradient values. $N(p)$ are the neighboring pixels around p, e.g. in a 11×11 window, with p in the center of it. The term $w_{pq}(m(p) - m(q))^2$ ensures that pixels in $N(p)$ have similar depth values as p, if they are similar in the RGB image (high w_{pq}). The term $w_{pq} \cdot C(p)(m_{init}(p) - m(q))^2$ enforces that a pixel with high coherence $C(p)$ propagates its depth value $m_{init}(p)$ to neighboring pixels q, which are similar in the RGB image. The result of this propagation step can be seen in Fig. 4.1 (f). Eq. (4.1) is minimized using L-BFGS-B [12].

4.4 Refinement step

The refinement step is based on the observation that the sub-aperture images can be almost entirely explained and predicted from the center image $I := I_{S/2,T/2}$ alone, namely by shifting the pixel from the image I along the lines visible in the epipolar image. This has already been observed by others [44, 58]. Fig. 4.2c visualizes the case for an EPI E_{y^*,t^*}.

To make full use of the recorded 4D lightfield the central image pixels are not only moved along a line, but along a 2D plane. For a fixed s^* or t^* this 2D plane becomes a line visible in the respective EPI E_{x^*,s^*} or E_{y^*,t^*}. Note that the slopes of the line m_{x_c,y_c} going through the center pixel (x_c, y_c) are the same in E_{y_c,t^*} and E_{x_c,s^*}.

When moving a central image pixel $I(x_c, y_c)$ along the 2D plane with given slope m_{x_c,y_c}, its new coordinates in the sub-aperture image $I_{S/2+d_s,T/2+d_t}$ become $(x_c + m_{x_c,y_c} d_s, y_c + m_{x_c,y_c} d_t)$, where d_s and d_t denote the angular distances from the center sub-aperture image to the target sub-aperture image, formally $d_s = s - S/2, d_t = t - T/2$. The predicted light field that arises from shifting pixels from the center sub-aperture image to all other sub-aperture views is denoted by $\widetilde{L}_m(s, t, x, y)$.

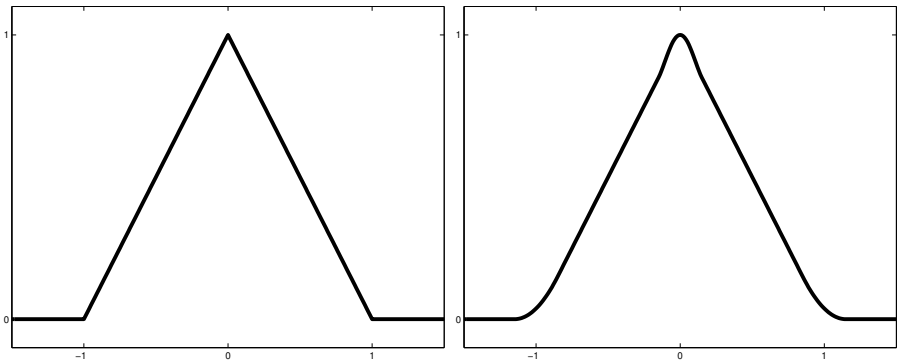
The influence of pixel $I(x_c, y_c)$ on pixel $\widetilde{L}_m(s,t,x,y)$ is determined by the distances $x-(x_c+m_{x_c,y_c}d_s)$ and $y-(y_c+m_{x_c,y_c}d_t)$. Only if the absolute values of both distances are smaller than one, the pixel $I(x_c, y_c)$ influences $\widetilde{L}_m(s,t,x,y)$, see Fig. 4.4. Hence, a pixel $\widetilde{L}_m(s,t,x,y)$ can be computed as:

$$\begin{aligned}\widetilde{L}_m(s,t,x,y) &= \sum_{x_c}\sum_{y_c} I(x_c,y_c)w(m,s,t,x_c,y_c) \\ &= \sum_{x_c=1}^{X}\sum_{y_c=1}^{Y} I(x_c,y_c)\Lambda\big(x-(x_c+m_{x_c,y_c}d_t)\big)\cdot\Lambda\big(y-(y_c+m_{x_c,y_c}d_s)\big) \quad (4.2)\end{aligned}$$

where $\Lambda : \mathbb{R} \to [0,1]$ is a weighting function and denotes the differentiable version of the triangular function defined as

$$\bar{\Lambda}(x) = \begin{cases} 0 & \text{if } x < -1 \vee x > 1 \\ 1+x & \text{if } -1 \le x < 0 \\ 1-x & \text{if } 0 \le x \le 1 \end{cases}$$

To make $\bar{\Lambda}(x)$ differentiable, we approximate each of the three kinks at $x=-1$, $x=0$, $x=1$ in a $\delta-$area ($\delta = 0.001$) around the kink with a polynomial. At $x=-1$ and $x=1$ a polynomial of degree 3 was used, at $x=0$ a polynomial of degree 4. The resulting smooth and differentiable function is denoted by $\Lambda(x)$, see Fig. 4.3.



(a) triangular function (b) differentiable triangular function

Figure 4.3: (a) Triangular Function and (b) a differentiable smooth version of the triangular function with $\delta = 0.15$. (b) was approximated at each of the three kinks at $x=-1, x=0$ and $x=1$ with a polynomial.

To allow for sub-pixel shifts, we use the following bilinear interpolation scheme: As shown in Fig. 4.4 the intensity value of the pixel $\widetilde{L}_m(s,t,x,y)$ is the weighted sum of all overlapping pixels (in Fig. 4.4 the four colored pixels). The weight of each pixel

is the area of overlap with the pixel $\widetilde{L}_m(s,t,x,y)$, which is $\Lambda\big(x-(x_c+m_{x_c,y_c}d_t)\big)\cdot\Lambda\big(y-(y_c+m_{x_c,y_c}d_s)\big)$.

If only a small patch (e.g. 2×2) is considered, one can assume that the difference in depth between the four pixels is negligible resulting in the same translation for each pixel. For this case our scheme reduces to the known bilinear interpolation , see Fig. 4.5.

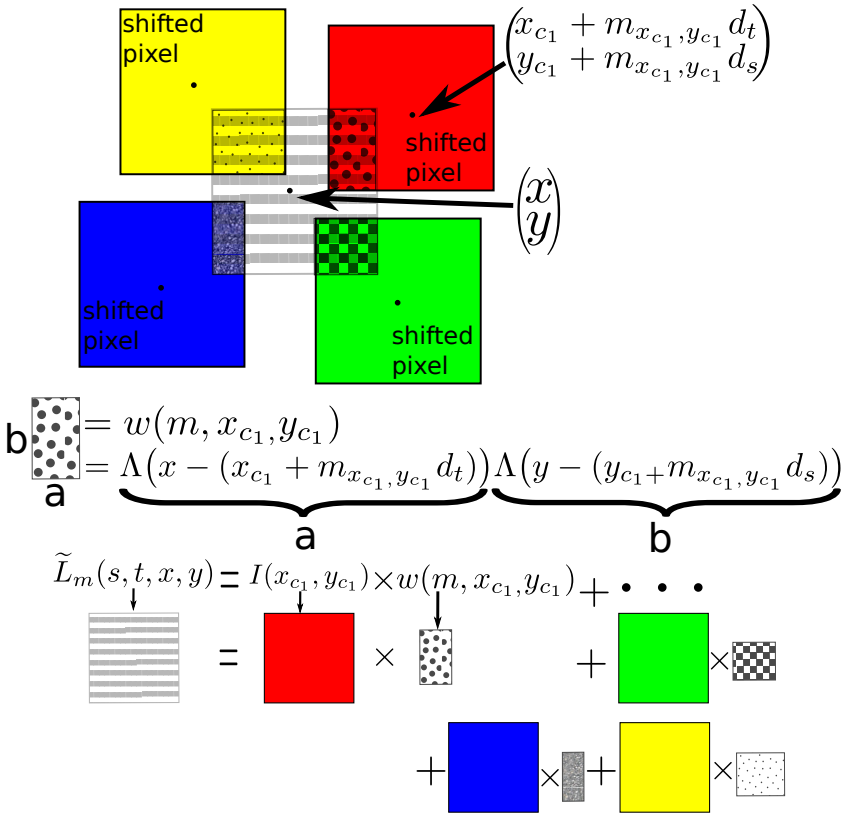


Figure 4.4: Modified Bilinear Interpolation: The color of the estimated pixel is determined by using a weighted sum of the shifted pixels that overlap with it. The weight of each pixel is the area of overlap with the estimated pixel.

Note that to construct the pixel $\widetilde{L}_m(s,t,x,y)$ it is of course not necessary to loop over all central image pixels, but only over those which are in a window around (x,y). We define

$$\begin{aligned} min\_x_c &:= \max(\min(x+\min(m)d_t-1),1), \\ max\_x_c &:= \min(\max(x+\max(m)d_t+1),X) \end{aligned}$$

and analogously min_y_c and max_y_c. Note that it is necessary to subtract and add one, because the triangular function Λ is approximated at the kinks. Then Eq. (4.2) can be

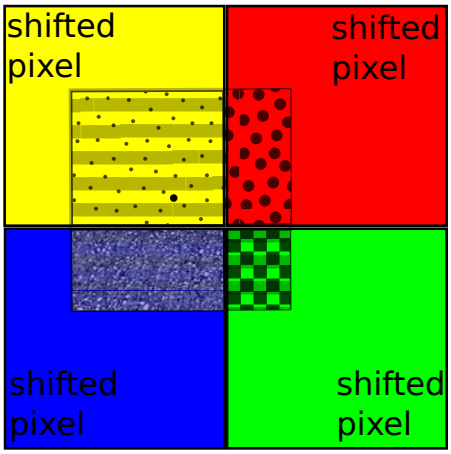


Figure 4.5: Bilinear Interpolation: The no-occlusion assumption mostly holds for small patches of the image, resulting in the normal bilinear interpolation. Here each pixel of the 2×2 patch is translated/shifted in the same way, as each pixel has the same depth value.

reformulated to

$$\widetilde{L}_m(s,t,x,y) = \sum_{x_c=min\_x_c}^{max\_x_c} \sum_{y_c=min\_y_c}^{max\_y_c} I(x_c,y_c)w(m,x_c,y_c)$$

Also note that by the no-occlusion assumption a normalization term is not necessary in Eq. (4.2).

To get a refined depth estimation map we optimize the following objective function:

$$m = \underset{m}{\operatorname{argmin}} ||\widetilde{L}_m - L||_2^2 + \lambda\, R(m) \tag{4.3}$$

with $R(m)$ being a regularizer which will be described in more detail in the remainder of this section

Non local means (NLM) regularizer

The NLM regularizer was first proposed in [9]. We define it as

$$R_{NLM}(m) = \sum_p \sum_{q \in N(p)} w_{pq}(m(p) - m(q))^2,$$

with $N(p)$ being the search window around a pixel p, e.g. a 11×11 window and w_{pq} is the weight expressing the similarity of the pixels p and q. We define w_{pq} as

$$w_{pq} = \exp\left(- \sum_{\substack{p' \in N'(p) \\ q' \in N'(q)}} \frac{[I(p') - I(q')]^2}{\sigma^2_{Color}} + \frac{[\nabla_{p,p'} I - \nabla_{q,q'} I]^2}{\sigma^2_{Grad}} \right) \tag{4.4}$$

where $\nabla_{p,p'} I := I(p) - I(p')$ is the image gradient at pixel position p. σ^2_{Color} and σ^2_{Grad} are the variance in the color and gradient values of the image. $N'(p)$ is the neighborhood around pixel p. In all experiments N' was a 3×3 window. w_{pq} is 1 if the 3×3 patches around p and q are completely identical and close to 0, if they differ a lot.

4.5 Implementation Details

The current implementation is in MATLAB. For optimization we used the MATLAB interface by Peter Carbonetto [12] of the gradient based optimizer L-BFGS-B [144].

The only code-wise optimization we performed is to implement some of the computational expensive parts in C (MEX) using the multiprocessing API openmp[2] . Current runtime for computing the depth map from the $7 \times 7 \times 375 \times 375 \times 3$ light field from the Lytro camera is about 270 seconds = 4.5 minutes on a 64bit Intel Xeon CPU E5-2650L 0 @ 1.80GHz architecture using 8 cores (226 seconds with 16 cores). Runtime for the $17 \times 17 \times 1280 \times 960 \times 3$ Stanford truck image is about 338 minutes on the same architecture with 8 cores and 274 minutes with 16 cores. Runtime for the $17 \times 17 \times 768 \times 1024 \times 3$ Stanford amethyst image on the same architecture is 162 minutes with 16 cores. For the exact parameter settings used in the experiments and for a short explanation of the influence of the parameter settings to the depth map estimation see Sec. 4.7.

4.6 Experimental Results

Comparison on Lytro images

In Fig. 4.6 we compare our results with those of [118]. The resolution of the Lytro light field is $S \times T = 7 \times 7$, and the resolution of each sub-aperture image is $X \times Y = 375 \times 375$ pixels, much lower than the resolution of the Stanford light field images. Note that our sub-aperture image size (375×375) differs from the image size from [118] (311×362). We decoded the raw Lytro image with the algorithm by [26][3], whereas [118] developed

[2] http://openmp.org .

[3] The images have not been gamma compressed nor rectified.

an own decoding procedure. Our algorithm is able to recover more details, better defined edges and has less speckles.

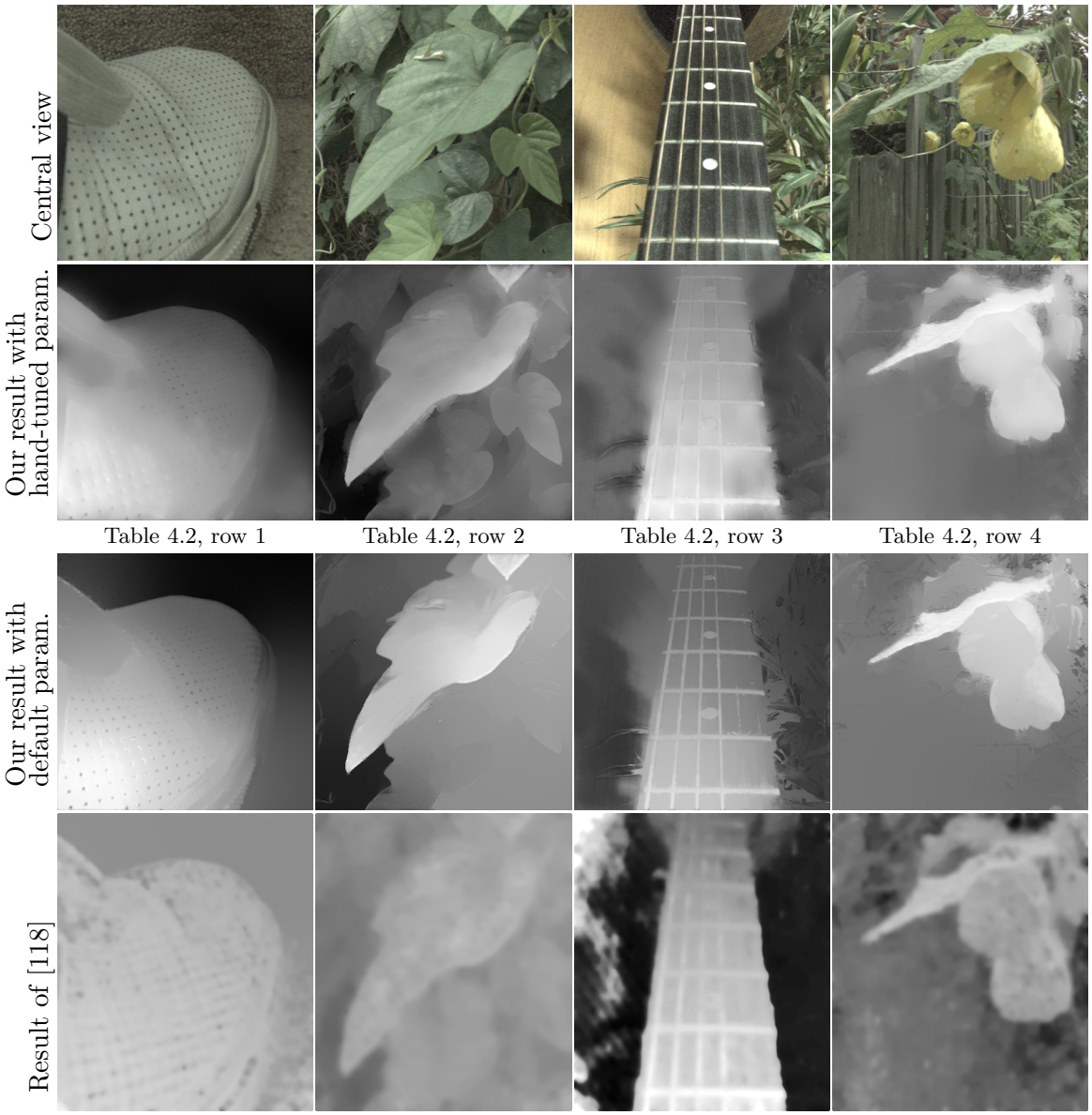


Figure 4.6: Comparison on Lytro images from the recent work [118], best viewed on screen. Our algorithm is able to recover finer details and produces fewer speckles in the depth map. The image sizes of our result and of [118] differ due to the use of different raw-decoding algorithms. The default parameter setting can be found in Table 4.1. The hand-tuned parameters for each image can be found in Table 4.2 in the indicated row.

To demonstrate the robustness of our approach we also show results on a few challenging Lytro images that we captured ourselves in Fig. 4.7.

Comparison on the Stanford light field dataset

In Fig. 4.8 we compare our method on images of the Stanford light field dataset [113]. The light field has a resolution of $S \times T = 17 \times 17$ sub-aperture views.

Each sub-aperture image in the truck example has a resolution of $X \times Y = 1280 \times 960$ pixels. Compared to the already visually pleasing results of [129] and [62], we are able to recover finer and more accurate details (see closeups in Fig. 4.8). Additionally the edges of the depth map match better with the edges of the RGB image.

In the amethyst example each sub-aperture image has a resolution of $X \times Y = 768 \times 1024$ pixels. Again, the contours of the amethyst within the RGB image match well with the contours of our depth map. Note, that the depth value in the background cannot be estimated due to the lack of available edge information, it can only be propagated. In our estimation it is propagated incorrectly, it should be dark.

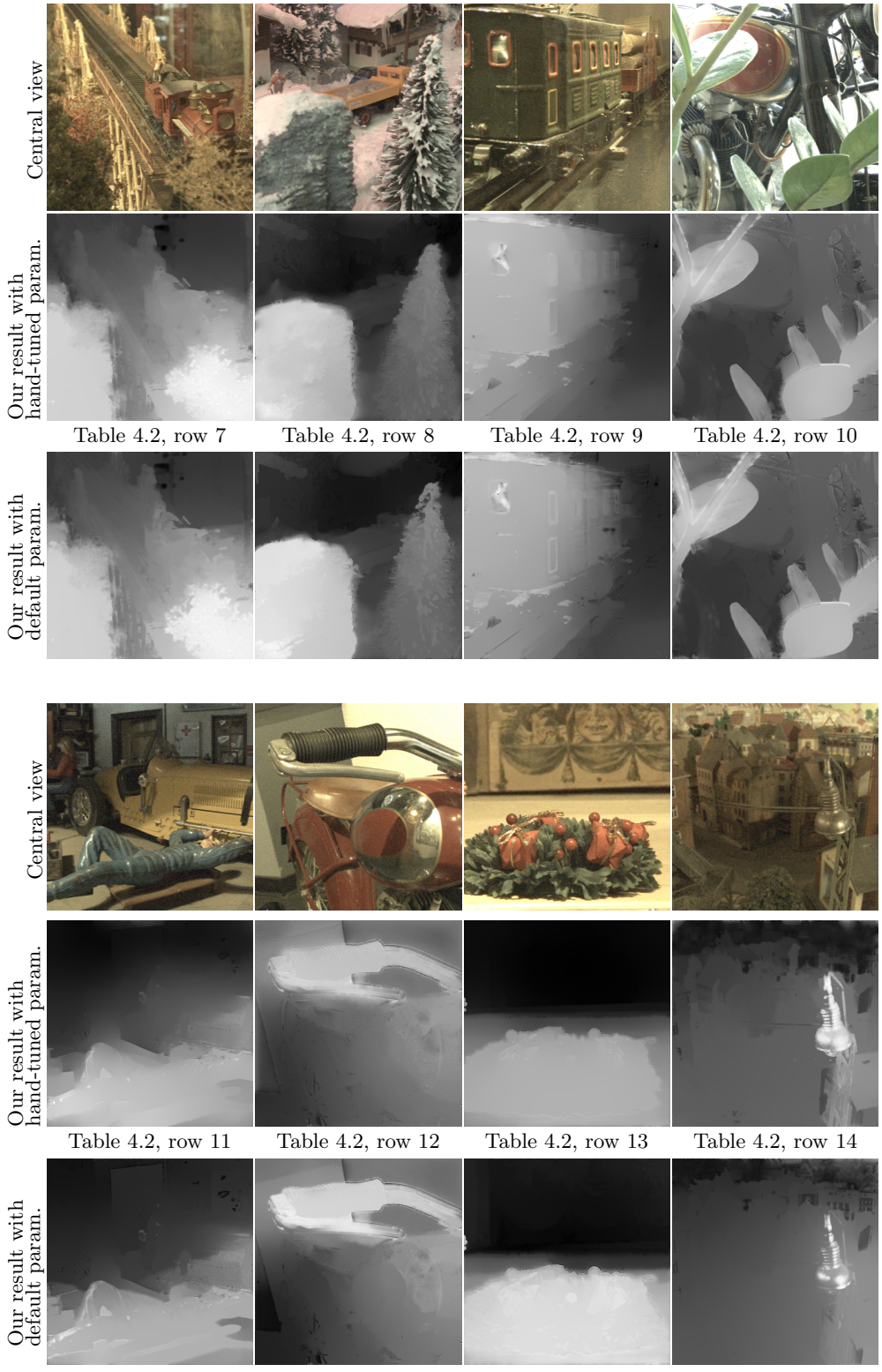
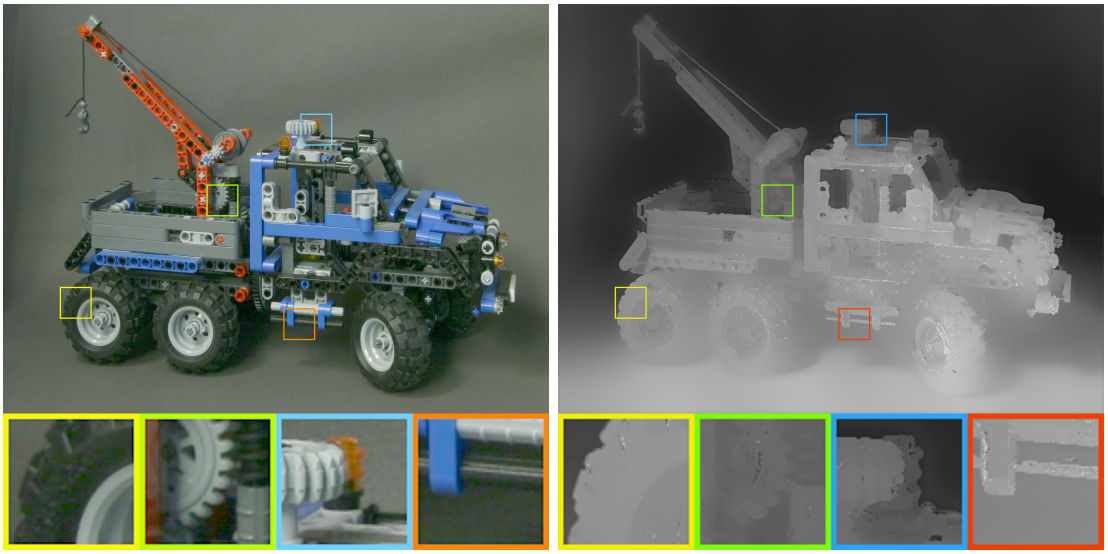
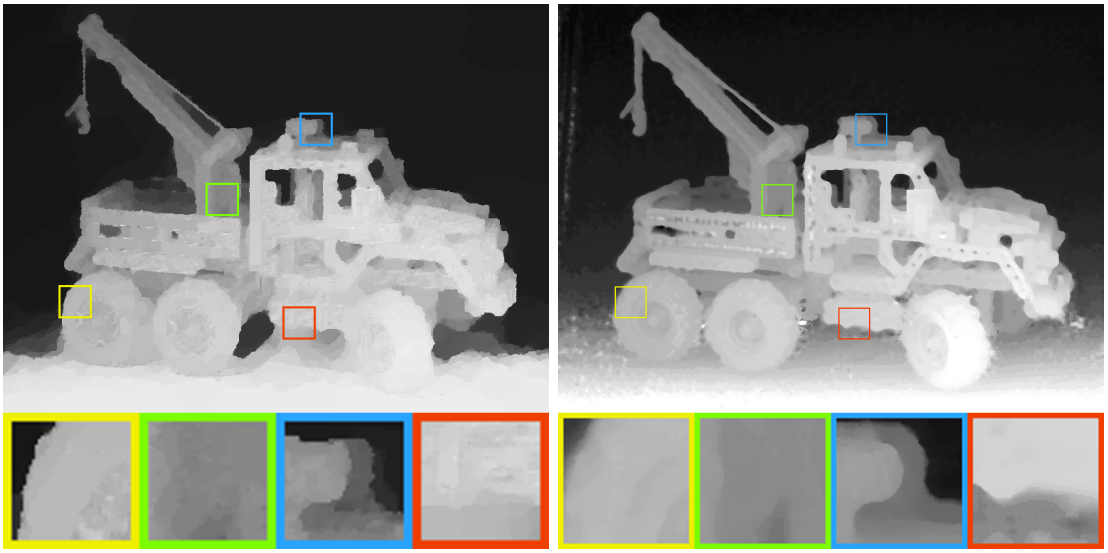


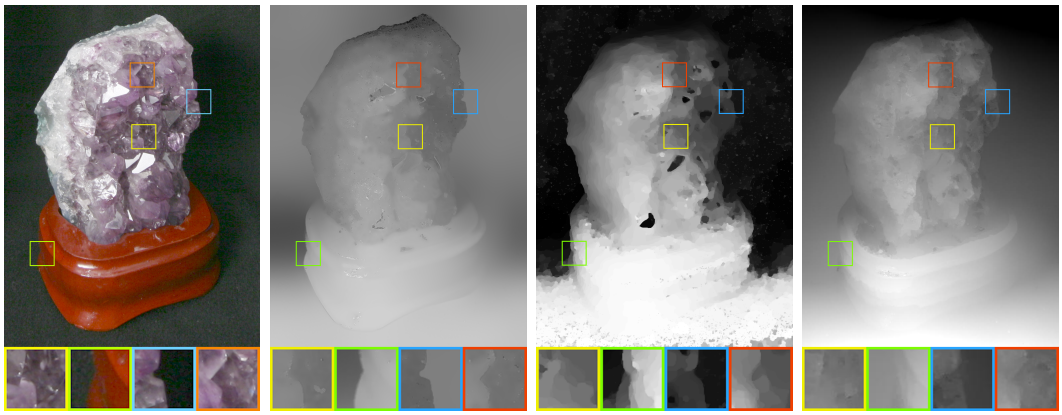
Figure 4.7: Results on Lytro images taken by ourselves with default and hand-tuned parameters. The default parameter setting can be found in Table 4.1. The hand-tuned parameters for each image can be found in Table 4.2 in the indicated row. Note that the Lytro images are rather noisy, and often violate the Lambertian assumption, due to reflective surfaces. Nonetheless, our method manages to recover faithful depth maps with fine details.



(a) central image (b) our result



(c) result of [129] (d) result of [62]



(a) central image (b) our result (c) result of [129] (d) result of [52]

Figure 4.8: Comparison on images of the Stanford light field dataset [113], best viewed on screen. We are able to recover finer details and our depth boundaries match better with the RGB image boundaries. Images from [62], [129] and [52] are taken from the respective paper. The image resolutions are 1280×960 *(truck) and* 768×1024 *(amethyst), except for [129], where they are* 768×576 *(truck) and* 720×960 *(amethyst).*

4.7 Parameter Settings

The following parameters need to be set for computing the rough depth map, see Sec. 4.3:

- coherence threshold (cohThr)
- search window size $N(p)$
- σ^2_{Color}
- σ^2_{Grad}

The following parameters need to be set for the refinement step, see Sec. 4.4:

- λ
- search window size $N(p)$
- σ^2_{Color}
- σ^2_{Grad}

A good set of default parameters that yields reasonable results on all Lytro images, as evident from the images in Fig. 4.6 and Fig. 4.7, is summarized in Tab. 4.1.

	Rough Estimation				Refinement			
image	cohThr	$N(p)$	σ^2_{Color}	σ^2_{Grad}	λ	$N(p)$	σ^2_{Color}	σ^2_{Grad}
Lytro	0.8	31	0.07	0.06	0.1	31	0.07	0.06

Table 4.1: Default parameter settings for Lytro images.

Table 4.2 summarizes the hand-tuned parameter settings that were used for optimal depth map results.

Some comments and insights on these parameters and their settings:

- The value of the coherence threshold was set to either 0.5 for the images from the Stanford dataset and to 0.8 for the Lytro images. The images from the Stanford dataset are less noisy and hence lower coherence values are trustworthy.

	Rough Estimation				Refinement			
image	cohThr	$N(p)$	σ^2_{Color}	σ^2_{Grad}	λ	$N(p)$	σ^2_{Color}	σ^2_{Grad}
1 (shoe)	0.8	11	0.05	0.01	0.1	11	0.1	0.1
2 (leaves)	0.8	11	0.05	0.01	0.1	11	0.005	0.1
3 (guitar)	0.8	11	0.05	0.01	0.1	11	0.3	0.3
4 (flower)	0.8	11	0.05	0.1	0.1	11	0.05	0.05
5 (truck)	0.5	61	0.05/8	0.01/8	0.1	11	0.005	0.005
6 (amethyst)	0.5	61	0.05/4	0.01/4	0.1	11	0.001	0.001
7 (bridge)	0.8	21	0.1	0.02	1	21	0.5	0.01
8 (winterland)	0.8	11	0.05	0.1	0.1	31	0.01	0.01
9 (train)	0.8	41	0.1	0.02	3	11	0.1	0.1
10 (bike)	0.8	21	0.1	0.02	3	11	0.1	0.1
11 (garage)	0.8	41	0.1	0.2	1	11	0.1	0.1
12 (bike-brake)	0.8	41	0.1	0.02	0.1	11	0.1	0.1
13 (wreath)	0.8	41	0.1	0.02	0.5	21	0.5	0.5
14 (houses)	0.8	11	0.05	0.1	1	31	0.01	0.01

Table 4.2: Hand tuned parameter settings used to compute the depth map on each image.

- The parameter $N(p)$, the search window size of the NLM regularizer, was set to 11×11 pixels for most of the Lytro images. When the regions with unreliable depth information (the red pixels in Fig. 4.1(d) were too large for a window of size 11×11 pixels, we increased the search window accordingly. Note that one possibility to free the user from setting the parameter $N(p)$ manually would be to adjust the window size $N(p)$ depending on the accumulated weights $\sum_{q \in N(p)} w_{pq}$, i.e. if there are not enough 3×3 windows, which are similar to the 3×3 window around pixel p, the window size $N(p)$ is increased.
- λ determines the weight of the NLM regularizer. It was set between $[0.1, 3]$, most often 0.1. The higher λ the smoother the result.
- σ^2_{Color} and σ^2_{Grad} are parameters that determine the variance of the RGB color image in the window $N(p)$ and the variance in the gradients. The higher they are set, the smoother the result.

Following parameters were kept constant throughout all experiments:

- maximal number of iterations for L-BFGS-B for solving Eq. (4.1) (rough depth estimation): 30.

- maximal number of iterations for L-BFGS-B for solving Eq. (4.3) (refinement step): 100.
- $N'(p)$ in w_{pq} (Eq. (4.4)) was set to a window of size 3×3 pixels.

4.8 Limitations

Our algorithm relies on the no-occlusion assumption, which holds for most parts of the image, but might be violated for pixels at object boundaries. For such pixels it is hard to fit a line in the EPI images and therewith estimate the right depth value, see Fig. 4.9.

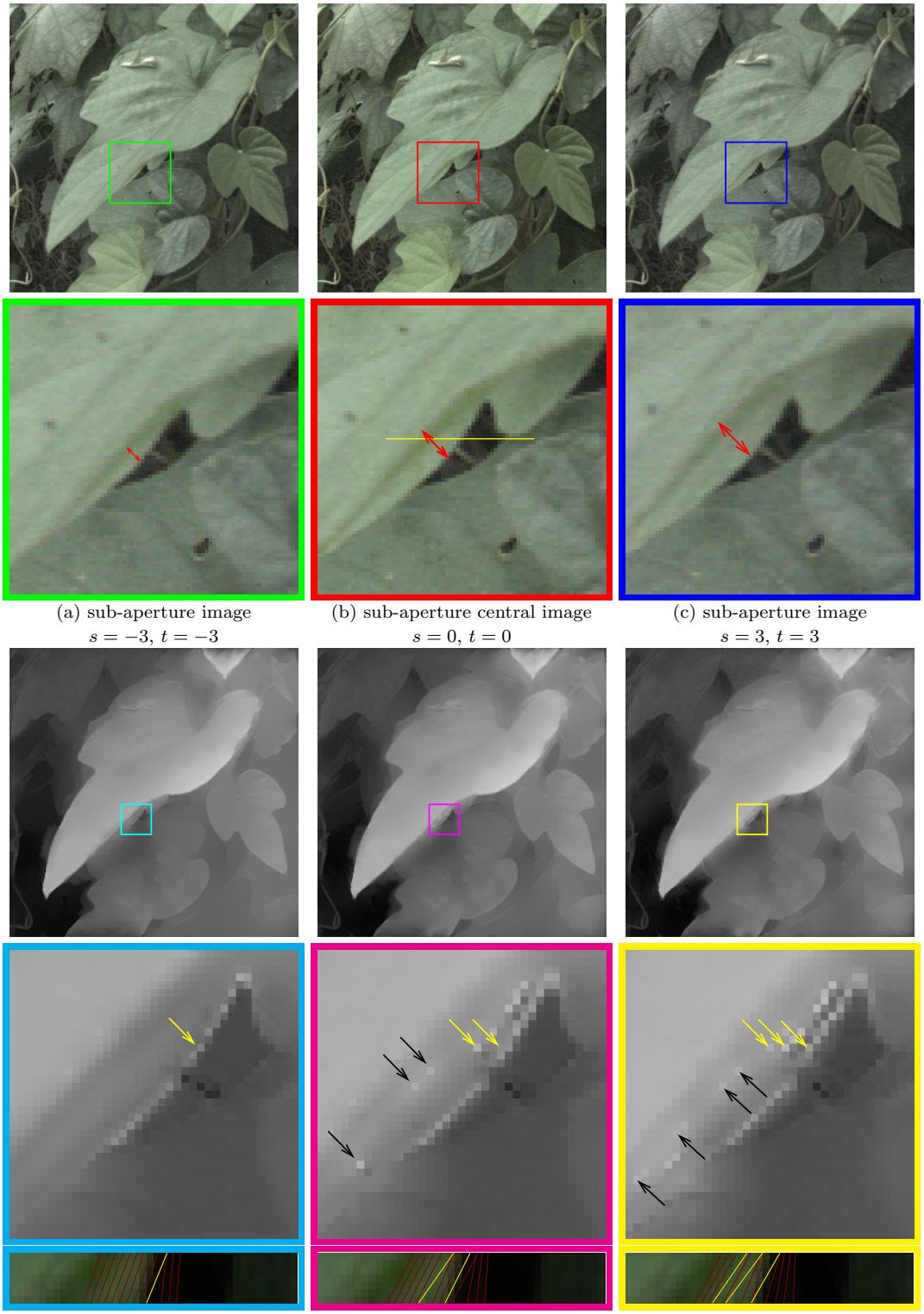
It is possible to mitigate these pixel artifacts by introducing an additional weight term in the L2 error in Eq.(4.3) that controls the influence of sub-aperture images according to their distance from the central image. If sub-aperture images which are further away from the center sub-aperture image are weighted higher / lower, there are more / less pixel artifacts in regions where the no-occlusion assumption does not hold. To achieve this reweighting, Eq.(4.3) is reformulated to:

$$m = \underset{m}{\operatorname{argmin}} \sum_s \sum_t w_{st} \left\| \widetilde{L}_m(s,t,:,:) - L(s,t,:,:) \right\|_2^2 + \lambda\, R(m)$$

For putting less weight on further sub-aperture images, we chose the weight w_{st} to be $1/|(s-s_{ctr}+t-t_{ctr})|$, where the central sub-aperture image has the coordinates (s_{ctr}, t_{ctr}). For putting more weight, we set $w_{st} = |(s - s_{ctr} + t - t_{ctr})|$. In this chapter we only reported results for $w_{st} = 1$, as differences are subtle.

Another possibility to make our algorithm more robust to deviations of the no-occlusion assumption - besides weighting the influence of the sub-aperture images - is to alleviate this assumption by introducing a weighting term that determines whether a pixel from the central sub-aperture image is occluded in the other sub-aperture images.

Another limitation is the non-arbitrary initialization of the refinement step. We think this is due to the fact that it is not possible to estimate the slopes of lines in the EPI for smooth non-boundary regions, as several slope values are feasible. This can be overcome by first estimating the slopes for boundary pixels and propagating the results to smoother regions.



(a) sub-aperture image $s = -3$, $t = -3$ (b) sub-aperture central image $s = 0$, $t = 0$ (c) sub-aperture image $s = 3$, $t = 3$

depth map of the center sub-aperture image, $s = 0$, $t = 0$

(d) less weight (e) equal weight (f) more weight

Figure 4.9: Severe violation of the no-occlusion assumption leads to pixel artifacts. **(a-c)** *Three sub-aperture images from different s and t positions. In the close-ups it can be seen that the small leave in the background is most visible in image (c), and most occluded by the big foreground leave in image (a).***(d-f)** *Depth map of image (b) with less (panel (d)), equal (panel (e)), and more weight (panel (f)) on more distant sub-aperture images. The higher the weight, the more artifacts arise. Some pixels exhibiting artifacts are highlighted with an arrow. For the single row that is highlighted in yellow in the sub-aperture central image (b), the EPI is visualized for all three cases in the bottom row. The yellow lines in the EPI images correspond to the yellow arrows in the close-ups.*

5

Conclusion and Outlook

In this chapter the contributions of the thesis are summarized. Furthermore shortcomings of the presented material and possible future directions for further research will be discussed.

5.1 Summary of Contributions

In this thesis three major contributions in the field of Computational Imaging were presented:

Benchmarking Blind Deblurring Algorithms

We built a dataset for benchmarking seven state of the art single image blind deblurring algorithms. The dataset includes 48 blurry images with four different scenes and twelve different point spread functions. We did not only record the PSFs but the full 6D-motion trajectory of the camera while the shutter was open. From that 6D trajectory it is possible to reconstruct the PSFs for a given lens model and scene setup. The recording was done in a laboratory that was equipped with 16 high-speed Vicon MX-13 cameras running at a frame rate of 500 Hz. The recorded trajectories were played back on a very precise Hexapod with six degrees of freedom, featuring repeatability at micrometer accuracy. We showed that the recorded and played back trajectories were indeed resulting in the same PSFs.

The 6D trajectories and the benchmark dataset can be downloaded from our project website [65].

We measured the deblurring results using four different image quality metrics: PSNR, MS-SIM [127], IFC [111] and VIF [110]. It is difficult to design an algorithmic IQM

that measures the image quality in the exact way as a group of humans would do. Hence we used and compared the scores of four different IQMs. The results of the four different IQMs were similar, with PSNR varying the most from the other three IQMs. We showed that the twelve blur kernels we used had a significant influence on the deblurring performance, meaning that they were of different difficulty. Furthermore we showed that the scenes 'clock' and 'roof' are harder to deblur than the scenes 'backyard' and 'curch'. We reckon probably due to the more edges in the first two images that got destroyed during the blurring picture taking process. It was possible to rank the deblurring algorithms according to their deblurring performance and the algorithm by Xu et al.[139] performed the best w.r.t all four IQMs.

Inpainting using a Multi-Layer-Perceptron

We showed that a machine learning approach can be used to tackle the inpainting problem. As the machine learning algorithm, we decided to employ a Multi-Layer-Perceptron. The key was to use an abundance of training examples, which were generated from the imagnet dataset [27], and to incorporate the information of the mask into the inpainting process. We were able to achieve better results w.r.t PSNR as state-of-the art inpainting algorithms by using a pure learning approach without incorporating any inpainting specific domain knowledge.

Surprisingly, it is also possible to train the neural network without the input of the mask to *blindly* inpaint an image. This is possible if we know the corruption process, e.g. font and font-size, because we can then generate characteristic data to train the neural network. This possibility goes beyond the capabilities of the usual approaches to inpainting which commonly require the exact location of the missing pixels. Although the results were worse for blind inpainting compared to the scenario, where the MLP was trained also on the mask, the results were still visually pleasing and comparable to state of the art inpainting algorithms.

Depth Estimation from Light Field Images

We presented a novel approach to estimate the depth map for a given sub-aperture image from a light field. Our approach consists of two steps. First, a rough initialization of the depth map is computed. In the second step, this initialization is refined by using a gradient based optimization approach. The main component of our approach, a generative model of the refinement step, can also be used for other image processing tasks on light fields.

In the refinement step the main idea is to improve the depth map by minimizing the L2 error between the estimated light field and the recorded light field. The estimated light

field is constructed by reconstructing all sub-aperture images, except the central one, by shifting pixels from the central sub-aperture image of the recorded light field to the other sub-aperture images. The amount of shifting depends on the current depth map estimation.

The underlying principle we employed for shifting the pixels of the central image is connected to the slope of the lines visible in the epipolar plane image (EPI): the slope of a line in an EPI is inversely proportional to the depth. We did not only make use of the slope of the line, but more precisely we employed the 2D plane emerging in the 4D light field.

Using described optimization approach results in depth maps, which showed fine details, but which are noisy to some extend. We reckon that this depth-estimation-noise is partly also due to the pixel noise in the sub-aperture images and also to imperfect optics.

Hence we additionally incorporated a non-local means prior that enforces that two pixels in the depth map have similar values if the corresponding pixels in the RGB image have similar color value and edge structure. This prior reduces the noise in the depth estimation while maintaining fine structure in the depth map.

We have evaluated our approach both on light field images from the Stanford light field dataset [113] and on images taken with a Lytro camera and compared our depth estimation method to state of the art algorithms. Despite the visible noise in the Lytro sub-aperture images, our recovered depth maps exhibit fine details with well-defined boundaries.

5.2 Outlook and Discussion

In this section we will focus on the shortcomings of the presented work and discuss possible future research directions.

Benchmarking Blind Deblurring Algorithms

One interesting future direction is to study whether users exhibit repeating camera shake patterns. Such a finding would motivate to learn personalized 6D-motion-trajectory-priors which ultimately could facilitate blind deconvolution.

An extension to the benchmark dataset would be to include 3D scenes and the corresponding depth maps. We decided against including 3D scenes for this version of the benchmark dataset as all tested algorithms rely on the implicit assumption that the scene does not have depth discontinuities. Having depth discontinuities would break the

assumption of a stationary blur and even the two tested algorithms that model non-stationary blur [134, 56] assume a smoothly varying non-stationary blur, which does not hold for scenes with depth discontinuities.

Besides 3D scenes, also real world outdoor scenes could be included in the benchmark dataset. Although the current dataset consists of images of outdoor scenes, they were acquired by taking images of posters showing outdoor scenes. Including outdoor scenes in the benchmark, which were actually captured in an outdoor environment, would have the advantage that the image taking process would be in the same environment as it is normally taking place, including all the parameters of such an environment (e.g. a possibly different noise distribution). Furthermore capturing real world outdoor scenes would automatically include realistic 3D scenes in the benchmark dataset. One challenge with the current setup in outdoor environments would be the change in illumination conditions. As described in Sec. 2.4.2 the ground truth images are captured by playing the trajectory step by step and taking an image at each step. A change in illumination conditions would possibly alter the scene by introducing reflections or shadows and possibly would make it necessary to change camera settings. Both may results in ground truth images which, beyond the difference due to perspective shift, might look different.

Another extension could be to include the controlled evaluation of the deblurred images by a group of human subjects. As seen in the different outcomes of the image quality metrics, it is not clear which image quality metric is the most accurate to evaluate deblurring results. A human based metric might make it easier to compare the results of the various deblurring algorithms. On the other hand a human based metric makes it harder to further include deblurring algorithms in the benchmark as the deblurred images should also be evaluated by a - ideally the same - group of people.

Inpainting

The inpainting algorithm presented was based on a learning based approach and hence does not have many parameters, despite the fixed parameters for ADADELTA and the architecture choice of the Multi-Layer-Perceptron.

Further research could be conducted in the alleviation of the requirement that the MLP has to be trained on the distortion type to have best performance, see Sec. 3.2.4. A similar behavior of the MLP was observed by [11] on the denoising task. There the performance of the MLP was specific to the chosen σ-value of the Gaussian noise. A possible solution would be to train multiple MLPs on different distortion types and to train one MLP that picks the correct MLP for the distortion type present in the corrupted image. Another approach would be to move away from the pure learning approach and incorporate domain knowledge into the Deep inpainting approach.

A further question is, if there is an optimal way to inpaint a whole image, as the MLP is only trained on patches. In the presented approach we extracted overlapping patches with stride one from the corrupted image, inpainted each patch, and combined them by averaging them, see Sec. 3.2.5. We experienced that the quality slightly deteriorates if the stride is chosen larger than one. We did preliminary experiments to also train the MLP to learn how to best combine the patches: Therefore we trained the MLP not only on patches, of e.g. size 39×39, but on regions, of e.g. size 70×70. From those regions we extracted overlapping patches with a fixed stride, fed all those patches to the MLP, combined the inpainted patches to a region by adding and averaging them and evaluated the current error of the MLP on this inpainted region and then backpropagated this error and updated the weights. We did preliminary experiments with different strides. This experimental setup did not improve the inpainting performance, but only resulted in a slower learning procedure. We reckon that the drop in learning speed was due to the patches of the region having been highly correlated.

We put our focus on the inpainting problem of *hole removal*, i.e. to fill small missing regions of an image with an appropriate color. Further research could be done to find out if a pure learning approach is also capable of the *image completion* task (e.g. removing another tourist from a holiday shot), i.e. to complete a whole region of an image with adequate information. The presented approach is not capable of doing so, as it concentrates on patches and does not look on the whole image. Having a whole image as input, with e.g. only 1MP, would let explode the number of needed parameters of the fully connected MLP. The training of such a large MLP would currently not be practically feasible. But a Convolutional Neural Network (CNN) might be possible to also learn the image completion task, as it looks at the whole image.

Depth Estimation from Light Field Images

Our work can be extended and improved in several directions. Firstly, the algorithm lends itself well to parallelization and can be implemented on a GPU.

The sub-pixel interpolation type we used to generate the estimated sub-aperture images from the center image is similar to a bilinear interpolation. Although it was not possible to visually see artifacts in the sub-aperture images arising from this interpolation type, a more accurate interpolation, like bicubic, might lead to better results. Incorporating a bicubic interpolation into presented framework, makes the optimization task more difficult and it is not clear if the optimization will be still practically feasible.

In practice, a major obstacle can be that the Lytro images are prone to noise. We conjecture that the estimation of the slopes could be improved if the original ground truth sub-aperture images were denoised. For a denoised light field, the L2 distance

between the estimated light field and the original light field should be less distorted by noise. On the other hand, due to the many sub-aperture images used for the slope estimation and due to the random nature of noise, the influence of noise might cancel out in the slope estimation, and denoising might only help for the depth estimation of some pixels. Preliminary experiments using the denoising algorithm BM3D [23] did not yield better results, as denoising tends to flatten edges. Incorporating a customized light field denoising algorithm, e.g. [24], in our optimization framework might lead to better denoising results and a better depth estimation.

We conjecture that the presented generative model or part of it might be also used for deblurring of motion-blurred light field images. This particular research area seems not to be well explored at the moment. To the best of our knowledge, there currently exists only one paper [15] about blind deblurring of motion-blurred light field images, which was put on arXiv on the 16th of August, 2014. In their work [15] the strong assumption of uniform blur (of Lambertian objects) is made. The assumption of uniform blur means - besides only allowing translational movement of the camera - that the distance of the object to the camera has to be large enough, so that depth changes can be neglected. The motion blurred light field in [15] is modeled as a linear combination of parallel convolutions. The algorithm is tested on synthetic data and on real light field images, captured with a self-build plenoptic camera. A Hasselblad H2 camera was transformed into a plenoptic camera by placing a microlens array in front of the CCD sensor.

Since the launch of the Illum light field camera by the company Lytro, there is now a consumer plenoptic camera which allows for manual settings [84]. It is now possible to take blurred light field images without building an own plenoptic camera. The predecessor model of the Illum, the First Generation Lytro, did not have any manual settings and preferred a high ISO value to a blurred light field image. Because of the ease in creating blurred light field images, it is to be expected that soon there will be many more publications about light field deblurring.

6

Appendix

Appendix to chapter 2

The full tables for all four IQMs are presented on the following pages:

	Blurred	Cho [18]	Xu [139]	Shan [109]	Fergus [36]	Krishnan [69]	Whyte [134]	Hirsch [56]
(1,1)	27.577	33.948	33.236	31.366	20.994	33.849	33.999	33.161
(1,2)	30.529	32.971	34.202	32.290	33.451	33.721	34.276	32.975
(1,3)	35.034	33.164	35.003	33.921	37.436	34.609	37.880	35.776
(1,4)	30.234	31.521	33.720	31.532	34.466	33.095	34.364	33.527
(1,5)	26.643	33.008	34.259	28.419	19.434	30.038	33.940	33.723
(1,6)	27.617	31.171	33.310	32.129	26.480	27.702	33.729	33.358
(1,7)	27.222	33.460	33.899	30.528	21.229	28.152	31.680	32.902
(1,8)	22.120	24.940	26.328	21.683	21.282	15.978	21.791	21.989
(1,9)	22.541	29.056	28.681	26.154	22.142	24.742	22.561	26.672
(1,10)	23.248	25.937	26.748	24.144	22.334	23.211	22.816	23.507
(1,11)	25.582	26.949	28.354	25.857	20.935	24.679	26.534	24.365
(1,12)	27.059	31.187	31.953	25.002	26.726	26.765	27.241	28.256
(2,1)	21.611	28.516	28.522	26.002	16.730	28.448	27.883	29.830
(2,2)	24.692	29.155	29.542	25.229	26.672	29.921	27.790	30.036
(2,3)	28.996	27.812	29.028	27.471	24.965	28.549	30.120	29.731
(2,4)	25.148	27.298	28.290	24.272	25.726	27.934	27.698	29.248
(2,5)	20.858	28.096	28.704	23.091	13.769	24.382	28.056	28.987
(2,6)	21.889	26.611	26.985	21.284	20.845	22.278	27.060	22.944
(2,7)	21.907	27.305	27.838	22.336	19.711	22.909	26.744	27.624
(2,8)	16.438	20.809	20.411	16.432	15.572	13.186	16.422	16.302
(2,9)	17.023	26.745	26.322	19.799	15.357	20.096	17.051	20.367
(2,10)	17.269	22.139	23.451	18.221	15.062	18.104	17.063	17.192
(2,11)	19.578	22.568	24.853	21.998	14.356	19.697	20.585	22.057
(2,12)	20.249	25.295	25.737	19.572	15.523	20.037	27.084	24.562
(3,1)	27.530	35.754	35.551	31.976	29.262	33.454	36.249	33.804
(3,2)	30.435	33.519	34.555	29.240	32.388	32.535	35.884	32.498
(3,3)	35.483	34.369	33.196	33.139	37.719	34.487	35.972	36.822
(3,4)	30.928	33.654	34.383	31.559	32.972	32.189	35.216	33.842
(3,5)	34.807	34.493	35.062	34.527	34.909	34.227	35.636	34.490
(3,6)	28.295	33.551	32.445	28.171	24.555	25.843	34.188	33.122
(3,7)	28.199	33.543	33.862	28.898	27.216	29.232	33.058	32.325
(3,8)	22.386	22.611	22.782	21.989	20.188	17.546	20.877	22.122
(3,9)	21.716	29.673	31.490	24.093	21.177	22.752	21.629	25.560
(3,10)	23.405	25.334	24.856	22.163	20.733	21.344	24.333	22.218
(3,11)	26.147	26.289	26.667	26.106	17.128	22.863	27.209	24.490
(3,12)	26.282	32.974	32.502	25.459	18.292	24.657	33.331	28.387
(4,1)	22.828	31.753	32.476	28.346	15.523	30.540	31.329	31.367
(4,2)	25.732	31.796	32.033	26.353	25.011	28.197	31.702	24.839
(4,3)	29.834	30.409	31.643	28.560	25.919	29.030	33.348	33.449
(4,4)	26.061	31.094	31.650	27.034	24.748	29.853	32.016	31.905
(4,5)	34.587	33.937	33.879	34.125	34.921	33.688	34.063	34.067
(4,6)	23.083	28.715	28.985	25.834	17.495	22.537	29.567	28.715
(4,7)	22.423	29.349	29.994	25.547	14.634	22.919	27.213	24.414
(4,8)	17.270	20.995	20.623	16.667	16.236	14.670	17.265	17.531
(4,9)	17.990	26.506	26.893	21.556	16.794	20.987	18.031	21.396
(4,10)	18.077	22.413	21.414	17.413	16.747	17.938	17.823	17.823
(4,11)	20.274	22.338	23.612	20.291	15.380	19.475	19.919	22.675
(4,12)	21.818	26.461	27.859	25.001	19.912	21.822	27.209	26.048

Table 6.1: PSNR values of various state-of-the-art motion deblurring algorithms for all 48 images of our benchmark dataset. Color code : blue corresponds to low PSNR (poor performance), green to intermediate and red to high PSNR values (good performance).

	Blurred	Cho [18]	Xu [139]	Shan [109]	Fergus [36]	Krishnan [69]	Whyte [134]	Hirsch [56]
(1,1)	1.852	3.206	3.290	2.705	0.707	3.197	3.458	3.140
(1,2)	1.471	2.788	3.150	1.970	2.877	2.494	2.952	2.763
(1,3)	4.592	3.544	4.221	4.074	4.861	4.631	5.029	4.339
(1,4)	1.801	2.865	3.731	2.629	3.669	3.489	3.880	3.498
(1,5)	0.636	2.718	3.524	1.762	0.342	2.288	3.065	3.166
(1,6)	0.882	2.646	3.289	2.676	0.798	1.272	3.188	2.838
(1,7)	0.935	2.696	2.974	1.786	0.441	1.291	2.140	2.686
(1,8)	0.445	1.518	1.896	0.547	0.256	0.299	0.483	0.407
(1,9)	0.991	2.469	2.690	2.080	0.331	1.305	1.171	2.178
(1,10)	0.483	1.114	1.414	0.582	0.408	0.584	0.414	0.448
(1,11)	0.707	1.212	1.549	0.959	0.346	0.929	1.178	0.389
(1,12)	1.600	2.510	2.784	1.446	1.702	1.642	1.636	2.692
(2,1)	1.714	3.144	3.433	2.343	0.784	3.384	3.038	3.585
(2,2)	1.760	3.175	3.342	1.668	2.819	3.347	2.845	3.508
(2,3)	3.846	2.926	3.992	3.013	3.264	4.003	4.320	3.685
(2,4)	1.820	2.871	3.316	1.713	2.640	3.305	3.080	3.597
(2,5)	0.806	2.679	2.997	1.565	0.267	2.152	2.746	2.993
(2,6)	0.872	2.649	2.846	0.837	1.276	1.314	2.905	1.441
(2,7)	0.956	2.542	2.619	1.113	1.032	1.513	2.296	2.691
(2,8)	0.318	1.690	1.497	0.392	0.288	0.317	0.321	0.286
(2,9)	0.808	2.760	2.915	1.578	0.272	1.487	0.987	2.043
(2,10)	0.493	1.399	1.707	0.517	0.269	0.639	0.427	0.470
(2,11)	0.847	1.501	2.048	1.507	0.308	1.206	1.121	1.656
(2,12)	1.206	2.359	2.390	1.001	0.551	1.393	2.828	2.451
(3,1)	2.087	3.526	3.954	2.808	2.108	3.167	3.696	3.518
(3,2)	1.488	2.441	2.778	1.480	2.117	2.174	2.869	2.391
(3,3)	4.006	4.016	4.313	4.059	4.905	4.541	4.470	4.613
(3,4)	1.710	3.140	3.490	2.585	2.657	3.004	3.368	3.369
(3,5)	1.240	2.800	3.087	1.900	0.351	2.281	3.036	2.559
(3,6)	1.129	2.586	2.853	1.653	1.193	0.996	2.787	2.709
(3,7)	1.097	2.346	2.562	1.480	1.209	1.880	2.371	2.316
(3,8)	0.326	1.236	1.384	0.446	0.322	0.350	0.373	0.339
(3,9)	0.794	2.194	2.561	1.888	0.314	1.067	0.881	1.927
(3,10)	0.446	1.065	1.227	0.431	0.344	0.512	0.540	0.388
(3,11)	0.801	1.051	1.256	0.989	0.412	0.976	1.003	1.135
(3,12)	1.479	2.421	2.649	1.322	0.340	1.068	2.672	2.117
(4,1)	1.853	3.723	4.220	2.831	0.490	3.412	3.990	3.822
(4,2)	1.528	3.253	3.507	1.659	2.371	2.175	3.368	1.621
(4,3)	3.844	4.385	5.169	4.003	4.601	4.730	5.312	5.149
(4,4)	1.664	3.677	4.177	2.127	3.354	3.562	4.087	3.817
(4,5)	0.948	3.009	3.461	1.822	0.490	1.863	3.004	3.124
(4,6)	1.004	3.118	3.388	2.572	0.824	1.268	3.214	3.001
(4,7)	0.928	2.676	2.947	1.932	0.605	1.315	2.121	2.471
(4,8)	0.745	1.174	1.292	0.356	0.361	0.260	0.780	0.092
(4,9)	1.264	2.376	2.869	1.775	0.254	1.191	1.271	1.821
(4,10)	0.432	1.028	1.334	0.453	0.312	0.509	0.403	0.322
(4,11)	0.605	1.063	1.423	0.659	0.273	0.780	0.508	1.222
(4,12)	1.467	2.261	2.587	2.005	1.508	1.638	2.741	2.608

Table 6.2: IFC values of various state-of-the-art motion deblurring algorithms for all 48 images of our benchmark dataset. Color code : blue corresponds to low IFC (poor performance), green to intermediate and red to high IFC values (good performance).

	Blurred	Cho [18]	Xu [139]	Shan [109]	Fergus [36]	Krishnan [69]	Whyte [134]	Hirsch [56]
(1,1)	0.856	0.978	0.975	0.949	0.589	0.974	0.973	0.972
(1,2)	0.922	0.969	0.978	0.954	0.972	0.969	0.971	0.969
(1,3)	0.981	0.977	0.986	0.981	0.991	0.982	0.992	0.984
(1,4)	0.927	0.960	0.981	0.951	0.979	0.971	0.981	0.976
(1,5)	0.818	0.965	0.979	0.895	0.416	0.947	0.969	0.974
(1,6)	0.847	0.956	0.977	0.961	0.817	0.870	0.972	0.968
(1,7)	0.843	0.970	0.974	0.936	0.600	0.902	0.945	0.965
(1,8)	0.579	0.851	0.899	0.583	0.469	0.437	0.585	0.575
(1,9)	0.644	0.951	0.953	0.860	0.526	0.865	0.655	0.867
(1,10)	0.626	0.842	0.877	0.700	0.538	0.762	0.588	0.668
(1,11)	0.770	0.850	0.897	0.815	0.494	0.805	0.823	0.701
(1,12)	0.816	0.948	0.960	0.788	0.824	0.838	0.820	0.932
(2,1)	0.804	0.972	0.977	0.929	0.655	0.973	0.958	0.981
(2,2)	0.913	0.977	0.980	0.927	0.963	0.981	0.962	0.982
(2,3)	0.976	0.973	0.984	0.967	0.962	0.980	0.987	0.979
(2,4)	0.925	0.968	0.975	0.929	0.947	0.973	0.969	0.981
(2,5)	0.760	0.961	0.971	0.862	0.345	0.915	0.958	0.969
(2,6)	0.806	0.959	0.966	0.804	0.821	0.865	0.962	0.879
(2,7)	0.811	0.952	0.958	0.851	0.788	0.891	0.940	0.953
(2,8)	0.406	0.853	0.838	0.449	0.352	0.369	0.410	0.413
(2,9)	0.530	0.947	0.938	0.777	0.391	0.836	0.534	0.817
(2,10)	0.454	0.813	0.871	0.552	0.319	0.641	0.470	0.492
(2,11)	0.680	0.839	0.909	0.838	0.383	0.796	0.743	0.828
(2,12)	0.714	0.921	0.927	0.697	0.541	0.753	0.945	0.877
(3,1)	0.882	0.987	0.988	0.964	0.927	0.976	0.985	0.980
(3,2)	0.935	0.974	0.978	0.943	0.967	0.967	0.980	0.967
(3,3)	0.982	0.987	0.983	0.985	0.993	0.985	0.989	0.991
(3,4)	0.942	0.979	0.983	0.966	0.973	0.969	0.983	0.980
(3,5)	0.958	0.975	0.976	0.950	0.935	0.942	0.977	0.960
(3,6)	0.888	0.975	0.971	0.925	0.845	0.858	0.973	0.970
(3,7)	0.886	0.969	0.971	0.925	0.893	0.941	0.963	0.963
(3,8)	0.643	0.841	0.806	0.629	0.537	0.495	0.622	0.625
(3,9)	0.692	0.946	0.964	0.862	0.585	0.838	0.705	0.881
(3,10)	0.705	0.852	0.858	0.654	0.572	0.744	0.759	0.665
(3,11)	0.835	0.865	0.876	0.847	0.415	0.799	0.862	0.837
(3,12)	0.833	0.967	0.963	0.829	0.473	0.820	0.968	0.921
(4,1)	0.823	0.984	0.988	0.954	0.537	0.976	0.984	0.980
(4,2)	0.897	0.981	0.983	0.935	0.944	0.956	0.978	0.910
(4,3)	0.966	0.987	0.992	0.981	0.975	0.984	0.992	0.992
(4,4)	0.904	0.981	0.986	0.944	0.956	0.975	0.985	0.981
(4,5)	0.954	0.968	0.977	0.940	0.930	0.922	0.960	0.970
(4,6)	0.803	0.974	0.976	0.952	0.671	0.861	0.970	0.966
(4,7)	0.772	0.963	0.966	0.904	0.494	0.861	0.925	0.940
(4,8)	0.463	0.813	0.804	0.487	0.363	0.442	0.462	0.721
(4,9)	0.570	0.942	0.954	0.821	0.441	0.845	0.579	0.814
(4,10)	0.488	0.800	0.821	0.478	0.436	0.679	0.486	0.496
(4,11)	0.600	0.783	0.841	0.671	0.402	0.720	0.592	0.799
(4,12)	0.753	0.943	0.957	0.935	0.794	0.830	0.948	0.921

Table 6.3: MS-SSIM values of various state-of-the-art motion deblurring algorithms for all 48 images of our benchmark dataset. Color code : blue corresponds to low MS-SSIM (poor performance), green to intermediate and red to high MS-SSIM values (good performance).

	Blurred	Cho [18]	Xu [139]	Shan [109]	Fergus [36]	Krishnan [69]	Whyte [134]	Hirsch [56]
(1,1)	0.235	0.542	0.562	0.480	0.118	0.542	0.517	0.502
(1,2)	0.221	0.484	0.539	0.363	0.459	0.449	0.445	0.465
(1,3)	0.593	0.594	0.711	0.679	0.742	0.776	0.739	0.723
(1,4)	0.277	0.492	0.641	0.474	0.610	0.610	0.604	0.566
(1,5)	0.097	0.460	0.589	0.332	0.048	0.410	0.469	0.497
(1,6)	0.126	0.458	0.557	0.482	0.137	0.222	0.494	0.461
(1,7)	0.131	0.450	0.501	0.332	0.069	0.235	0.333	0.434
(1,8)	0.050	0.260	0.322	0.099	0.034	0.050	0.064	0.069
(1,9)	0.102	0.430	0.453	0.358	0.039	0.234	0.128	0.325
(1,10)	0.049	0.184	0.238	0.101	0.042	0.101	0.055	0.080
(1,11)	0.101	0.210	0.277	0.186	0.045	0.180	0.181	0.072
(1,12)	0.189	0.437	0.487	0.271	0.268	0.299	0.238	0.440
(2,1)	0.203	0.444	0.484	0.348	0.115	0.476	0.407	0.494
(2,2)	0.227	0.449	0.473	0.251	0.391	0.470	0.377	0.478
(2,3)	0.469	0.414	0.544	0.445	0.455	0.557	0.555	0.521
(2,4)	0.245	0.410	0.481	0.260	0.380	0.475	0.424	0.500
(2,5)	0.100	0.386	0.442	0.234	0.034	0.315	0.379	0.423
(2,6)	0.114	0.382	0.419	0.128	0.172	0.192	0.399	0.208
(2,7)	0.124	0.363	0.385	0.172	0.149	0.226	0.317	0.383
(2,8)	0.035	0.247	0.230	0.061	0.032	0.048	0.036	0.042
(2,9)	0.092	0.398	0.420	0.247	0.033	0.223	0.113	0.289
(2,10)	0.052	0.195	0.256	0.079	0.033	0.092	0.052	0.070
(2,11)	0.100	0.207	0.300	0.231	0.038	0.179	0.151	0.225
(2,12)	0.140	0.343	0.358	0.146	0.078	0.205	0.388	0.346
(3,1)	0.249	0.620	0.659	0.508	0.363	0.553	0.558	0.583
(3,2)	0.215	0.457	0.512	0.288	0.367	0.407	0.456	0.423
(3,3)	0.538	0.672	0.741	0.712	0.750	0.763	0.691	0.732
(3,4)	0.251	0.548	0.613	0.494	0.471	0.535	0.549	0.558
(3,5)	0.144	0.497	0.545	0.352	0.061	0.413	0.466	0.438
(3,6)	0.149	0.455	0.525	0.344	0.216	0.187	0.443	0.471
(3,7)	0.143	0.421	0.464	0.279	0.212	0.355	0.360	0.404
(3,8)	0.040	0.228	0.262	0.086	0.039	0.065	0.050	0.065
(3,9)	0.093	0.373	0.443	0.342	0.048	0.201	0.120	0.303
(3,10)	0.046	0.204	0.225	0.087	0.041	0.092	0.082	0.071
(3,11)	0.109	0.171	0.231	0.182	0.056	0.181	0.172	0.209
(3,12)	0.166	0.443	0.480	0.251	0.051	0.206	0.422	0.385
(4,1)	0.209	0.520	0.576	0.406	0.071	0.476	0.528	0.506
(4,2)	0.329	0.457	0.485	0.269	0.317	0.303	0.447	0.282
(4,3)	0.454	0.583	0.651	0.538	0.574	0.630	0.654	0.635
(4,4)	0.284	0.507	0.568	0.318	0.451	0.498	0.539	0.507
(4,5)	0.104	0.407	0.472	0.257	0.068	0.262	0.394	0.417
(4,6)	0.118	0.438	0.472	0.379	0.117	0.182	0.430	0.413
(4,7)	0.105	0.374	0.415	0.274	0.084	0.185	0.284	0.338
(4,8)	0.066	0.180	0.208	0.051	0.034	0.043	0.075	0.039
(4,9)	0.126	0.343	0.411	0.260	0.032	0.174	0.139	0.246
(4,10)	0.041	0.144	0.199	0.071	0.035	0.072	0.049	0.047
(4,11)	0.064	0.143	0.198	0.092	0.029	0.108	0.063	0.156
(4,12)	0.158	0.323	0.374	0.297	0.198	0.234	0.365	0.359

Table 6.4: VIF values of various state-of-the-art motion deblurring algorithms for all 48 images of our benchmark dataset. Color code : blue corresponds to low VIF (poor performance), green to intermediate and red to high VIF values (good performance).

Bibliography

[1] E. H. Adelson and J. Y. A. Wang. "Single lens stereo with a plenoptic camera". *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 14(2):pp. 99–106 (1992).

[2] M. Bertalmio, A. L. Bertozzi, and G. Sapiro. "Navier-stokes, fluid dynamics, and image and video inpainting". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2001).

[3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. "Image inpainting". In *Proceedings of ACM SIGGRAPH* (2000).

[4] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. "Simultaneous structure and texture image inpainting". *IEEE Transactions on Image Processing*, vol. 12(8):pp. 882–889 (2003).

[5] T. E. Bishop and P. Favaro. "The light field camera: Extended depth of field, aliasing, and superresolution". *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 34(5):pp. 972–986 (2012).

[6] T. E. Bishop, S. Zanetti, and P. Favaro. "Light field superresolution". In *Proceedings of the IEEE International Conference on Computational Photography (ICCP)* (2009).

[7] D. Bitouk, N. Kumar, S. Dhillon, P. Belhumeur, and S. K. Nayar. "Face swapping: automatically replacing faces in photographs". *Proceedings of ACM SIGGRAPH* (2008).

[8] R. C. Bolles, H. H. Baker, and D. H. Marimont. "Epipolar-plane image analysis: An approach to determining structure from motion". *International Journal of Computer Vision*, vol. 1(1):pp. 7–55 (1987).

[9] A. Buades, B. Coll, and J.-M. Morel. “A non-local algorithm for image denoising”. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2005).

[10] H. C. Burger, C. J. Schuler, and S. Harmeling. “Image denoising: Can plain Neural Networks compete with BM3D?” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012).

[11] H. C. Burger, C. J. Schuler, and S. Harmeling. “Image denoising with multi-layer perceptrons, part 1: comparison with existing algorithms and with bounds”. *arXiv preprint arXiv:1211.1544* (2012).

[12] P. Carbonetto. “A MATLAB interface for L-BFGS-B”. `https://github.com/pcarbo/lbfgsb-matlab` (2014). [Online; accessed 15-April-2015].

[13] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum. “Plenoptic sampling”. In *Proceedings of ACM SIGGRAPH* (2000).

[14] T. F. Chan and J. Shen. “Nontexture inpainting by curvature-driven diffusions”. *Journal of Visual Communication and Image Representation*, vol. 12(4):pp. 436–449 (2001).

[15] P. Chandramouli, D. Perrone, and P. Favaro. “Light Field Blind Deconvolution”. *arXiv preprint arXiv:1408.3686* (2014).

[16] S.-C. Cheung, M. Venkatesh, J. Paruchuri, J. Zhao, and T. Nguyen. “Protecting and managing privacy information in video surveillance systems”. In *Protecting Privacy in Video Surveillance*, pp. 11–33. Springer (2009).

[17] D. Cho, S. Kim, and Y.-W. Tai. “Consistent Matting for Light Field Images”. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2014).

[18] S. Cho and S. Lee. “Fast Motion Deblurring”. In *Proceedings of ACM SIGGRAPH ASIA* (2009).

[19] T. S. Cho, A. Levin, F. Durand, and W. T. Freeman. “Motion blur removal with orthogonal parabolic exposures”. In *Proceedings of the IEEE International Conference on Computational Photography (ICCP)* (2010).

[20] W. S. Cleveland, E. Grosse, and W. M. Shyu. “Local regression models”. *Statistical models in S*, pp. 309–376 (1992).

[21] A. Criminisi, P. Pérez, and K. Toyama. “Region filling and object removal by exemplar-based image inpainting”. *IEEE Transactions on Image Processing*, vol. 13(9):pp. 1200–1212 (2004).

[22] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. *Mathematics of control, signals and systems*, vol. 2(4):pp. 303–314 (1989).

[23] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. “Image denoising by sparse 3-D transform-domain collaborative filtering”. *IEEE Transactions on Image Processing*, vol. 16(8):pp. 2080–2095 (2007).

[24] D. G. Dansereau, D. L. Bongiorno, O. Pizarro, and S. B. Williams. “Light field image denoising using a linear 4D frequency-hyperfan all-in-focus filter”. In *IS&T/SPIE Electronic Imaging*, pp. 86570P–86570P (2013).

[25] D. G. Dansereau, I. Mahon, O. Pizarro, and S. B. Williams. “Plenoptic flow: Closed-form visual odometry for light field cameras”. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)* (2011).

[26] D. G. Dansereau, O. Pizarro, and S. B. Williams. “Decoding, calibration and rectification for lenselet-based plenoptic cameras”. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013).

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “ImageNet: A Large-Scale Hierarchical Image Database”. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2009).

[28] M. Diebold and B. Goldlücke. “Epipolar plane image refocusing for improved depth estimation and occlusion handling”. In *Workshop on Vision, Modeling and Visualization (VMV)* (2013).

[29] I. Drori, D. Cohen-Or, and H. Yeshurun. “Fragment-based image completion”. In *Proceedings of ACM SIGGRAPH* (2003).

[30] A. A. Efros and T. K. Leung. “Texture synthesis by non-parametric sampling”. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (1999).

[31] M. Elad, J.-L. Starck, P. Querre, and D. L. Donoho. “Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA)”. *Applied and Computational Harmonic Analysis*, vol. 19(3):pp. 340–358 (2005).

[32] U. Engelke and H.-J. Zepernick. “Perceptual-based quality metrics for image and video services: A survey”. In *Proceedings of the IEEE Next Generation Internet Networks Conference (EuroNGI)* (2007).

[33] D. Erhan, A. Courville, and Y. Bengio. “Understanding representations learned in deep architectures”. *Technical Report 1355, Université de Montréal/DIRO* (2010).

[34] M.-J. Fadili, J.-L. Starck, and F. Murtagh. “Inpainting and zooming using sparse representations”. *The Computer Journal*, vol. 52(1):pp. 64–79 (2009).

[35] L. Fahrmeir, R. Künstler, I. Pigeot, and G. Tutz. *Statistik*. Springer-Verlag (2007).

[36] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. “Removing camera shake from a single photograph”. In *Proceedings of ACM SIGGRAPH* (2006).

[37] D. Ferstl, C. Reinbacher, R. Ranftl, M. Rüther, and H. Bischof. “Image guided depth upsampling using anisotropic total generalized variation”. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2013).

[38] D. J. Field. “What is the goal of sensory coding?” *Neural computation*, vol. 6(4):pp. 559–601 (1994).

[39] K.-I. Funahashi. “On the approximate realization of continuous mappings by neural networks”. *Neural networks*, vol. 2(3):pp. 183–192 (1989).

[40] T. Georgiev. “100 Years Lightfield”. `http://www.tgeorgiev.net/Lippmann/100_Years_LightField.pdf` (2008). [Online; accessed 24-April-2015].

[41] T. Georgiev, Z. Yu, A. Lumsdaine, and S. Goma. “Lytro camera technology: theory, algorithms, performance analysis”. In *IS&T/SPIE Electronic Imaging*, pp. 86671J–86671J. International Society for Optics and Photonics (2013).

[42] A. Gershun. “The light field”. *Journal of Mathematics and Physics*, vol. XVIII:pp. 51–151 (1936). [translated by Moon, Parry and Timoshenko, Gregory (1939)].

[43] B. Goldluecke and S. Wanner. “The variational structure of disparity and regularization of 4D light fields”. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013).

[44] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. “The lumigraph”. In *Proceedings of ACM SIGGRAPH* (1996).

[45] M. Guizar-Sicairos, S. T. Thurman, and J. R. Fienup. “Efficient subpixel image registration algorithms”. *Optical Letters*, vol. 33:pp. 156–158 (2008).

[46] A. Gupta. “Single image deblurring using motion density functions (Project web page)”. `http://grail.cs.washington.edu/projects/mdf_deblurring` (2010). [Online; accessed 02-May-2015].

[47] A. Gupta, N. Joshi, C. L. Zitnick, M. Cohen, and B. Curless. “Single Image Deblurring Using Motion Density Functions”. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2010).

[48] "Project Gutenberg". `http://www.gutenberg.org`. [Online; accessed 07-May-2015].

[49] P. C. Hansen, J. G. Nagy, and D. P. O'leary. *Deblurring Images: Matrices, Spectra, and Filtering*. Siam, Philadelphia,Pennsylvania (2006).

[50] S. Harmeling, M. Hirsch, and B. Schölkopf. "Space-Variant Single-Image Blind Deconvolution for Removing Camera Shake". In *Advances in Neural Information Processing Systems (NIPS)* (2010).

[51] J. Hays and A. A. Efros. "Scene completion using millions of photographs". *ACM Transactions on Graphics (TOG)*, vol. 26(3):p. 4 (2007).

[52] S. Heber and T. Pock. "Shape from Light Field Meets Robust PCA". In *Proceedings of the European Conference on Computer Vision (ECCV)* (2014).

[53] S. Heber, R. Ranftl, and T. Pock. "Variational shape from light field". In *International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)* (2013).

[54] I. Hirota. "Solid-state image pickup device and camera system" (2012). US Patent App. 14/116,426.

[55] M. Hirsch. *Blind Deconvolution in Scientific Imaging & Computational Photography*. Ph.D. thesis, Universität Tübingen (2011).

[56] M. Hirsch, C. J. Schuler, S. Harmeling, and B. Schölkopf. "Fast Removal of Non-Uniform Camera-Shake". In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2011).

[57] K. Hornik, M. Stinchcombe, and H. White. "Multilayer feedforward networks are universal approximators". *Neural networks*, vol. 2(5):pp. 359–366 (1989).

[58] A. Isaksen, L. McMillan, and S. J. Gortler. "Dynamically reparameterized light fields". In *Proceedings of ACM SIGGRAPH* (1996).

[59] J. Jaccard, M. A. Becker, and G. Wood. "Pairwise multiple comparison procedures: a review." *Psychological Bulletin*, vol. 96(3):p. 589 (1984).

[60] N. Joshi, S. B. Kang, C. L. Zitnick, and R. Szeliski. "Image deblurring using inertial measurement sensors". In *Proceedings of ACM SIGGRAPH* (2010).

[61] G. Keppel. *Design and analysis: A researcher's handbook* . Prentice-Hall, Inc (1991).

[62] C. Kim, H. Zimmer, Y. Pritch, A. Sorkine-Hornung, and M. H. Gross. "Scene reconstruction from high spatio-angular resolution light fields." In *Proceedings of ACM SIGGRAPH* (2013).

[63] D. King. *The commissar vanishes: The falsification of photographs and art in Stalin's Russia.* Henry Holt and Company, New York, New York (1997).

[64] R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf, and S. Harmeling. "Recording and Playback of Camera Shake: Benchmarking Blind Deconvolution with a Real-World Database". In *Proceedings of the European Conference on Computer Vision (ECCV)* (2012).

[65] R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf, and S. Harmeling. "Recording and Playback of Camera Shake (Project web page)". `http://webdav.is.mpg.de/pixel/benchmark4camerashake` (2012). [Online; accessed 14-May-2015].

[66] R. Köhler, B. Schölkopf, and M. Hirsch. "Precise Depth Estimation from Light Field Images". In *[Submitted to] International Conference on Computer Vision (ICCV) Workshop on Inverse Rendering.*

[67] R. Köhler, C. Schuler, B. Schölkopf, and S. Harmeling. "Mask-specific inpainting with deep neural networks". In *Proceedings of the German Conference on Pattern Recognition (GCPR)* (2014).

[68] D. Krishnan and R. Fergus. "Fast Image Deconvolution using Hyper-Laplacian Priors". In *Advances in Neural Information Processing Systems (NIPS)* (2009).

[69] D. Krishnan, T. Tay, and R. Fergus. "Blind Deconvolution using a Normalized Sparsity Measure". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2011).

[70] D. Kundur and D. Hatzinakos. "Blind Image Deconvolution". *Signal Processing Magazine*, vol. 13(3):pp. 43–64 (1996).

[71] N. Le Roux and Y. Bengio. "Deep belief networks are compact universal approximators". *Neural computation*, vol. 22(8):pp. 2192–2207 (2010).

[72] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. "Efficient backprop". In *Neural networks: Tricks of the trade*, pp. 9–50. Springer (1998).

[73] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. "Understanding and evaluating blind deconvolution algorithms". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2009).

[74] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. "Efficient Marginal Likelihood Optimization in Blind Deconvolution". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2011).

[75] M. Levoy and P. Hanrahan. "Light field rendering". In *Proceedings of ACM SIGGRAPH*, pp. 31–42. ACM (1996).

[76] N. Li, J. Ye, Y. Ji, H. Ling, and J. Yu. "Saliency detection on light field". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014).

[77] W. Lin and C.-C. J. Kuo. "Perceptual visual quality metrics: A survey". *Journal of Visual Communication and Image Representation*, vol. 22(4):pp. 297–312 (2011).

[78] G. Lippmann. "Epreuves reversibles donnant la sensation du relief". *J. Phys. Theor. Appl.*, vol. 7(1):pp. 821–825 (1908).

[79] D. Liu, X. Sun, F. Wu, S. Li, and Y.-Q. Zhang. "Image compression with edge-based inpainting". *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17(10):pp. 1273–1287 (2007).

[80] Y. Liu and V. Caselles. "Exemplar-based image inpainting using multiscale graph cuts". *IEEE Transactions on Image Processing*, vol. 22(5):pp. 1699–1711 (2013).

[81] L. B. Lucy. "An iterative technique for the rectification of observed distributions". *The Astronomical Journal*, vol. 79:pp. 745–754 (1974).

[82] A. Lumsdaine and T. Georgiev. "Full resolution lightfield rendering". *Indiana University and Adobe Systems, Tech. Rep* (2008).

[83] "Lytro desktop software". `https://www.lytro.com/desktop/` (2015). [Online; accessed 24-April-2015].

[84] "Lytro ILLUM - About Exposure Modes". `https://support.lytro.com/hc/en-us/articles/202737530-ILLUM-About-Exposure-Modes-P-I-S-M` (2014). [Online; accessed 24-July-2015].

[85] J. Mairal, M. Elad, and G. Sapiro. "Sparse representation for color image restoration". *IEEE Transactions on Image Processing*, vol. 17(1):pp. 53–69 (2008).

[86] D. Martin, C. Fowlkes, D. Tal, and J. Malik. "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics". In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2001).

[87] S. Masnou and J.-M. Morel. "Level lines based disocclusion". In *Proceedings of the IEEE International Conference on Image Processing (ICIP)* (1998).

[88] M. Mišáni. "Treatment Report". *Archive of the Conservation and Restoration Department at the Academy of Fine Arts, Bratislava, Slovakia* (2008). `http://www.vsvu.sk/en/about-us/departments/department-of-conservation-and-restoration/studio-of-easel-and-panel-paintings-conservation-and-restoration/`, [Online; accessed 21-July-2015].

[89] J. Miskin. "Train ensemble library". `www.inference.phy.cam.ac.uk/jwm1003/train_ensemble.tar.gz` (2000). [Online; accessed 07-August-2015].

[90] J. Miskin and D. J. C. MacKay. "Ensemble learning for blind image separation and deconvolution". In *Advances in Independent Component Analysis* (2000).

[91] R. Ng. *Digital light field photography*. Ph.D. thesis, Stanford University (2006). [Ren Ng founded Lytro].

[92] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan. "Light field photography with a hand-held plenoptic camera". *Computer Science Technical Report CSTR*, vol. 2(11) (2005).

[93] A. Oliva and A. Torralba. "Building the gist of a scene: The role of global image features in reconition". *Progress in brain research*, vol. 155:pp. 23–36 (2006).

[94] S. Osher and L. I. Rudin. "Feature-oriented image enhancement using shock filters". *SIAM Journal of Numerical Analysis*, vol. 27(4):pp. 919–940 (1990).

[95] J. Park, H. Kim, Y.-W. Tai, M. S. Brown, and I. Kweon. "High quality depth map upsampling for 3D-TOF cameras". In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2011).

[96] M. Pedersen and J. Y. Hardeberg. "Survey of full-reference image quality metrics". In *Research Report, Rapportserien/ GUC reports (ISSN 1890-520X), Gjøviks University* (2009).

[97] P. Pérez, M. Gangnet, and A. Blake. "Poisson image editing". In *Proceedings of ACM SIGGRAPH* (2003).

[98] P. Pérez, M. Gangnet, and A. Blake. "PatchWorks: Example-based region tiling for image editing". *Microsoft Research, Redmond, WA, Tech. Rep. MSR-TR-2004-04*, pp. 1–8 (2004).

[99] C. Perwass and L. Wietzke. "The next generation of photography". `https://github.com/pcarbo/lbfgsb-matlab` (2010). [Online; accessed 15-April-2015; Perwass and Wietzke founded Raytrix GmbH].

[100] C. Perwass and L. Wietzke. "Single lens 3D-camera with extended depth-of-field". In *IS&T/SPIE Electronic Imaging*, pp. 829108–829108 (2012).

[101] R. Raskar, A. Agrawal, and J. Tumblin. "Coded exposure photography: motion deblurring using fluttered shutter". In *Proceedings of ACM SIGGRAPH* (2006).

[102] W. H. Richardson. "Bayesian-based iterative method of image restoration". *Journal of the Optical Society of America*, vol. 62(1):pp. 55–59 (1972).

[103] S. Roth and M. J. Black. "Fields of experts: A framework for learning image priors". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2005).

[104] D. E. Rumelhart, G. E. Hintont, and R. J. Williams. "Learning representations by back-propagating errors". *Nature*, vol. 323(6088):pp. 533–536 (1986).

[105] U. Schmidt, Q. Gao, and S. Roth. "A generative perspective on mrfs in low-level vision". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2010).

[106] C. Schuler, H. Burger, S. Harmeling, and B. Schölkopf. "A machine learning approach for non-blind image deconvolution". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013).

[107] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. "Learning to Deblur". *arXiv preprint arXiv:1406.7444* (2014).

[108] I. O. Sebe, P. Ramanathan, and B. Girod. "Multi-view Geometry Estimation for Light Field Compression". In *Workshop on Vision, Modeling and Visualization (VMV)*, pp. 265–272 (2002).

[109] Q. Shan, J. Jia, and A. Agarwala. "High-quality motion deblurring from a single image". In *Proceedings of ACM SIGGRAPH* (2008).

[110] H. R. Sheikh and A. C. Bovik. "Image information and visual quality". *IEEE Transactions on Image Processing*, vol. 15(2):pp. 430–444 (2006).

[111] H. R. Sheikh, A. C. Bovik, and G. De Veciana. "An information fidelity criterion for image quality assessment using natural scene statistics". *IEEE Transactions on Image Processing*, vol. 14(12):pp. 2117–2128 (2005).

[112] H. R. Sheikh, M. F. Sabir, and A. C. Bovik. "A statistical evaluation of recent full reference image quality assessment algorithms". *IEEE Transactions on Image Processing*, vol. 15(11):pp. 3440–3451 (2006).

[113] "The (New) Stanford Light Field Archive". `http://lightfield.stanford.edu` (2008). [Online; accessed 15-April-2015].

[114] I. Sutskever and G. E. Hinton. "Deep, narrow sigmoid belief networks are universal approximators". *Neural Computation*, vol. 20(11):pp. 2629–2636 (2008).

[115] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media (2010).

[116] Y.-W. Tai, P. Tan, and M. S. Brown. "Richardson-Lucy Deblurring for Scenes under A Projective Motion Path". *Technical Report of Korea Advanced Institue of Science and Technology (KAIST)* (2009).

[117] Y.-W. Tai, P. Tan, and M. S. Brown. "Richardson-Lucy Deblurring for Scenes under A Projective Motion Path". *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2011).

[118] M. W. Tao, S. Hadap, J. Malik, and R. Ramamoorthi. "Depth from combining defocus and correspondence using light-field cameras". In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2013).

[119] A. Telea. "An image inpainting technique based on the fast marching method". *Journal of graphics tools*, vol. 9(1):pp. 23–34 (2004).

[120] C. Tomasi and R. Manduchi. "Bilateral filtering for gray and color images". In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (1998).

[121] V. Vaish, B. Wilburn, N. Joshi, and M. Levoy. "Using plane+ parallax for calibrating dense camera arrays". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2004).

[122] K. Venkataraman, D. Lelescu, J. Duparré, A. McMahon, G. Molina, P. Chatterjee, R. Mullis, and S. Nayar. "Picam: An ultra-thin high performance monolithic camera array". In *Proceedings of ACM SIGGRAPH ASIA* (2013).

[123] Y. Wang, J. Yang, W. Yin, and Y. Zhang. "A new alternating minimization algorithm for total variation image reconstruction". *SIAM Journal on Imaging Sciences*, vol. 1(3):pp. 248–272 (2008).

[124] Y. Wang and W. Yin. "Compressed sensing via iterative support detection". *Technical Report TR09-30, Rice University, Department of Computational and Applied Mathematics* (2009).

[125] Z. Wang and A. C. Bovik. "Mean squared error: love it or leave it? A new look at signal fidelity measures". *Signal Processing Magazine, IEEE*, vol. 26(1):pp. 98–117 (2009).

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

[127] Z. Wang, E. P. Simoncelli, and A. C. Bovik. "Multiscale structural similarity for image quality assessment". In *Proceedings of the Asimolar Conference on Signals, Systems and Computers.* (2003).

[128] S. Wanner and B. Goldlücke. "Spatial and angular variational super-resolution of 4D light fields". In *Proceedings of the European Conference on Computer Vision (ECCV)* (2012).

[129] S. Wanner and B. Goldluecke. "Globally consistent depth labeling of 4D light fields". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012).

[130] S. Wanner and B. Goldluecke. "Variational light field analysis for disparity estimation and super-resolution". *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 36(3):pp. 606–619 (2014).

[131] S. Wanner, C. Straehle, and B. Goldluecke. "Globally consistent multi-label assignment on the ray space of 4D light fields". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013).

[132] Y. Wexler, E. Shechtman, and M. Irani. "Space-time completion of video". *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 29(3):pp. 463–476 (2007).

[133] O. Whyte, J. Sivic, and A. Zisserman. "Deblurring Shaken and Partially Saturated Images". In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2011).

[134] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. "Non-uniform Deblurring for Shaken Images". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2010).

[135] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. "Non-uniform deblurring for shaken images". *International Journal of Computer Vision*, vol. 98(2):pp. 168–186 (2012).

[136] Wikipedia. "Censhorship of images in the Soviet Union". `https://en.wikipedia.org/wiki/Censorship_of_images_in_the_Soviet_Union`. [Online; accessed 03-August-2015].

[137] A. Wong and J. Orchard. "A nonlocal-means approach to exemplar-based inpainting". In *Proceedings of the IEEE International Conference on Image Processing (ICIP)* (2008).

[138] J. Xie, L. Xu, and E. Chen. "Image Denoising and Inpainting with Deep Neural Networks". In *Advances in Neural Information Processing Systems (NIPS)* (2012).

[139] L. Xu and J. Jia. "Two-Phase Kernel Estimation for Robust Motion Deblurring". In *Proceedings of the European Conference on Computer Vision (ECCV)* (2010).

[140] Z. Xu and J. Sun. "Image inpainting by patch propagation using patch sparsity". *IEEE Transactions on Image Processing*, vol. 19(5):pp. 1153–1165 (2010).

[141] J. Yang, Y. Zhang, and W. Yin. "An efficient TVL1 algorithm for deblurring multichannel images corrupted by impulsive noise". *SIAM Journal on Scientific Computing*, vol. 31(4):pp. 2842–2865 (2009).

[142] L. Yuan, J. Sun, L. Quan, and H.-Y. Shum. "Image Deblurring with Blurred/Noisy Image Pairs". In *Proceedings of ACM SIGGRAPH* (2008).

[143] M. D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". *arXiv preprint arXiv:1212.5701* (2012).

[144] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization". *Transactions on Mathematical Software (TOMS)*, vol. 23(4):pp. 550–560 (1997).

Index