Variational and Deep Learning Approaches for Intrinsic Light Field Decomposition

Doctoral thesis for obtaining the academic degree Doctor of Natural Sciences (Dr.rer.nat.)

submitted by Anna Alperovich

at the Universität Konstanz

Faculty of Science
Department of Computer and Information Science

Konstanz, 2020
Date of the oral examination: 19. February 2020

1. Reviewer: Prof. Dr. Bastian Goldlücke
2. Reviewer: Prof. Dr. Dietmar Saupe
3. Reviewer: Prof. Dr. Christine Guillemot
Abstract

Intrinsic image decomposition aims to separate an illumination invariant reflectance image from an input color image, which is still one of the fundamental problems in computer vision. This decomposition is widely used in photo and material editing, image segmentation and shape estimation tasks. According to the dichromatic reflection model, the light reflected from a scene point has two independent components: light reflected from the surface body and light at the interface. Body reflection is known as the diffuse component and it is independent of viewing direction, while interface reflection is known as the specular component and it is view-dependent. Most intrinsic image algorithms are designed for Lambertian scenes, with only diffuse reflection. However, their performance decreases if a scene contains specularity. In the real world, there are few scenes with only Lambertian objects. Instead, they have specular surfaces, which makes the decomposition problem harder due to the complicated nature of specular reflection.

This thesis focuses on intrinsic light field decomposition, where we formulate and solve the problem with respect to three variables: albedo, shading, and specularity. Thus, we can deal with non-Lambertian scenes. We use a 4D light field, which is a collection of images sampled on a regular grid, instead of a single image. Rich information inherited from the light field allows us to distinguish between diffuse and specular reflection, and also allows us to robustly recover the intrinsic components. We tackle the problem with variational and deep learning approaches, compare their performance, and discuss the strengths and weaknesses of both techniques.

In the variational method, we introduce priors for the intrinsic components and we solve an energy minimization problem with convex optimization. Because geometrical information plays an important role in the appearance and behavior of intrinsic components, we develop a disparity estimation method, where we not only optimize the disparity labels but also enforce piecewise smoothness of a normal map.

Our deep learning approach is based on the assumption that if mathematical models allow us to compute a disparity and intrinsic components from a light field, then these models can be approximated with a deep convolutional neural network. Moreover, because disparity estimation and intrinsic light fields are closely related, a single network can be sufficient to perform all tasks together and they can benefit from each other. Thus, we establish a multi-task learning strategy for light fields, which is not only limited to the particular collection of tasks but (in theory) can also be used for various computer vision applications. We demonstrate the advantage of our approach on four state-of-the-art computer vision problems: disparity estimation, reflection separation, intrinsic images, and super-resolution.

Extensive evaluations based on multiple, publicly-available, synthetic and real-world datasets prove our methodology and show the advantage of using light fields over other data structures. Our proposed algorithms outperform state-of-the-art methods for intrinsic images and disparity estimation, and achieve a competing quality for super-resolution and reflection separation.
Zusammenfassung


Für die Zerlegung mithilfe variationeller Methoden führen wir Prioren für die intrinsischen Komponenten ein und lösen das Energieoptimierungsproblem durch konvexe Optimierung. Da geometrische Informationen eine wichtige Rolle für das Erscheinungsbild und das Verhalten der intrinsischen Komponenten spielen, entwickeln wir eine Methode zur Schätzung der Disparität, wobei wir nicht nur die Disparitäts-Karte optimieren, sondern auch die abschnittsweise Glattheit der Normalen-Karte erzwingen.

Unser Deep Learning Ansatz basiert auf der Annahme, dass mathematische Modelle, die Disparitäten und intrinsische Komponenten eines Lichtfeldes berechnen, durch ein Deep Convolutional Neural Network approximiert werden können. Da die Berechnung der Disparitäten eng mit den intrinsischen Komponenten verknüpft ist, kann ein einzelnes Netzwerk davon profitieren die Berechnungen gemeinsam auszuführen. Dementsprechend nutzen wir eine Lernstrategie für Lichtfelder, die mehrere Aufgaben gleichzeitig optimiert, und somit nicht auf bestimmte Aufgaben beschränkt ist, sondern theoretisch für verschiedene Anwendungen aus dem Bereich der Computer Vision verwendet werden kann.

Wir demonstrieren die Vorteile unserer Herangehensweise an vier aktuellen Problemen der Computer Vision: Disparitätsschätzung, Reflexionsseparierung, intrinsische Zerlegung und die Erhöhung der Auflösung eines Bildes.

Unsere umfangreichen Auswertungen auf mehreren öffentlich zugänglichen synthetischen und realen Datensätzen unterstreichen die Qualität unserer Methodik und zeigen den Vorteil von Lichtfeldern gegenüber anderen Datenstrukturen. Die vorgeschlagenen Algorithmen übertreffen modernste Methoden zur Berechnung der intrinsischen Zerlegung und Disparitäten und erzielen konkurrenzfähige Resultate für Superauflösung und Reflexionsseparierung.
First of all, I would like to thank my first advisor, Prof. Dr. Bastian Goldlücke, for the great support and excellent guidance during the whole time of my PhD. His advice and feedback helped to improve the research papers, such that they were accepted to large computer vision conferences, and he inspired me to challenge myself with the new projects.

Also, I would like to thank my colleagues Ole Johansen and Antonin Sulc for their scientific contributions, everyday support, warm atmosphere and fruitful discussions. It was a pleasure to work with you.

Special thanks to our Master’s students Minchen Zhu and Michael Strecke for their help and contributions, hard-working attitude and creativity. Thank you for being good and supportive friends.

Next, I would like to thank Prof. Dr. Dietmar Saupe for the great social events that he organized for the department members with lots of activities in the mountains.

Finally, I would like to thank Vlad Hosu for sharing amazing times in the mountains, interesting discussions and a nice teaser picture for one of our papers, and Marcel Lederle for a good company in the last year of my stay in Konstanz and scientific translation of the abstract to this thesis in German.

This work was supported by the ERC Starting Grant ”Light Field Imaging and Analysis” (LIA 336978, FP7-2014) and the SFB Transregio 161 ”Quantitative Methods for Visual Computing”.

Acknowledgements
Contents

1 Introduction ................................................................. 1
   1.1 Light Field Theory ................................................... 2
      1.1.1 Light Field Representation .................................. 2
      1.1.2 Acquisition Methods .......................................... 4
   1.2 Depth Estimation .................................................... 5
      1.2.1 Contributions .................................................. 6
   1.3 Intrinsic Light Fields .............................................. 6
      1.3.1 Contributions .................................................. 7
   1.4 Deep CNN for Light Fields ........................................ 7
      1.4.1 Contributions .................................................. 9
   1.5 List of Projects .................................................... 9

2 Disparity Estimation ................................................... 11
   2.1 Related Work ....................................................... 11
      2.1.1 Two-plane representation and sub-aperture views .......... 11
      2.1.2 Epipolar plane images ........................................ 11
      2.1.3 Focal stack ..................................................... 12
      2.1.4 Regularization and optimization ............................ 12
   2.2 Focal Stack Symmetry .............................................. 12
      2.2.1 Focal stack symmetry [82] ................................... 13
      2.2.2 Occlusion-aware focal stack symmetry ...................... 14
      2.2.3 Proposition .................................................... 14
   2.3 Regularization of Normals ......................................... 15
      2.3.1 Relationship between depth and normals .................... 15
      2.3.2 The final prior on normal maps ............................... 16
      2.3.3 Optimization ................................................... 16
      2.3.4 Optimization for depth ....................................... 16
      2.3.5 Optimization for the normal map ......................... 17
   2.4 Results and Comparisons .......................................... 17
      2.4.1 Benchmark performance ....................................... 17
      2.4.2 Comparison to focal stack symmetry ...................... 19
      2.4.3 Normal map accuracy ......................................... 19
      2.4.4 Real-world results .......................................... 19
   2.5 Conclusions .......................................................... 23

3 Variational Models for Intrinsic Light Field Decomposition .... 24
   3.1 Related Work ........................................................ 24
   3.2 Variational Formulation ............................................ 25
      3.2.1 Model 1: albedo, shading and specularity .................. 26
      3.2.2 Model 2: albedo, direct shading, indirect shading and specularity 27
   3.3 Specularity .......................................................... 28
      3.3.1 Modeling assumptions ......................................... 28
      3.3.2 Computing the specular mask ................................ 30
      3.3.3 Final prior on specularity ................................... 32
### Contents

3.4 Albedo ................................................. 32
3.5 Shading .................................................. 34
  3.5.1 Simple shading ..................................... 35
  3.5.2 Cast shadow detection ............................... 35
  3.5.3 Direct shading ...................................... 38
  3.5.4 Indirect shading ................................... 38
3.6 Ray Space Regularization and Optimization .................. 39
3.7 Results and Comparisons ................................ 41
3.8 Conclusions ............................................ 49

4 Deep Autoencoder for Light Fields .......................... 50
  4.1 Related Work .......................................... 50
    4.1.1 Neural networks for light field analysis ........... 50
    4.1.2 Encoding light fields ............................. 51
    4.1.3 Disparity estimation ............................... 51
    4.1.4 Reflection separation .............................. 51
    4.1.5 Intrinsic images .................................. 52
    4.1.6 CNN-based single image super-resolution ........... 52
    4.1.7 Light field super-resolution ........................ 52
  4.2 General Architecture .................................. 55
    4.2.1 Encoder pathway ................................ 55
    4.2.2 Decoder pathways and output ...................... 55
  4.3 Loss Function ......................................... 56
  4.4 Building Blocks ........................................ 58
    4.4.1 Convolution and convolution transposed ............ 58
    4.4.2 Activation functions ............................... 60
    4.4.3 Batch normalization ................................ 61
    4.4.4 Skip connections .................................. 61
    4.4.5 Gradient Descent and Adaptive Moment Estimation (ADAM) ........................................ 62
  4.5 Data Preprocessing ..................................... 64
    4.5.1 4D light field benchmark [60] ..................... 64
    4.5.2 New light fields rendered with Blender .......... 65
    4.5.3 Real-world light fields ............................ 66
    4.5.4 Augmentation ...................................... 66
  4.6 Reflection Separation and Disparity Estimation CNN .......... 67
    4.6.1 3D reflection separation and 2D disparity estimation architecture .............................. 68
    4.6.2 Network implementation and training strategy ......... 68
    4.6.3 Qualitative and numerical evaluations ............ 69
    4.6.4 Conclusions ...................................... 71
  4.7 Deep Intrinsic Light Field Decomposition CNN ............ 78
    4.7.1 Underlying model .................................. 78
    4.7.2 Intrinsic light field CNN architecture ............ 79
    4.7.3 Experiments ...................................... 81
    4.7.4 Conclusions ...................................... 83
  4.8 Generative Adversarial Network for Light Field Super Resolution .. 89
    4.8.1 Introduction ...................................... 89
    4.8.2 Outline of model and losses ....................... 90
    4.8.3 Detailed network architecture ...................... 92
    4.8.4 Training strategy .................................. 94
    4.8.5 Experiments ...................................... 95
    4.8.6 Conclusions ...................................... 97
  4.9 Summary ................................................ 102
5 Conclusions and Future Work 103
Vision is a complicated task that is natural and easy for most people but very challenging for a machine. When we look at an object, we can roughly tell how far it is from us and what kind of texture, color and shape it has. In many cases, we also can recognize lighting conditions, such as daylight or artificial illumination of a certain color. In fact, our brain solves the inverse problem where, given a view or collection of views of a scene, it recovers objects properties based on the observation and prior knowledge. For decades, computer vision research has aimed to reproduce and enhance human vision. Nowadays, there are thousands of algorithms that solve various state-of-the-art problems. In this work, we concentrate on inverse problems, where we want to recover certain components of a scene given its photographic images. We investigate two approaches: the first approach explores the modeling of the problem based on physical principles of image formation, and the second approach employs deep learning.

Our work mainly differs from many of its kind in two aspects. First, instead of a single image of a scene, we use a collection of images taken from a slightly different perspective (light fields) as an input to our algorithms. In prior work, it was shown that rich information inherited from the light field can be used to robustly estimate geometry and recover various physical properties of a scene. Second, we concentrate on the non-Lambertian phenomena that can be separated from the diffuse reflection thanks to the light field data.

This work consists of two parts: Chapters 2 and 3 are devoted to the variational methods for intrinsic light fields, where the input light field is decomposed into three rendering components: albedo, shading, and specularity. Albedo represents the true color of the objects in a scene, shading captures intensity changes due to geometry and illumination, and specularity contains highlights and reflection at an interface. To model that components the precise geometry of a scene is required. Thus, we start our work with the disparity estimation method, Chapter 2, which not only recovers disparity labels but also outputs piecewise smooth normal map. Consequently, the 3D representation of a scene can be computed and used for modeling the intrinsic components. The main drawback of this method is that to compute disparity, we assume that the scene is pure Lambertian, which is almost never the case in reality. Thus, the proposed algorithm fails to recover accurate disparity labels in specular regions. Regularization of normal map partially solves this problem for small highlighted areas that are surrounded by almost diffuse regions, but the problem persists for large specular surfaces.

The second part, Chapter 4, describes a deep learning approach where we investigate the possibility to learn a physical model by only feeding the neural network with training examples. We develop a 3D convolutional neural network (CNN) to deal with angular dimension in the light field and show the benefit of this network for the state-of-the-art computer vision problems. Because we train the network on the non-Lambertian data, we enforce it to cope with specular areas and as the result to overcome the limitation of modeling approach. In numerous experiments, we compare the performance of both approaches and illustrate the advantage of using deep network for disparity estimation and intrinsic images.

We show that having the light field as an input, both variational and deep learning methods outperform single image and image+depth methods. In particular, we illustrate the advantage of using light fields for
specular scenes, where most of the algorithms fail due to the Lambertian assumption. This opens a wide range of applications and directions for future research towards the understanding of non-Lambertian scenes with light fields.

1.1 Light Field Theory

With the development of hardware technology, computer vision and image processing have attracted considerable attention in recent decades. Although single image approaches achieve good performance in many state-of-the-art computer vision tasks, additional information about the scene is required to improve the quality of these methods and avoid ambiguities that may occur due to the lack of information in the single image. A while ago, researchers started to focus on data structures, such as image sequences, video and various sensor data. The light field is another rich source of information, where instead of a single image, we have an array of images of the same scene, taken from a slightly different viewpoint. It has been shown [41] that many computer vision problems can benefit from such additional information. This work illustrates how rich information inherited from light field can be used in reflection separation, intrinsic images, super-resolution and disparity estimation tasks. We start with an introduction to light fields and we then mention some of its main properties.

1.1.1 Light Field Representation

The concept of light field originated from the plenoptic function, which was introduced by Adelson and Bergen [2] to describe models of low-level vision. The function

\[ P = P(\theta, \phi, \lambda, t, V_x, V_y, V_z) \]  

measures the intensity of the light rays passing through the eye at the position \((V_x, V_y, V_z)\) in 3D space at the angle \((\theta, \phi)\), for the wavelength \(\lambda\), at time \(t\). The angles \((\theta, \phi)\) are computed with respect to the axis that is parallel to the \(V_z\) axis. In computer graphics and vision, it is common to parametrize the rays entering the point or eye in terms of \((x, y)\) coordinates, where \(x\) and \(y\) are coordinates of the image.
McMillan and Bishop [87] reduced plenoptic function (1.1) representation to 5D, describing the flow of light at every 3D spatial position $(V_x, V_y, V_z)$ for every 2D direction $(\theta, \phi)$.

Later, Gortler et al. [44] reduced the 5D representation to 4D under the free space assumption. If the air is transparent, then the radiance along the ray through empty space remains constant unless it is blocked. If we only consider the light leaving the convex hull of a bounded object, then we can evaluate the plenoptic function on some surface that surrounds the object.

The 4D light field or Lumigraph is the most commonly used representation in research and industry. The key idea of light field is to represent a scene not as a traditional 2D image, which contains information about accumulated intensity at each image point, but as a collection of images of the same scene from slightly different view points, see Fig. 1.1.

A light field is defined on 4D ray space $\mathcal{R} = \Pi \times \Omega$, where a ray is identified by four coordinates $r = (s, t, x, y)$, which describe the intersections with to parallel planes. Here, $(s, t)$ are view point coordinates on the focal plane $\Pi$, and $(x, y)$ are coordinates on the image plane $\Omega$. Figure 1.2 visualizes the light field geometry. Epipolar plane images (EPIs) can be obtained by restricting 4D ray space to 2D slices, see Fig. 1.1, 1.2. For instance, we fix the two coordinates $(t^*, y^*)$, then the map

$$L_{t^*, y^*} : (s, x) \rightarrow L(s, t^*, x, y^*)$$

represents an EPI. The other restrictions $L_{s^*, x^*}$, $L_{s^*, t^*}$ are defined similarly to equation (1.3). Note, that $L_{s^*, t^*}$ is a pinhole or sub-aperture view of the light field $L$. Let $P$ be a scene point, $P \in \mathbb{R}^3$, see Fig. 1.2. $P$ projects to a point in $\Omega$ depending on the chosen camera center in $\Pi$. According to Bolles et al.[18] the shift $\Delta x$ in the image plane

$$\Delta x = -\frac{f}{Z} \Delta s$$

linearly depends on the shift $\Delta s$ in the camera plane, where $Z$ is the distance from $P$ to the camera plane.
Π, which is called depth of $P$, $f$ is the distance between two planes or focal length. The rate of change $d(P) = \frac{1}{2}$ is called the disparity of $P$. For more information and a thorough introduction to light field geometry, we refer to [18, 44, 80].

### 1.1.2 Acquisition Methods

![Light field acquisition methods: a. Single camera on the gantry [126]. b. The setup of a camera array [126]. c. A detailed look of the image captured by a microlens array camera. On the right-hand side is the whole image, which consists of sub-aperture views that are illustrated in the zoomed-in part on the left-hand side.](image)

Figure 1.3: Light field acquisition methods: a. Single camera on the gantry [126]. b. The setup of a camera array [126]. c. A detailed look of the image captured by a microlens array camera. On the right-hand side is the whole image, which consists of sub-aperture views that are illustrated in the zoomed-in part on the left-hand side.

There are several ways to capture light fields, some of them already output sub-aperture views and others need additional post-processing to obtain a light field representation as described in the previous section 1.1.1. A light field of static scenes can be acquired by moving a single camera over the scene; for example, mounting the camera on a gantry or the robot arm and taking images of the scene step by step, see Fig. 1.3 a). The quality of the images depends on the specification of the camera. The main drawback of moving a single camera is that it is time-consuming because each exposure takes only one image of the scene.

The second way to capture light field is to use the camera array; i.e. arranging multiple cameras as an array or a matrix with their centers on the same plane, see Fig. 1.3 b). The advantage of the camera array is that, with a single exposure of all the cameras at the same time, one can obtain the light field and the image quality also depends on the specification of each camera. However, the angular resolution of the captured light field is limited by the number of cameras used in this setup. Furthermore, it is close to impossible to move the camera array around thus the usage is strongly restricted.

As light field imaging becomes more popular, the consumer level plenoptic cameras equipped with microlens array, such as the Lytro plenoptic cameras, have been introduced, see Fig. 1.3 c) for an example of the raw output of a light field camera. The microlens array, as the name suggests, is set up with multiple lenses of small size placed in front of the sensor in the camera. When capturing an image with a microlens array camera, the light rays firstly go through the main lens and project onto a specific area of the sensor through the microlens. After the post-capture processing—which involves decoding, calibration and rectification procedure [31], the acquired image consists of sub-aperture views of the scene; i.e. a light field, see section 1.1.1. Although consumer-level microlens array cameras have the advantage that they are easy to carry, the images usually suffer from low spatial resolution, the noise caused by the natural illumination and sometimes unwanted effects arise due to optical diffraction at the small features.
Chapter 1 Introduction

Figure 1.4: We present a novel idea to compute disparity cost volumes that is based on the concept of occlusion-aware focal stack symmetry. Using the proposed framework, we can optimize jointly for depth and normals to reconstruct challenging real-world scenes captured with a Lytro Illum plenoptic camera. The left-hand image shows the center view of the light field, the top right-hand image illustrates the disparity labels, and the bottom right-hand represents the normal map.

With the help of computer graphics, it is possible to generate high-resolution light fields through rendering. Compared to the optical-equipment based methods mentioned above, rendering provides better ground truth light fields because the information (e.g. resolution, disparity, illumination condition, objects, and background) can be customized and stored for later use.

1.2 Depth Estimation

The specific structure of the light field allows a wide range of application; for example, it can be used for a large class of inverse problems, including depth estimation, light field denoising and inpainting [41], virtual refocusing, automatic glare reduction, as well as object insertion and removal [30, 70, 80]. Recently, the inherent structure of the light field was leveraged for shape and BRDF estimation [118, 130]. A common use of the light field is to estimate the depth in the captured scene to furthermore reconstruct its 3D structure, see equation (1.4).

Current algorithms for disparity estimation, e.g. [23, 63, 71, 82, 128] and many more cited in the references, work exceedingly well when estimating depth from light field images. However, these methods are usually not designed with normal estimation in mind. Thus, the depth estimates from algorithms based on cost volumes, even when optimized with sublabel accuracy [90], are often piecewise flat and thus fail to predict accurate normal maps. Frequently, their accuracy is also naturally limited around occlusion boundaries [67, 82, 133]. The aim of the disparity estimation algorithm described in the thesis is to contribute towards a remedy for these drawbacks, see Fig. 1.4 for an example output of the proposed method.
1.2.1 Contributions

First, we introduce a novel way to handle occlusions when constructing cost volumes based on the idea of focal stack symmetry [82]. This novel data term achieves substantially more accurate results than the previous method [82] when a global optimum is computed with sublabel relaxation [90]. Second, we propose post-processing using joint regularization of depth and normals to achieve a smooth normal map that is consistent with the depth estimate. For this, we employ ideas from Graber et al. [45] to linearly couple depth and normals, and employ the relaxation in Zeisl et al. [144] to deal with the non-convexity of the unit length constraint on the normal map. The resulting sub-problems on depth and normal regularization can be efficiently solved with the primal-dual algorithm from [22]. By the submission time of our research paper [109], our results had substantially outperformed all previous work that had been evaluated on the recent benchmark for disparity estimation on light fields [60] with respect to accuracy of disparity and normal maps and several other metrics.

1.3 Intrinsic Light Fields

Intrinsic image decomposition separates an illumination invariant reflectance image from an input color image, see example in figure 1.5. Such a decomposition has numerous applications in color enhancement, image segmentation, pattern recognition, and object tracking [14, 77, 106]. The separation of the shading component is used in BRDF estimation and shadow removal methods [37, 38, 130]. However, while intrinsic images have many applications, recovering them remains a substantial challenge for researchers. Estimation of intrinsic components is an ill-conditioned problem: a single image can be decomposed into infinitely many different combinations of reflectance and illumination. Thus, additional constraints or priors are needed to select an appropriate solution.

Priors on reflectance (albedo) and shading are usually based on physical principles of light and object interaction, scene geometry, and material properties, as well as on expert knowledge of how intrinsic images should look like. Finally, decomposition into reflectance and illumination components is suitable only for diffuse (Lambertian) objects. According to the dichromatic model introduced by Shafer [101], if glossy (non-Lambertian) objects are present in a scene, a specular term should be taken into account, see example in figure 1.6. Many classical approaches fail when the target scene has non-Lambertian objects; as specularity depends on viewpoint, it is hardly possible to estimate it from a single image.

To improve the quality of intrinsic image decomposition, researchers use additional information, such as a video sequence instead of a single image, RGB-D imaging sensors, or manual labeling. However,
Figure 1.6: Non-Lambertian intrinsic image decomposition. The light reflected from a scene point has two independent components, light reflected from the surface body and at interface. Body reflection is known as diffuse or Lambertian component and it is independent of viewing direction, interface reflection is specular component, that is viewpoint-dependent.

this information may be incomplete, suffer from sensor noise, calibration errors, and be dependent on a human factor. Computing this information may be time consuming, require complex experiments, and special equipment. Thus, it is hardly possible to use it, for example, in industrial applications.

Such a dense collection of viewpoints is described by the light field of a scene. Dedicated light field cameras efficiently acquire dozens of views by multiplexing the rays onto a single sensor or using multiple standard industrial cameras in an array. Compared to stereo imaging, light fields are more redundant and provide richer information about the scene; furthermore, the baseline is typically much smaller. Thus, instead of point correspondences, pixels are projected onto lines in EPIs that can be more reliably detected and analyzed.

Depending on the input data, the state-of-the-art approaches can be divided into those dealing with a single image [39, 46, 122], multiple images [79, 135], and image + depth methods [25, 64]. Most of the algorithms use a Lambertian assumption, and they decompose input image into albedo and shading components. However, this is not sufficient for most of the real world scenes where most of the objects have both diffuse and specular reflection.

1.3.1 Contributions

We propose an intrinsic light field models for non-Lambertian scenes [4, 6], where decomposition is performed with respect to albedo, shading, and specularity. We exploit the structure of the light field: while albedo and shading are view-independent and therefore constant along projections of the same scene point in EPIs, specularity is view-dependent and shows a different behavior. We combine geometrical and color information to define a novel shading prior. We design a shadow detection model with light field data, which to our knowledge is the first time this problem has been addressed for light fields. We apply the estimated shadow score to model cast shadows and inter-reflections explicitly, which results in more consistent shading compared to previous approaches, with better identification of soft and hard shadows. Furthermore, we use pre-estimated specularity positions to make both albedo and shading priors less affected by highlights. For this, we compute a specular free representation of the input light field with [114], and use this information to model albedo and shading. A thorough evaluation of our algorithms on synthetic datasets with ground truth, as well as the real world examples demonstrates that we outperform previous approaches based on RGB+D images or light fields.

1.4 Deep CNN for Light Fields

Light fields have a complex, heavily redundant structure, see Fig. 1.1. For scenes with purely diffuse reflection, EPIs exhibit patterns of oriented lines of constant color. Each of these lines corresponds to the
Figure 1.7: Encoder-decoder architecture for light fields. It takes horizontal and vertical epipolar volumes as an input and passes them through the autoencoder chain of convolutions and upsampling layers. The architecture can have various decoding pathways. In this work we illustrate the application of the proposed network to three state-of-the-art computer vision tasks: joined reflection separation and disparity estimation, intrinsic light field decomposition, and super-resolution.

projection of a single 3D point in space, and its slope, or disparity, is inversely proportional to the point’s distance to the observer. Discontinuities in the pattern are caused by occlusions, which cause transitions between multiple orientations at the occlusion edge [128].

The situation also becomes less straightforward when reflection or glossy, non-Lambertian surfaces come into play because the EPIs then show superimposed patterns [67]. The orientation of the patterns corresponding to specular reflection does not correspond to disparity but does correspond to the specular flow direction, which depends on the intrinsic surface geometry. To distinguish between those two cases, one must know if a point exhibits diffuse or specular reflection. With a known geometry, the specular flow can be directly estimated and reflection components can be separated [110]. If both shape and reflectance are unknown, then it is hardly possible to tell which phenomena gave rise to a particular EPI.

Nevertheless, EPIs from natural light fields exhibit an overall regular structure, and it seems likely that they form a comparatively low-dimensional manifold within all of epipolar plane image space. Furthermore, encoding an EPI well with only a few parameters is related to the difficult interrelated tasks, such as disparity estimation or separation of albedo, shading and specular components. Intuition suggests that if you learn how to do compression well, then you will be able to better succeed at the other tasks. The idea of this part of the work is therefore to learn a low-dimensional representation of EPIs from arbitrary example light fields, but in such a way that the latent variables can be used jointly to accurately solve various supervised tasks in light field analysis. For this, we propose an encoder-decoder neural network based on the concept of deep autoencoders [57], which have recently been highly successful in finding
meaningful manifold representations [59, 88].

1.4.1 Contributions

We introduce the first network architecture for multi-task learning in light fields, see Fig. 1.7 for a preview. In this work, we concentrate on four state-of-the-art computer vision problems, disparity regression, intrinsic images, reflection separation, and super-resolution. Our fully-convolutional encoder-decoder network can be trained, not only unsupervised to just learn representations but also supervised to solve the above tasks based on the latent space. We employ 3D convolutions to compute features integrated over the whole range of both vertical and horizontal stacks to deal with complex occlusions and reflections. To make the network lighter in terms of GPU memory consumption, we also suggest use of spatial and angular 2D convolutions to approximate 3D convolutions. The networks are trained on datasets rendered with Blender taken from the benchmark [60], as well as a custom random light field generator, which in theory can synthesize an arbitrary amount of training data for reflection separation as well as disparity estimation. We demonstrate via extensive comparisons that our networks can quantitatively and qualitatively outperform existing light-field methods for diffuse and specular separation and intrinsic light fields, and can also robustly compute depth for highly specular scenes. We show that our super-resolution network can have similar performance to the conventional approaches but needs much less input views from the light field, which supports the choice of the proposed architecture.

1.5 List of Projects

Here, I list all of the research projects that I have conducted during the PhD and which are related to light field analysis.

- **A variational model for intrinsic light field decomposition [4]**
  *Asian Conference on Computer Vision, 2016*
  We introduce an intrinsic light field model where input is decomposed into three components: albedo, shading, and specularity. Contrary to the previous work, we estimate specular mask that contains regions of a scene where specularity may be visible from a certain viewing direction. Then, we solve an energy minimization problem with respect to unknown intrinsic components, given the light field of a non-Lambertian scene.

- **Reflection separation in light fields based on sparse coding and specular flow [110]**
  *Vision, Modeling and Visualization Conference, 2016*
  Reflection separation aims to split the input image into diffuse and specular contributions, instead of full intrinsic decomposition. We model reflection separation as an energy minimization problem. The diffuse or body reflection is regularized along the disparity direction and the specular component is forced to be constant along the specular flow. This work is not included in the thesis content.

- **Shadow and specularity priors for intrinsic light field decomposition [6]**
  *International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition, 2017*
  One of the drawbacks of our method [4] is the simplicity of the shading model, which fully relies on the geometry of a scene and does not take into account possible inter-reflections in case of multiple objects. In this work, we improve the variational model [4] by introducing additional prior for cast shadows and inter-reflections, where we analyze both geometrical and color information.
• Accurate depth and normal maps from occlusion-aware focal stack symmetry [109]
  *International Conference in Computer Vision and Pattern Recognition, 2017*
  The quality of the normal map is crucial for estimating (for instance) the shading component.
  Together with a Master’s student, we developed a disparity estimation method that is optimized for disparity labels and for a normal map, outputting piecewise smooth geometry that is suitable for modeling intrinsic components.

• Light field intrinsics with a deep encoder-decoder network [7]
  *International Conference in Computer Vision and Pattern Recognition, 2018*
  With the recent success of deep learning, we explore its potential for light field data. The idea is to solve disparity estimation and reflection separation tasks all together using a single encoder-decoder network. We first extract a small set of features from input epipolar volumes and we then upsample those features into separate decoding pathways.

• Intrinsic light field decomposition and disparity estimation with a deep encoder-decoder network [5]
  *European Signal Processing Conference, 2018*
  We extend our work [7] towards full intrinsic light field decomposition and disparity estimation. We propose a “light” version of the original network, where 3D convolutions are replaced with a sequence of 2D convolutions.

• An epipolar volume autoencoder with adversarial loss for deep light field super-resolution [145]
  *International Conference in Computer Vision and Pattern Recognition, 2019*
  This projects solves angular and spatial light field super-resolution. Together with a Master’s student, we designed a encoder-decoder network that upsamples the input light field two and four times. We adopt the adversarial loss function to improve the sharpness of the high-resolution version.
CHAPTER 2
Disparity Estimation

We introduce a novel approach to jointly estimate consistent depth and normal maps from 4D light fields, with two main contributions. First, we build a cost volume from focal stack symmetry. However, in contrast to previous approaches, we introduce partial focal stacks to be able to robustly deal with occlusions. This idea already yields significantly better disparity maps. Second, even recent sublabel-accurate methods for multi-label optimization recover only a piecewise flat disparity map from the cost volume, with normals pointing mostly towards the image plane. This renders normal maps recovered from these approaches unsuitable for potential subsequent applications. Therefore, we propose regularization with a novel prior linking depth to normals and imposing smoothness of the resulting normal field. We then jointly optimize over depth and normals to achieve estimates for both, which surpass previous work in accuracy based on a recent benchmark.

2.1 Related Work

The methods for disparity estimation from light fields can roughly be classified according to the underlying representation. For this reason, and to fix notation for the upcoming sections, we will briefly review the most common parametrizations of a light field while discussing related methods.

2.1.1 Two-plane representation and sub-aperture views

We consider 4D light fields, which capture the radiance of rays passing through an image plane $\Omega$ and focal plane $\Pi$. By fixing view point coordinates $(s, t)$, one obtains a 2D sub-aperture image in $(x, y)$-coordinates as if captured by an ideal pinhole camera. Matching sub-aperture images means only doing multiview stereo, for which there is a multitude of existing methods. Interesting variants specific to the light field setting construct novel matching scores based on the availability of a dense set of views [53, 55]. A useful transposed representation considers the projections of scene points at a certain distance to the image plane into all sub-aperture views. The resulting view on the light field is called an S-CAM [23] or angular patch [128], and it can be statistically analyzed to obtain disparity cost volumes robust to occlusion [128] and specular reflections [23].

2.1.2 Epipolar plane images

EPIs were pioneered in [18] by analyzing horizontal and vertical slices through the 4D radiance volume. Subsequently, these ideas have been adapted to disparity estimation in various ways. The proposed methods include leveraging of the structure tensor [133] or special filters [125] to estimate the slope of epipolar lines, iterative epipolar line extraction for better occlusion handling [30], fine-to-coarse approaches focusing on correct object boundaries [71], building patch dictionaries with fixed disparities [67], or
training a convolutional neural network for orientation analysis [54]. EPI lines are employed in [63] for distortion correction but they subsequently construct a cost volume from subaperture view matching.

2.1.3 Focal stack

It is well-known that shape can be estimated accurately from a stack of images focused at different distances to the camera [92]. Furthermore, a 4D light field can be easily transformed into such a focal stack, see Fig. 2.1, to apply these ideas to estimate depth. Lin et al. [82] exploit the fact that slices through the focal stack are symmetric around the true disparity, see Fig. 2.2. Correct occlusion handling is critical in this approach, and we improve upon this in Section 2.2. The authors of [117, 128] combine stereo and focus cues to arrive at better results. In particular, Tao et al. [117] propose confidence measures to automatically weight the respective contributions to the cost volumes, which we can also apply as an additional step to increase resilience to noise, see section 2.4.

2.1.4 Regularization and optimization

Regardless of how the disparity cost volume is constructed, more sophisticated methods typically perform optimization of a functional weighting the cost with a regularization term. Key differences lie in the type of regularization and how a minimizer of the cost function is found. Popular optimization methods include discrete methods, such as graph cuts [75] or semi-global matching [58], or continuous methods based on the lifting idea [95], which was recently extended to achieve sublabel accuracy [90]. If one wants to obtain the exact global minimum of cost and regularization term, then the class of regularizers is severely restricted. To become more general and also speed up the optimization, coarse-to-fine approaches are commonly used [45], extending the possible class of regularizers to sophisticated ones such as total generalized variation [20]. Recently, an efficient minimal surface regularizer [45] was constructed using a linear map between a reparametrization of depth and the normals scaled by the local area element. We embrace this idea and extend it to a coupled optimization of depth and normal map to achieve better smoothing of the latter. This will be shown in Section 2.3.

2.2 Focal Stack Symmetry

Good depth estimates are crucial for accurate normal estimation. Several algorithms exploit the depth cues available from multiple views and focus information that can be obtained from light field data.
Chapter 2 Disparity Estimation

Figure 2.2: Left: comparison of Lin et al.’s [82] focal stack \( \varphi \) (a) with our versions \( \varphi^+ \) (b) and \( \varphi^- \) (c) for the green scanline of the light field boxes to the right. One can clearly see that our focal stacks do provide sharper edges near occlusion boundaries while still being pairwise symmetric around the true disparity \( d \). Right: comparison of disparity maps obtained from Lin et al.’s [82] focal stack cost and our proposed cost volume. The numbers show the percentage of Pixels that deviate more than 0.07 from the ground truth.

Among the most robust algorithms with respect to noise is Lin et al.’s [82] cost volume based on focal stack symmetry. This algorithm is based on the observation that for planar scenes parallel to the image plane, focal shifts in either direction from the ground truth disparity result in the same color values. Thus, there is a symmetry in the focal stack around the ground truth disparity \( d \). We will briefly review the foundations and then slightly generalize the symmetry property.

2.2.1 Focal stack symmetry [82]

To refocus the light field, one integrates a sheared version of the radiance volume \( L \) over the sub-aperture views \((u, v)\) weighted with an aperture filter \( \sigma \),

\[
\varphi_p(\alpha) = \int_{\Pi} \sigma(v) L(p + \alpha v, v) dv,
\]

(2.1)

where \( p = (x, y) \) denotes a point in the image plane \( \Omega \), and \( v = (s, t) \) represents the focal point of the respective sub-aperture view. Without loss of generality, we assume that the center (or reference) view of the light field has coordinates \( v = (0, 0) \). To further simplify formulas, we omit \( \sigma \) in the following (one may assume that it is subsumed into the measure \( dv \)). Finally, \( \alpha \) denotes the disparity of the synthetic focal plane.

Lin et al.[82] observed that under relatively mild conditions, the focal stack is symmetric around the true disparity \( d \); that is, \( \varphi_p(d + \delta) = \varphi_p(d - \delta) \) for any \( \delta \in \mathbb{R} \). The conditions are that the scene is Lambertian and there is a locally constant disparity. In practice, it is sufficient for the disparity to be slowly varying on the surfaces. In their work, they leverage this observation to define a focus cost as

\[
s_p^p(\alpha) = \int_0^{\delta_{\text{max}}} \rho(\varphi_p(\alpha + \delta) - \varphi_p(\alpha - \delta)) d\delta,
\]

(2.2)
which is small if the stack is more symmetric around $\alpha$. Above, $\rho(v) = 1 - e^{-|v|/2/(2\sigma^2)}$ is a robust distance function.

The main problem of this approach is that it does not show the desired behaviour near occlusion boundaries. Because pixels on the occluder smear into the background when refocusing to the background, one can observe that the focal stack is actually more symmetric around the occluder’s ground truth disparity instead of the desired background disparity, see Fig. 2.2. Of course, Lin et al.[82] already observed this and proposed handling the problem by choosing an alternative cost for occluded pixels detected by an estimated occlusion map. We propose an alternative approach, which does not require error-prone estimation of an occlusion map and instead only uses light field data.

### 2.2.2 Occlusion-aware focal stack symmetry

To tackle problems around occlusions, we use occlusion-free partial focal stacks. We do not refocus the light field using all sub-aperture views but create four separate stacks using only the views right of, left of, above and below the reference view. The assumption is that the baseline is small enough, so that if occlusion is present it occurs only in one direction of view point shift.

We will see that depending on the occlusion edge orientation, there will be symmetry around the background disparity between the top and bottom or the left and right focal stacks. To see this, we prove the following observation, which will lead to the definition of our modified focus cost volume. Essentially, it refines the focal stack symmetry property defined on the complete stack to symmetry along arbitrary directions of view point shift.

### 2.2.3 Proposition

Let $d$ be the true disparity value of the point $p$ in the image plane of the reference view. Let $e$ be a unit view point shift. Then for all $\delta \in \mathbb{R}$,

\[
\varphi^{-}_{e,p}(d + \delta) = \varphi^{+}_{e,p}(d - \delta),
\]

where

\[
\varphi^{-}_{e,p}(\alpha) = \int_{-\infty}^{0} L(p + \alpha se, se) \, ds
\]

\[
\varphi^{+}_{e,p}(\alpha) = \int_{0}^{\infty} L(p + \alpha se, se) \, ds
\]

(2.3)

are partial focal stacks integrated only in direction $e$.

**Proof.** We assume that the scene is (locally) parallel to the image plane and Lambertian with ground truth disparity $d$. We thus get for any view point $v$

\[
L(p + (d \pm \delta)v, v) = L(p \pm \delta v, v_c),
\]

(2.4)

because view $v$ is the same as the reference view $v_c = (0, 0)$ shifted by $d$. The integrals from (2.3) thus take the form

\[
\varphi^{-}_{e,p}(d + \delta) = \int_{-\infty}^{0} L(p + \delta se, v_c) \, ds,
\]

\[
\varphi^{+}_{e,p}(d - \delta) = \int_{0}^{\infty} L(p - \delta se, v_c) \, ds.
\]

(2.5)

Because $\int_{-\infty}^{0} f(x) \, dx = \int_{0}^{\infty} f(-x) \, dx$ for any real-valued function $f$, we get
\[
\varphi(d + \delta) = \int_{-\infty}^{0} L(p + \delta s e, v_c) \, ds \\
= \int_{0}^{\infty} L(p + \delta s(-e), v_c) \, ds \\
= \int_{0}^{\infty} L(p - \delta s e, v_c) \, ds \\
= \varphi^+(d - \delta).
\]

This completes the proof. \(\square\)

Taking into account the proposition, we modify the cost function (2.2),

\[
s^p_\varphi(\alpha) = \int_{0}^{\delta_{\text{max}}} \min \left( \min \left( \rho(\varphi^-_{(1,0)}, p(\alpha + \delta) - \varphi^+_{(1,0)}, p(\alpha - \delta)), \rho(\varphi^-_{(0,1)}, p(\alpha + \delta) - \varphi^+_{(0,1)}, p(\alpha - \delta)) \right) \right) \, d\delta
\]

where \(\rho\) is the same robust distance function as defined in Equation (2.2).

Note that we create four partial focal stacks corresponding to a crosshair of views around the center view. In future work, we plan to exploit symmetry in other directions to make the method more rotation-invariant. Assuming that occlusion occurs only in one direction—that is, the occluders are not too thin—, then it is always guaranteed that focal stack regions unaffected by occlusion are compared to each other, and lead to zero (or at least very low) cost if \(\alpha\) is the correct disparity. In our experiments, we set \(\sigma = 1\) for cost computation and \(\delta_{\text{max}}\) to one-fifth of the disparity range.

### 2.3 Regularization of Normals

The result from finding a globally optimal solution for the cost with total variation prior is essentially locally flat, even if one uses sublabel relaxation [90]; see Fig. 2.6. The resulting normal field is not useful for interesting applications, such as intrinsic decomposition of the light field. Unfortunately, only priors of a restricted form are allowed if one wants to achieve the global optimum.

Thus, we propose to post-process the result by minimizing a second functional. The key requirements are that we still want to be faithful to the original data term, while at the same time we want to obtain a piecewise smooth normal field. We achieve this by optimizing over depth and normals simultaneously, and linearly couple them using the ideas in [45], which we will describe first.

#### 2.3.1 Relationship between depth and normals

In [45], it was shown that if depth is reparametrized in a new variable \(\zeta := \frac{1}{2} z^2\), the linear operator \(N\) given by

\[
N(\zeta) = \begin{bmatrix}
-\zeta_x \\
-\zeta_y \\
\zeta_{xx} + \zeta_{yy} 2\zeta
\end{bmatrix}
\]

maps a depth map \(\zeta\) to the map of corresponding normals scaled with the local area element of the parametrized surface. Above, \(f\) is the focal length; that is, distance between \(\Omega\) and \(\Pi\), and \((\hat{x}, \hat{y})\) the homogenous coordinate of the pixel where the normal is computed. In particular, \(N\) is spatially varying. \(\zeta_x\) and \(\zeta_y\) denote partial derivatives.
The authors of [45] leveraged this map to introduce a minimal surface regularizer by encouraging small $\|N\zeta\|$. However, we want to impose smoothness of the field of unit length normals. It thus becomes necessary to introduce an unknown point-wise scaling factor $\alpha$ to relate $N\zeta$ and $n$, which will converge to the area element.

### 2.3.2 The final prior on normal maps

Finally, we not only want the normals to be correctly related to depth but we also want them to be piecewise smooth. Thus, the functional for depth reparametrized in $\zeta$ and unit length normals $n$ we optimize is

$$E(\zeta, n) = \min_{\alpha > 0} \int_\Omega \rho(\zeta, x) + \lambda \|N\zeta - \alpha n\|_2 \, dx + R(n).$$

(2.9)

Above, $\rho(\zeta, x)$ is the reparametrized cost function, and $R(n)$ a convex regularizer of the normal map. To obtain a state-of-the-art framework, we extend the total generalized variation [20, 97] to vector-valued functions $n : \Omega \to \mathbb{R}^m$ by defining

$$R(n) = \sup_{w \in C^1_0(\Omega, \mathbb{R}^{m \times m})} \int_\Omega \alpha \|w - Dn\| + \gamma g \|Dw\|_F \, dx.$$  

(2.10)

The constants $\alpha, \gamma > 0$ defining amount of smoothing are user-provided, while $g := \exp(-c\|\nabla I\|)$ is a point-wise weight adapting the regularizer to image edges in the reference view $I$. Intuitively, we encourage $Dn$ to be close to a matrix-valued function $w$ which has itself a sparse derivative, so $n$ is encouraged to be piecewise affine.

### 2.3.3 Optimization

The functional $E$ in (2.9) is overall non-convex, due to the multiplicative coupling of $\alpha$ and $n$, and the non-convexity of $\rho$. Therefore, we follow an iterative approach and optimize for $\zeta$ and $n$ in turn, initializing $\zeta_0$ with the solution from sublabel relaxation [90] of (2.7) and $n_0 = N\zeta_0$. Note that we could just as well embed (2.9) in a coarse-to-fine framework similar to the implementation [45] to make it computationally more efficient, but we decided to evaluate the likely more accurate initialization from global optimization in this work. We now show to perform the optimization for the individual variables. Note that we provide source code for the complete framework on our webpage, so we will omit most of the technical details here and just give a quick tour instead.

### 2.3.4 Optimization for depth

We remove from (2.9) the terms which do not depend on $\zeta$, replace the norms by their second convex conjugates, and linearize $\rho$ around the current depth estimate $\zeta_0$. Consequently, we find that we have to solve the saddle point problem

$$\min_{\zeta, \alpha > 0} \max_{\|p\|_2 \leq \lambda, |\xi| \leq 1} \left\{ (p, N\zeta - \alpha n) + \right.$$  

$$\left. (\xi, p|_{\zeta_0} + (\zeta - \zeta_0)\partial_\zeta p|_{\zeta_0}) \right\}.$$  

(2.11)

The solver that we employ is the pre-conditioned primal-dual algorithm in [22]. Note that the functional (2.11) intuitively makes sense: it tries to maintain small residual cost $\rho$ from focal stack symmetry, while at the same time adjusting the surface $\zeta$ so that $N\zeta$ becomes closer to the current estimate $n$ for the smooth normal field scaled by $\alpha$. Because $N\zeta$ is the area-weighted Gauss map of the surface, $\alpha$ will converge to the local area element.
Chapter 2 Disparity Estimation

2.3.5 Optimization for the normal map

This time, we remove the terms from (2.9) that do not depend on $n$. Given that $\alpha$ should be at the optimum equal to $\| N \zeta \|$, which is now known explicitly, we set $w := N \zeta / \| N \zeta \|$ and end up with the $L^1$ denoising problem

$$\min_{\|n\|=1} \int_{\Omega} \lambda \| N \zeta \| \|w - n\| \, dx + R(n).$$

(2.12)

The difficulty is the constraint $\|n\| = 1$, which makes the problem non-convex. Therefore, we adopt the relaxation ideas in [144], which solves for the coefficients of the normals in a local parametrization of tangent space around the current solution, thus effectively linearizing the constraint. For further details, we refer the reader to [144]. Note that we use a different regularizer, image-driven TGV instead of vectorial total variation, which requires more variables [97]. Regardless, we obtain a sequence of saddle point problems with iteratively updated linearization points, which we again solve with [22].

2.4 Results and Comparisons

We evaluate our algorithm on the benchmark [60] tailored to light field disparity map estimation. The given ground truth disparity is sufficiently accurate to also compute ground truth normal maps using the operator $N$ from the previous section without visible discretization artifacts, except at the discontinuities.

2.4.1 Benchmark performance

By the time of submission, our results with several performance metrics evaluated can be observed in Fig. 2.4 under the acronym OFSY_330/DNR. Note that all of our parameters were tuned on the four training scenes to achieve an optimum BadPix score; that is, the percentage of pixels where disparity deviates by less than 0.07 pixels from the ground truth. The stratified scenes were not taken into account for parameter tuning because they are too artificial, but have of course been evaluated together with the test scenes. In accordance with the benchmark requirements, the parameters are exactly the same for all scenes.
Chapter 2 Disparity Estimation

Figure 2.4: Evaluations on various error metrics from light field benchmark [60]. For the description of those metrics we refer to the benchmark webpage or the paper by Honauer et al. [60]. We show comparisons with five algorithms: EPI1 [67], EPI2 [133], LF [63], LF_OCC [128], MV [60].

At the time of submission, we rank first in BadPix, with a solid first place on all test and training datasets. We also rank first place in the Bumpiness score for planes and continuous surfaces. This demonstrates the success of our proposed joint depth and normal prior to achieve smooth geometry. Finally, we rank first place on the Discontinuities score, which demonstrates the superior performance of the proposed occlusion-aware focal stack symmetry at occlusion boundaries. An overview of the results and a comparison for BadPix on the training scenes can be observed in Fig. 2.7; for further details and more evaluation criteria, we refer to the benchmark web page\footnote{http://www.lightfield-analysis.net}.

The single outlier is the performance on the stratified light field dots, which can be considered as a failure case of our method. This light field exhibits a substantial amount of noise in the lower right regions, see Fig. 2.3, and our method does not produce satisfactory results. One way to remedy this problem is to proceed like [128] and mix the focal stack cost volume with a stereo correspondence cost volume to increase resilience against noise. In contrast to their approach, we computed the mixture weights using the confidence measures in Tao et al. [118], and were able to drastically increase the performance on dots this way. However, for the benchmark evaluation, we decided to submit the pure version of our method with focal stack symmetry cost only. In contrast, in the real-world Lytro data that we evaluate later on, combining costs is very beneficial.
2.4.2 Comparison to focal stack symmetry

In addition to the detailed benchmark, we also compare our occlusion-aware cost volume to original focal stack symmetry [82]. For this, we compute cost volumes using both [82] and the method proposed in Section 2.2, and compute a globally optimal disparity map for both methods using sublabel relaxation, with the smoothness parameter tuned individually to achieve an optimal result. Our results can be observed in Fig. 2.6. We perform significantly better in terms of error metrics, easily visible from the behaviour at occlusion boundaries, in particular regions that exhibit occlusions with very fine detail.

Please note that these images do not show the results obtained by Lin et al.’s final algorithm in [82], but only those achieved by simply using the focal stack symmetry cost function (2.2) because we aim at a pure comparison of the two cost volumes.

2.4.3 Normal map accuracy

In another run of experiments, we verify that our joint scheme for depth and normal map regularization is an improvement over either individual regularization of just the depth map with the minimal surface regularizer in [45], as well as just regularization of the normal map using [144]. Our results can be observed in Fig. 2.6. Normal map regularization is non-convex and fails to converge towards the correct solution starting with the piecewise fronto-parallel intialization from sublabel relaxation [90]. The better result is achieved by smoothing the depth map directly. However, when imposing the same amount of smoothing as in our framework, it performs worse smoothing on planar surfaces and fails to preserve details.

2.4.4 Real-world results

To test the real-world performance of our proposed method, we evaluate the light fields captured with the Lytro Illum plenoptic camera. One can clearly see that our method performs relatively well, even on non-Lambertian objects such as the origami crane in Fig. 1.4 or the saxophone in Fig. 2.5 (top row). Figure
Figure 2.5 (bottom row) shows results on the Lego truck from the Stanford light field archive for comparison and proof that our method works with even a relatively large baseline.
Figure 2.6: Comparison of normal maps obtained with different methods. Numbers show mean angular error in degrees. The result obtained from sublabel relaxation is overall still fronto-parallel despite the large number of 330 labels. Consequently, the non-convex normal smoothing with \cite{144} fails to converge to a useful solution because the initialization is too far from the optimum. Smoothing the depth map using \cite{45} yields visually similar results than our method but we achieve lower errors and smoother surfaces while still preserving details such as the eyes of the teddy in dino. For visualization, normals $\mathbf{n}$ have been transformed to RGB space via $(r, g, b) = \frac{1}{2}(\mathbf{n} + [1 1 1]^T)$. 

Figure 2.7: Our disparity and normal maps for the training datasets compared to the results of the other methods listed on the benchmark at the time of submission. For all datasets, we achieve the lowest error for both, measured in percentage of pixels with a disparity error larger than 0.07 pixels (marked red in the disparity map), and mean angular error in degrees for the normal map, respectively.
2.5 Conclusions

We have presented occlusion aware focal-stack symmetry as a way to compute disparity cost volumes. The key assumptions are Lambertian scenes and slowly varying disparity within surfaces. Our experiments show that our proposed data term is the most accurate light field depth estimation approach to the submission date based on the recent benchmark [60]. It performs particularly well on occlusion boundaries and in terms of the overall correctness of the disparity estimate. As a small drawback, we get a slightly reduced noise resiliency because we only operate on a crosshair of views around the reference view as opposed to the full light field. However, on very noisy scenes, we can improve the situation by confidence-based integration of stereo correspondence costs into the data term; as suggested in previous literature on disparity estimation with focus cues [118, 128].

With additional post-processing using joint depth and normal map regularization, we can further increase accuracy slightly. In particular, we can obtain accurate and smooth normal fields that preserve small details in the scene. We again outperform previous methods on the benchmark datasets. Further experiments on real-world scenes show that we can deal with significant amounts of specularity, and obtain depth and normal estimates suitable for challenging applications such as intrinsic light field decomposition [4, 5], which are described in the next chapter 3.
Variational Models for Intrinsic Light Field Decomposition

We present a novel variational model for intrinsic light field decomposition, which is performed on 4D ray space instead of a traditional 2D image. Because most existing intrinsic image algorithms are designed for Lambertian objects, their performance suffers when considering scenes that exhibit glossy surfaces. In contrast, the rich structure of the light field with many densely sampled views allows us to cope with non-Lambertian objects by introducing an additional decomposition term that models specularity. Regularization along the epipolar plane images further encourages albedo and shading consistency across views.

We extend the original model by introducing a novel prior to cope with cast shadows and inter-reflections. In contrast to the first approach, which models inter-reflection based only on geometry, we model indirect shading by combining geometric and color information. We compute a shadow confidence measure for the light field and then use it in the regularization constraints. Extensive experiments show that this design choice improves the estimation of shading component by a large margin. Another improvement is adding frequency analysis in specularity estimation, which is based on the assumption that high variations in pixel intensities of a 3D point in sub-aperture views are caused by specularity. The new priors are embedded in a recent framework to decompose the input light field into albedo, shading, and specularity.

In the evaluations of our method on real-world datasets captured with a Lytro Illum plenoptic camera, we demonstrate the advantages of our approach with respect to intrinsic image decomposition and specular removal.

3.1 Related Work

Intrinsic images have been a challenging research topic for many years. They were introduced by Barrow and Tenenbaum [13] to divide an observed image into the product of a reflectance and illumination image. According to Land and McCann [77], large discontinuities in pixel intensities correspond to changes in reflectance, and the remaining variation corresponds to shading. Consequently, they proposed a Retinex theory that was successfully extended and implemented for intrinsic image decomposition by Tappen et al. [122], Chung et al. [27], Grosse et al. [46], Finalayson et al. [37, 38], and many others.

Besides the Retinex approach, it is common to include additional regularization terms that describe certain physical properties of intrinsic components. Barron and Malik [10, 11, 12] introduce priors of reflectance, shape, and illumination to recover intrinsic images. Shen et al. [102] employ texture information. Finalayson et al. [36] search for an invariant image, which is independent of lighting and shading. Gehler and Rother [40] model reflectance values drawn from a sparse set of basis colors. Bell
et al. [15] also assume that reflectance values come from a predefined set that is unique for every image and they then iteratively adjust reflectance values in this set.

Recently, a significant improvement in intrinsic image decomposition was achieved by using richer types of input data. Having a sequence of images with depth information available allows us to penalize albedo and shading consistency between different views, see Lee et al. [79]. Depth or disparity information allows to incorporate spatial dependencies between pixels to construct shading prior, Jeon et al. [64]. Chen and Koltun [25] develop a model based on RGB-D information. They separate shading into two components: direct and indirect irradiance, which significantly improved decomposition results. Barron and Malik [12] use depth to extend their SIRFS model [11] such that it is applicable for natural scenes.

Although decomposition algorithms can now achieve spectacular results for Lambertian scenes, their performance suffers in the non-Lambertian case in the presence of highlights or specularity. Therefore, in this work, we will make use of the rich structure in the light field to estimate specularity for non-Lambertian objects. According to the dichromatic model introduced by Shafer [101], diffuse and specular reflections behave differently. Diffuse objects reflect incident light in multiple directions equally, thus, their color is independent of viewpoint. Meanwhile, specular objects reflect light in a certain direction that depends on orientation, and thus their color depends on viewpoint, light source color, and physical material properties.

Blake and Bülthoff [17] made an extensive analysis of specular reflections and propose a strategy for recovering 3D structure using specularity. Swaminathan et al. [112] study the photometric properties of specular pixels and model their motion depending on the surface geometry. Sulc et al. [110] proposed a specular removal approach for light fields that is based on sparse coding and specular flow. Gryaditskaya et al. [47] develop a light field gloss editing technique to change the roughness parameters of surfaces in the scene. Wang et al. [127] remove specularity by clustering specular pixels into "unsaturated" and "saturated" categories, with further color refinement. Adato et al [1] model specular flow with non-linear partial differential equations. Tao et al. [120, 121] introduced depth estimation for glossy surfaces. They leverage the light field structure to cluster pixels in specular and specular-free groups, and they then remove specular components from the input light field.

### 3.2 Variational Formulation

We model an intrinsic light field as a function

\[ L(r) = A(r)S(r) + H(r), \]  
(3.1)

where the radiance \( L \) of every ray \( r \) is decomposed into albedo \( A \), shading \( S \), and specular component \( H \). The functions \( L, A, S, H : \mathcal{R} \to \mathbb{R}^3 \) map ray space to RGB values. Albedo represents the color of an object, independent of illumination and camera position. Shading describes intensity changes due to illumination, inter-reflections, and object geometry. Finally, specularity represents highlights that occur in case of non-Lambertian objects, which depend on illumination, object geometry, and camera position.

The common assumption in the literature related to intrinsic image decomposition is to model the shading component as mono-chromatic [40, 46, 122]. However, in the case of multiple light sources or non-Planckian light, this modeling assumption is not sufficient. Thus, we further decompose shading into mono-chromatic shading \( s \) and trichromatic light source color \( C \),

\[ S(r) = s(r)C(r). \]  
(3.2)
Figure 3.1: A variational intrinsic light field decomposition into the intrinsic components albedo, shading and specularity. This light field is captured with Lytro Illum plenoptic camera, 9 × 9 views, resolution 434 × 625.

We directly compute the illumination component \( C \) in a pre-processing step with the illuminant estimation algorithm that was developed by Yang et al. [140] applied to the center view, assuming that it will be similar across views. After the illumination color is computed, we exclude it from the original light field by switching to the new decomposition model

\[
\frac{L(r)}{C(r)} = A(r) s(r) + \frac{H(r)}{C(r)}
\]  

(3.3)

which is illumination color free. Vector division is to be understood component-wise.

We present two models for intrinsic light fields, one built on top of the other. The first model solves the decomposition for three components: albedo \( A \), shading \( s \), and specularity \( H \). The second model additionally recovers cast shadows and inter-reflections \( S \), which improves the quality of decomposition by a large margin. We start with the simple model, where three intrinsic components are computed.

3.2.1 Model 1: albedo, shading and specularity

The model is presented in our work [4]. When the light field \( L \) is transformed to the illumination color free representation (3.3), its intrinsic model

\[
L(r) = A(r) s(r) + H(r)
\]  

(3.4)

consist of three unknown components, where \( A \) and \( H \) are trichromatic, and \( s \) is monochromatic; see Fig. 3.1 for the illustration on the intrinsic components for the real world scene. As a further simplification, we obtain System (3.4) in linear form

\[
L_{\text{log}}(r) = A_{\text{log}}(r) + s_{\text{log}}(r) + H_{\text{log}}(r)
\]  

(3.5)

by applying the logarithm. We now want to solve (3.5) with respect to albedo, shading, and specularity.

System (3.5) is ill-posed because its number of variables is three times larger than the number of equations. To select a solution that agrees with physical meaning of intrinsic components, we pose it as an inverse problem and introduce a number of constraints or regularization terms for albedo, shading, and specularity. As usual, dependence of \( H_{\text{log}} \) on all arguments except \( r \) is ignored during optimization and it is estimated as another independent component. We thus solve a global energy minimization problem, where we weight the residual of (3.5) with different priors and regularization terms,

\[
\arg\min_{(A_{\text{log}}, s_{\text{log}}, H_{\text{log}})} \left\{ \|L_{\text{log}}(r) - A_{\text{log}}(r) - s_{\text{log}}(r) - H_{\text{log}}(r)\|_2^2 + \ldots + E_a(A_{\text{log}}) + E_{\text{sh}}(s_{\text{log}}) + E_s(H_{\text{log}}) + J(A_{\text{log}}, s_{\text{log}}) \right\}.
\]  

(3.6)
Chapter 3 Variational Models for Intrinsic Light Field Decomposition

3.2.2 Model 2: albedo, direct shading, indirect shading and specularity

As in the previous section 3.2.1 we consider a case where the scene is illuminated with a single white light source. In case of non-white illumination or multiple light sources, the model should be multiplied with the illumination color, which can be estimated with color constancy algorithms [115, 140]. This model is described in our paper [6]. Inspired by Chen and Koltun [25], we extend the model (3.4) by introducing an additional term for cast shadows and inter-reflections. Thus, the new decomposition model

\[ L(r) = A(r)s_d(r)S_i(r) + H(r) \]  

(3.7)

has components \( s_d \) and \( S_i \) that describe direct and indirect shading, see Fig. 3.2. Direct shading can be understood as the shading that an object would have if it were alone in the scene; that is, without shadows or reflected light. The second component \( S_i \) models inter-reflections and cast shadows. However, in contrast to Chen and Koltun [25], where \( S_i \) is modeled as smooth in 3D space, we model it using a shadow confidence measure that is proportional to shadow intensities in the scene. This allows us to handle hard shadows, where the spatial smoothness assumption is violated. In addition, given that even
under white light inter-reflections depend on the object colors, we model $S_i$ as tri-chromatic, while $s_d$ is mono-chromatic.

To obtain a linear decomposition equation from (3.7), we apply the logarithm,

$$L^\log(r) = A^\log(r) + 1s_d^\log(r) + S_i^\log(r) + H^\log(r, A, s_d, S_i, H)$$

with $H^\log = 1 + \frac{H(r)}{A(r)s_d(r)S_i(r)}$, (3.8)

and ignore the dependence of the specular component $H$ on albedo and shading by treating $H^\log$ as another independent variable. The decomposition (3.8) is ambiguous, and we thus need strong prior assumptions on all of the unknown intrinsic components. In addition, we need to leverage the light field structure and enforce a consistent decomposition across all sub-aperture views. We thus reformulate the decomposition problem as minimization of the energy

$$\arg\min_{(A^\log, s_d^\log, S_i^\log, H^\log)} \left\{ \|D(r)\|^2_2 + E_a(A^\log) + E_d(s_d^\log) + E_i(S_i^\log) + \ldots + E_s(H^\log) + J(A^\log, s_d^\log, S_i^\log) \right\}.$$  

(3.9)

The first data term enforces consistency of the decomposition because the residual $D = L^\log - A^\log - 1s_d^\log - S_i^\log - H^\log$ should be small. To reduce the complexity of the problem (3.9) we substitute specularity variable with the difference $L^\log - A^\log - 1s_d^\log - S_i^\log$ between input light field and other intrinsic components. The follow-up energies $E$ denote the previous priors, which will be explained later, while the term $J$ enforces consistency with the light field structure. The idea is to encourage smoothness of albedo and the two shading components along the projections of scene points in the epipolar plane images, and to encourage spatial smoothness on each sub-aperture view by means of a total generalized variation regularizer [20].

### 3.3 Specularity

We start by describing the priors, which are essential for obtaining an accurate solution for intrinsic light field decomposition from the variational models (3.6) and (3.9). In this section, we describe the specular prior in the variational energies (3.6), (3.9). We first discuss the modeling assumptions, we then show how to compute a mask for candidate specular pixels based on these assumptions, and we finally construct the prior $E_s$.

#### 3.3.1 Modeling assumptions

We combine several approaches to model specularity [1, 17, 106, 112, 120, 121, 123]. According to the specular motion model [17, 112], specularity changes depend on surface geometry, see Fig. 3.3. For instance, regions of low curvature on a specular object create color intensity changes within different views. Specular regions of high curvature result in high pixel intensities in all sub-aperture views. Thus, curvature information can be useful when estimating specularity.

However, in practice the curvature estimation is very sensitive to inaccuracies of the 3D model of a scene. Imperfect disparity maps lead to a certain amount of noise in the estimated spatial coordinates, and thus curvature information becomes highly unreliable. Instead of using curvature information directly, we therefore propose a heuristic approach that estimates candidate regions where specularity or highlights can occur. Our main modeling assumptions are thus:
Chapter 3  Variational Models for Intrinsic Light Field Decomposition

Figure 3.3: Specular motion model, the illustration is taken from [17]

S1. Specularity is either view-dependent or "non-saturated", and thus has high variation in pixel intensities over different sub-aperture views, or it is "saturated", and thus appears bright throughout all sub-aperture views.

S2. If a projected 3D point has high pixel intensities and its color is constant across all sub-aperture views, then the point may be part of a specular surface.

S3. If a projected 3D point has high variation in pixel intensities and the color of the corresponding rays changes across sub-aperture views, then the point may belong to a specular surface.

S4. If a point is classified as specular, then it is a part of specular surface and its local neighborhood in $\mathbb{R}^3$ may result in specular pixels from a certain viewing angle.

S5. The distribution of specularity is sparse.

Potential specular objects are identified based on magnitude and variation of pixel values over different views. We compute a specular mask for the center view and propagate it to the remaining views according
Chapter 3 Variational Models for Intrinsic Light Field Decomposition

Figure 3.5: Potential locations for specularity that were detected with our algorithm described in Section 3.3. From left to right, we present the center view of the light field, preliminary specular mask $h^*$ (3.12), final specular mask $h$ (3.14). The first row corresponds to the synthetic light field generated with the Blender software, the middle row shows the real world light field captured with Lytro Illum plenoptic camera, and the bottom row illustrates the specular mask for the light field captured with a camera array.

to disparity.

We mask out occlusion boundaries while computing the specular mask, thus we assume that there is no specularity on object edges. Dealing with specularity on occlusion boundaries is left for future work.

3.3.2 Computing the specular mask

Our proposed algorithm proceeds in five steps:

1. Under the assumption of known disparity $d$, we construct angular patches $\omega_p$ for every scene point $P$. Let $\Omega_c$ be the image plane for the center view, and $V = \{(s_1, t_1), \ldots, (s_N, t_N)\}$ the set of remaining $N$ view points. For every $p \in \Omega_c$, we compute the vector $\omega_p$ of color intensity changes with respect to $V$ according to

$$\omega_p^i = L_i(p + v_id(p)), \quad i = 1, \ldots, N. \quad (3.10)$$
where \( v_i = (s_c - s_i, t_c - t_i) \) is the viewpoint displacement and \( d(p) \) the estimated scalar disparity of \( p \). We then reshape \( \omega_p \) in the 2D image that represents how the scene point \( P \) looks in all of the views, \( V \). The angular patch represents the corresponding 3D point’s color over all of the sub-aperture views, see Fig. 3.4. We omit writing dependency of \( P(r) \) to keep the notation short.

2. We apply a Fourier transform to every angular patch \( \omega_p \), and search for significant low-frequency oscillations in every color channel of the patch according to assumption (S3). If these oscillations exist, then we conclude that the angular patch represents a specular point. We assume that high frequency oscillations are caused by noise and are not caused by specularity; see Fig. 3.4 for an overview of the proposed technique. Our algorithm proceeds in four steps:

- Apply Fourier transform to logarithm of \( \omega_p \) for each color channel independently.
- Compute the ratio
  \[
  r(P) = \frac{f_2(P)}{f_1(P)}
  \]
  between amplitude of the lowest frequency \( f_2(P) \) and amplitude of the frequency \( f_1(P) \) that corresponds to the mean value of the angular patch.
- Choose the largest ratio \( r(P) \) over the color channels. If \( r(P) \) is close to 1, then there is a significant low-frequency component, which we interpret as specularity. Usually, specularity exhibits high pixel intensity in at least one sub-aperture view. Thus, we multiply \( r(P) \) with the maximum brightness of the patch \( \omega_p \) to filter out other low frequency color variations.
- Compute a specular mask
  \[
  h^* = \begin{cases} 
  1, & \text{if } r \geq \tau_f \\ 
  0, & \text{otherwise}
  \end{cases}
  \]
  by thresholding low values of \( r(x) \).

3. Refine the specular mask \( h^* \) according to assumptions (S1) and (S3):

- Filter out a percentage \( \%n_{var} \) of pixels that have low luminance variation \( \sigma(\omega_p) \), where, by \( \Omega^*_c \), we define a set of remaining pixels.
- Exclude occlusion boundaries from \( \Omega^*_c \). To find occlusion boundaries, we compute the \( k \)-nearest neighbors in the image domain, and corresponding spatial coordinates in \( \mathbb{R}^3 \). If neighboring pixels in \( \Omega^*_c \) are far away in \( \mathbb{R}^3 \), with distances larger than \( d_{occ} \), then we classify those pixels as occlusion boundaries.
- From the remaining pixels, we finally exclude the percentage \( \%n_{conf} \) with the lowest confidence scores similar to the approach proposed by Tao et al. [120]. To compute confidence, we cluster the corresponding values of \( \omega_p \) in two groups using K-means. Let \( m(p) \) be the cluster centroid with the larger mean \( \mu(m) \). The confidence is computed as
  \[
  c(p) = \exp \left( -\frac{1}{\sigma_{spec}^2} \left( \frac{\beta_0}{\mu(m)} + \frac{\beta_1}{\xi(m)} \right) \right),
  \]
  where \( \xi(m) \) denotes the sum of all distances within the cluster. The confidence score grows with mean intensity and variation within the brightest cluster. Thus, we obtain pixels with varying values within sub-aperture views. Here, \( \beta_0 \) and \( \beta_1 \) are user-defined parameters that control exponential decay of brightness and distance terms, \( \sigma_{spec} \) scales the confidence function.

4. Identify pixels where intensity is high and color not changing within all sub-aperture views according to assumption (S2). According to Tian and Clark [123], regions with high unnormalized...
Wiener entropy, which is defined as the product of RGB values over all pixels, are likely to be specular. We adopt their approach and also identify those regions.

5. Combine remaining points of the specular mask \( h^* \) (3.12) found in Steps 3 and 4 into the final specular mask

\[
h = \begin{cases} 
1, & \text{specular} \\
0, & \text{non-specular}
\end{cases}
\] (3.14)

which is then grown according to assumption (S4) to include all \( k_{\text{spec}} \)-nearest neighbors for each specular pixel in the initial mask.

An example specular mask for different datasets is shown in Fig. 3.5.

### 3.3.3 Final prior on specularity

The specular component should be non-zero only within the candidate specular region given by the mask \( h \) defined above. Therefore, we strongly penalize non-zero values outside this region by defining the final sparsity prior as

\[
E_s(H_{\log}) = \lambda_{\text{spec}} \int_{\mathbb{R}} \gamma_w (1 - h) \| H_{\log}(r) \|_2^2 \, dr + \lambda_{\text{sparse}} \| H_{\log} \|_1.
\] (3.15)

where \( \gamma_w \gg 0 \) is a constant. We include an additional sparsity norm on \( H_{\log} \) to account for assumption (S5).

### 3.4 Albedo

To model albedo, we combine the ideas of Retinex theory, which is widely used to decompose an image into shading and reflectance components [27, 122, 135], with the idea that pixels with equal chromaticity are likely to have similar albedo [25, 36, 79]. Thus, the prior for albedo is the sum of two energies, \( E_a(A_{\log}) = E_{\text{retinex}}(A_{\log}) + E_{\text{chroma}}(A_{\log}) \), corresponding to these two models.

Under the simplifying assumption that image derivatives in the log-domain are caused by either shading or reflectance, we classify the derivative at every ray as caused by shading or albedo. The idea is to compute a modified gradient field \( \hat{g} \), which assigns a zero value to all derivatives that are caused by shading. The derivative classification uses an approach similar to the Color Retinex that was used in [27, 46]. A partial spatial derivative \( L_x \) of the light field is classified as albedo if the neighboring RGB vectors into the direction of differentiation are not parallel, or if it is above a certain magnitude. Thus, the modified derivative is

\[
\hat{g}_x = \begin{cases} 
L_x, & \text{if } \rho_{x+1,y} \cdot \rho_{x,y} < \tau_{\text{col}} \text{ or } |L_x| > \tau_{\text{grad}}, \\
0, & \text{otherwise.}
\end{cases}
\] (3.16)

Here, \( \rho = (R, G, B)^T \), the constant \( \tau_{\text{col}} > 0 \) is a threshold above which two vectors are assumed to be parallel, and \( \tau_{\text{grad}} > 0 \) is another user-defined constant. Similarly, we estimate the modified partial derivative \( \hat{g}_y \) in the second spatial direction.

The gradient of the albedo should be equal to the gradient field modified by retinex, thus we finally obtain the retinex energy

\[
E_{\text{retinex}}(A_{\log}) = \lambda_{\text{retinex}} \int_{\mathbb{R}} \| \partial_x A_{\log}(r) - \hat{g}_x(r) \|_2^2 + \| \partial_y A_{\log} r - \hat{g}_y(r) \|_2^2 \, dr.
\] (3.17)
Figure 3.6: Different chromaticity representations, based on RGB, c1c2c3 and HSV color spaces. We illustrate the center view of the $9 \times 9 \times 512 \times 512 \times 3$ synthetic light field generated with Blender software and plugin provided by Honauer et al. [60].

The second regularization term is based on chromaticity similarities between adjacent rays. By definition, the chromaticity is supposed to be illumination free. The basic idea is that if two neighboring rays of the same view have close chromaticity values, then they have the same albedo. Because chromaticity images may still contain shadows, we follow the previous work [94] and use several illumination-free representations based on RGB, HSV, and c1c2c3 color spaces to compute the chromaticity difference, see figure 3.6.

- **RGB color space**
  
  $$c_k(r) = \frac{\rho_k(r)}{\sum_{k=1}^{3} \rho_k(r)}$$  \hspace{1cm} (3.18)

- **HSV color space** The hue $H(r)$ of a ray $r$ refers to the pure color that it resembles. For instance all tints, tones or shades of red are the same. Saturation $S(r)$ describes how white the color is. Value $V(r)$ represents how dark the color is. A value 0 is black, with increasing lightness moving away from black. Because conversion formulas are implemented in all modern coding languages, we do not introduce formulas for computing HSV representation.

- **c1c2c3 color space**
  
  $$c_k(r) = \arctan \left( \frac{\rho_k(r)}{\max(\rho_{(k+1)\mod 3}(r), \rho_{(k+2)\mod 3}(r))} \right)$$  \hspace{1cm} (3.19)
where $\rho_k(r)$ is the $k^{th}$ RGB color component of ray $r$.

For the representations, we obtain different ray-wise chromaticity value candidates $\chi(r)$, which all enter the final score.

We use the chromaticity measure, which gives a weight $\alpha_{r,q}$ for how likely it is that two rays $r$ and $q$ have the same albedo,

$$\alpha_{r,q} = \exp \left( - \frac{\min_k \| \chi(r) - \chi(q) \|^2}{\sigma^2_{\chi}} \right),$$  \hspace{1cm} (3.20)

The minimum for $\alpha_{r,q}$ is computed over the different ways to compute the chromaticity values that were introduced earlier, the $\sigma_{\chi}$ is a user defined parameter.

To remove artifacts caused by specularity from the albedo component, we can give more weight to points that have high specularity confidence. Thus, to compute weights $\alpha_{r,q}$, we propose to use flexible thresholds for chromaticity differences, based on our novel specular confidence $h$ as described in Section 3.3.

$$\alpha^*_{r,q} = \left( 1 + \gamma_a \max \left( h(r), h(q) \right) \right) \alpha_{r,q}. \hspace{1cm} (3.21)$$

The weights for the pairs of rays above are defined in (3.20). We omit $^*$ for simplicity and assume that we use equation (3.21) to compute weights for albedo component.

The chromaticity energy

$$E_{\text{chroma}}(A^{\log}) = \lambda_{\text{chroma}} \int_{R} \sum_{N_A(r)} \alpha_{r,q} \| A^{\log}(r) - A^{\log}(q) \|^2 \, dr$$ \hspace{1cm} (3.22)

now penalizes more dissimilarity of albedos that have a higher chromaticity measure $\alpha_{r,q}$. Note that we use a mixed continuous/discrete notation for $r$ and $q$ because our choice of neighborhood is inherently discrete, while we require a variational rayspace model in the optimization framework, see Section 3.6.

To construct the neighborhoods $N_A(r)$ for every ray $r \in R$, we impose the assumption that spatially close points in $\mathbb{R}^3$ probably have similar albedo. We select $k_A$ nearest neighbors in $\mathbb{R}^3$ for the point $P$ on the scene surface intersected by $r$, and randomly choose $m_A$ out of $k_A$ neighbors. We believe that this connectivity strategy has several advantages over fully random connectivity: by defining neighbors, we increase the chance to meet points with similar chromaticity; and by random connectivity within neighboring points, we avoid disconnected chromaticity clusters.

### 3.5 Shading

To describe the shading prior, we start with the mono-chromatic version that is used in the model (3.4). We then design a shadow detection model with light field data, which to our knowledge is the first time that this problem has been addressed for light fields. We apply the estimated shadow score to model cast shadows and inter-reflections explicitly, which results in more consistent shading described in the model (3.7) compared to previous approach, with better identification of soft and hard shadows.
3.5.1 Simple shading

The shading prior form the energy (3.6) is also the sum of two components, \( E_{\text{sh}} = E_{\text{normal}} + E_{\text{spatial}} \). To model the first component, we adopt the well-known assumption [25, 79, 119] that scene points which are spatially close to each other and share the same orientation are likely to have similar shading. To facilitate this, we construct the 6D set

\[
\Gamma := \{ (P(r), n(P(r))) : r \in \mathcal{R} \},
\]

where \( P(r) \) is again the point of the scene surface intersected by \( r \), and \( n(P(r)) \) is the corresponding outer normal. The set of neighbours \( N_S(r) \) now consists of the \( k_S \)-nearest neighbours of \( r \in \mathcal{R} \) in the 6D space \( \Gamma \). The regularization term

\[
E_{\text{normal}}(s^{\log}) = \lambda_{\text{normal}} \int_\mathcal{R} \sum_{q \in N_S(r)} (s^{\log}(r) - s^{\log}(q))^2 \, dr
\]

(3.23)

thus penalizes shading components to be the same if corresponding 3D points are spatially close to each other and their outer normals have similar orientations.

To account for indirect shading, which is caused by inter-reflections between objects in a scene, we also include a purely spatial regularization term

\[
E_{\text{spatial}}(s^{\log}) = \lambda_{\text{space}} \int_\mathcal{R} \sum_{q \in N_D(r)} (s^{\log}(r) - s^{\log}(q))^2 \, dr,
\]

(3.24)

where the neighborhood \( N_D(r) \) denotes the \( k_D \) nearest neighbors of the 3D scene point first intersected by \( r \).

3.5.2 Cast shadow detection

To identify regions with cast shadows and remove them from the direct shading component, we compute a shadow score \( \beta \in [0,1] \). We first estimate the point-wise confidence \( \beta_{pw} \) of each ray to be shadowed, and then encourage shadowing to be consistent across all sub-aperture views. From these, we will later compute a shadow boundary score to set up the smoothness priors for the intrinsic shading components.

Point-wise shadow confidence

Inspired by the work of Xiao et al. [137], we use the spatial locations \( P \), surface normals \( n \), and chromaticity \( \chi \) of the scene to decide which rays are likely to be shadowed. Because the chromaticity images may still contain shadows, we use several illumination-free representations based on \( RGB \), \( HSV \), and \( cr2c3 \) color spaces described in Section 3.4.

Our main assumption is that if two points are spatially close to each other, share the same orientation, and have similar values in chromaticity space, but different intensities, then the point with lower intensity is likely to be shadowed. To put this into formulas, we compute a weight \( \theta_{r,q} \) that measures likely similarity in shading between rays \( r \) and \( q \), in the sense that \( \theta_{r,q} \) is close to one if the two points have similar shading. We set

\[
\theta_{r,q} = w_r^\chi w_r^n w_{r,q}^d,
\]

(3.25)
Chapter 3 Variational Models for Intrinsic Light Field Decomposition

<table>
<thead>
<tr>
<th>center view</th>
<th>point-wise shadow confidence</th>
<th>final shadow score</th>
<th>shadow boundary</th>
</tr>
</thead>
</table>

Figure 3.7: Shadow mask described in Section 3.5. From left to right, we present the center view on the light field, point-wise shadow confidence $\beta_{pw}$ computed with equation (3.30), final shadow score $\beta$ obtained from the solution of the energy minimization problem (3.32) and shadow boundary $\delta_\beta(r)$ obtained with (3.33). Again, the first row corresponds to the synthetic light field, the middle row shows the real world light field captured with Lytro Illum plenoptic camera, and the bottom row illustrates the shadow mask for the light field captured with a camera array.

where

$$w_{r,q}^\chi = \exp\left(-\frac{\min_\chi \| \chi(r) - \chi(q) \|^2}{\sigma_\chi^2}\right), \quad (3.26)$$

$$w_{r,q}^n = \exp\left(-\frac{\cos(n(r), n(q)) - 1}{\sigma_n^2}\right), \quad (3.27)$$

$$w_{r,q}^d = 1 - \frac{\max_{q \in N_M^D(r)} \| P(r) - P(q) \|^2}{\| P(r) - P(q) \|^2}. \quad (3.28)$$

Here $N_M^D(r)$ denotes subset of $k_D$ among the $M$-nearest neighbors of $r$ in 3D space within the same object as $r$. This neighborhood is designed to restrict comparison to suitable rays within the same object. Parameters $\sigma_n$ and $\sigma_\chi$ are defined by the user. The minimum for $w^\chi$ is computed with equation (3.20) over the different ways to compute chromaticity values introduced in Section 3.4.

Based on the similarity scores, we can then estimate color

$$m(r) = \left( \sum_{q \in N_M^D(r)} \theta_{r,q} L(q) \right) \Bigg/ \left( \sum_{q \in N_M^D(r)} \theta_{r,q} \right), \quad (3.29)$$

for each ray by computing a weighted average over its neighbors.
Finally, the point-wise shadow confidence score $\beta_{pw}$ is based on the observation that the estimated and actual intensities $v(\cdot)$ should be the same for an unshadowed ray. Here, by intensity we mean average over all color channels $R$, $G$ and $B$. Thus, we set ray-wise

$$
\beta_{pw} = 1 - \exp \left( -\frac{1}{\sigma_v^2} \max (v(m) - v(L), 0) \right),
$$

with $\sigma_v^2$ defined in Table 3.1 See Fig. 3.7 for an example of a point-wise shadow confidence map.

**Consistency across the light field**

While the locations of shadows are correct in most of the cases, point-wise estimation suffers from noise, and some shadow-free surfaces are classified as shadowed. To correct the initial shadow estimation, we minimize an energy $E(\beta)$, where we impose smoothness and consistency by embedding it in the optimization framework [41] similar to the main energy minimization problem (3.9). The optimization procedure is discussed in Section 3.6.

In addition to light-field consistency, shadowed regions $\beta$ are required to coincide with those regions where not only point-wise shadow confidence is high but also intensity is low. For this, we combine the point-wise confidence with a shadow score derived from the bright channel concept for shadow detection introduced by Panagopoulos et al. [94]. We want our shadow score to be equal to the inverse bright channel cue

$$
\overline{br}(r) = 1 - \max_{k \in \{R,G,B\}} L_{\rho_k}(r),
$$

in regions where point-wise shadow confidence $\beta_{pw}$ is high. We also include a sparsity prior to remove regions with low confidence altogether.

Thus, the final energy that we minimize is

$$
E(\beta) = \lambda_d \int_{R} \beta_{pw}(r) \| \beta(r) - \overline{br}(r) \|^2 \, dr + \lambda_\beta^s \| \beta(r) \|_1 + J(\beta),
$$

defined over all sub-aperture views. In the regularizer $J$, we weight the spatial regularization with $1 - w_{r,q}^x$ because the shadow should be the same if chromaticity is changing, while discontinuities that are related to reflectance changes should be smoothed. Fig. 3.7 shows final shadow confidence after optimization.

**Shadow boundary $\delta_\beta$**

To allow for shading discontinuities across the shadow boundary, we compute a mask $\delta_\beta$ for its likely location via the windowed inherited variation [139]

$$
\delta_\beta(r) = \sqrt{L_x^2(r) + L_y^2(r)},
$$

where

$$
L_x(r) = \sum_{q \in R(r)} g_{r,q} \langle \partial_x \beta \rangle_q,
$$

$$
L_y(r) = \sum_{q \in R(r)} g_{r,q} \langle \partial_y \beta \rangle_q,
$$

are sums of the gradients over the rectangular patch $R(r)$ centered at $r$, weighted with Gaussian $g$. In our experiments we fix patch size 15 and use $\sigma = 3$ for Gaussian filtering.
Thus, the shadow boundary $\delta_{\beta}$ reflects the overall change in gradients of $\beta$. See Fig. 3.7 for an example.

We are now ready to introduce priors for direct and indirect shading.

### 3.5.3 Direct shading

To model direct shading, we assume that spatially close surface points that share the same orientation should have similar shading. We thus formulate the corresponding direct shading term

$$E_d(s^\log_d) = \lambda_d^l \int \sum_{q \in N_S(r)} \|s^\log_d(r) - s^\log_d(q)\|^2 dr + ...$$

as the combination of local and non-local direct shading priors. Similar to the energy for simple shading $s$ described in Equation (3.23) by $N_S(r)$ we mean $k_S$-nearest neighbors of $r$ in 6D space, which here live in the 6D space of spatial locations $P$ and normals $n$ corresponding to the individual rays. $N^M_S(r)$ denotes the subset of $k_S$ among $M$-nearest neighbors of $r$ in 6D space that are randomly sampled.

For the global direct shading consistency, we also weight the similarity between two rays $r$ and $q$ with the angular difference of the outer normals and spatial locations,

$$w_{r,q}^{gl} = (1 + \gamma \max(\beta(r), \beta(q))) w_{r,q}^n w_{r,q}^d.$$  \hspace{1cm} (3.35)

Since direct shading should be free of cast shadows, we add additional weight for neighbors with high cast shadow score $\beta$.

### 3.5.4 Indirect shading

We model indirect shading by means of the shadow confidence measure $\beta$, whose detailed computation is described in Section 3.5.2. The main modeling assumptions are:

A1. Shading is spatially smooth except on shadow boundaries; that is, near discontinuities of $\beta$.

A2. Assume two points are spatially close to each other, none of them specular, they share the same orientation and their chromaticity is similar. Then $\theta_{r,q} = 1$ in (3.25), and their shadow free representation should be the same.

A3. The distribution of indirect shading is sparse, except inside the areas within cast shadows.

These assumptions lead to the indirect shading term

$$E_i(S^\log_i) = \lambda_i^l \int \sum_{q \in N_D(r)} \delta_{r,q} \|S^\log_i(r) - S^\log_i(q)\|^2 dr + ...$$

as

$$\cdot \|S^\log_i(r) - S^\log_i(q)\|^2 dr + \lambda_i^s \int (1 - \beta(r)) \|S^\log_i(r)\|^2 dr.$$  \hspace{1cm} (3.36)

To compute weight $\delta_{r,q} := (1 - \max(\delta_{\beta}(r), \delta_{\beta}(q)))$ we use $\delta_{\beta}$, shadow boundary (3.33) that is the norm of the Gaussian filtered gradient of $\beta$. The weight $\delta_{r,q}$ is small if and only if any of the two rays is close to a shadow boundary. Thus, we encourage discontinuity of indirect shading component across shadow boundary.
Figure 3.8: Epipolar images of disparity and intrinsic components: albedo shading and specularity, for the synthetic light field generated with Blender. The dashed line in the center view image shows the line alone which we illustrate the EPIs. We can observe the difference in the slope of epipolar lines for diffuse components, albedo and shading, and specular component. Slope of albedo and shading epipolar lines mostly coincide with the slope of the disparity lines, while specular epipolar lines are less slanted.

3.6 Ray Space Regularization and Optimization

We summarize the previously defined terms in the variational energies (3.6), (3.9) as a functional $F$. Consequently, to obtain the light field decomposition, we have to solve

$$\arg\min_{(A^{\log},S^{\log},H^{\log})} \left\{ F(A^{\log},S^{\log},H^{\log}) + J(A^{\log},S^{\log}) \right\}.$$  

(3.37)

Here, by $S^{\log}$ we mean shading component that can take form $S^{\log} = s^{\log}$ in case of model 1 described in Section 3.2.1 or $S^{\log} = s^{\log}_d + S^{\log}_i$ in case of model 2 presented in Section 3.2.2. Because all components $s, s_d$ and $S_i$ hold Lambertian property, which states that they are independent of viewing direction, we optimize them in the same way. As is typical in intrinsic image decomposition, the overall optimization problem is rather complex. However, taking a detailed look at the individual terms, it turns out that we have a convex objective $F$. Furthermore, our intention is to define the global smoothness term $J$ on ray space in a way that enforces spatial smoothness within the views and consistency with the disparity-induced structure on the epipolar plane images. Thus, the complete objective function exactly fits the light field optimization framework for inverse problems on ray space proposed by Goldluecke and Wanner [41]. The key advantage of this framework is that it is computationally efficient because it allows us to solve subproblems for each epipolar plane image and view independently. It is also generic in the sense that we just need to provide a way to compute $F$ and related proximity operators. We thus adopt their method to solve our problem.

In [41], the light field regularizer $J$ in (3.6) and (3.9) is a sum of several contributions

$$J_{\lambda,\mu} = \mu J_{xs} + \mu J_{yt} + \lambda J_{st}$$  

(3.38)

First, there are individual regularizers $J_{xs}$ and $J_{yt}$ for each epipolar plane image, which depend on the disparity map $d$ and employ an anisotropic total variation to enforce consistency of the fields in the arguments with the linear patterns on the epipolar plane image; see Fig. 3.8.

Let $p$ be a point of an EPI, then $d(p)$ is a disparity vector field that denotes in which direction the projection of the corresponding scene point moves if we change the view point. Let $U$ be a Lambertian intrinsic component, albedo or shading. The component $U$ should then be constant in the direction of $d$, except at the disparity discontinuities. For the fixed epipolar plane image with coordinates $(y^*, t^*)$, the
Chapter 3 Variational Models for Intrinsic Light Field Decomposition

regularization term

\[ J_{yt} = \int_{\Omega} \sqrt{\nabla U^T D_{\nu} \nabla U} \, dx \quad (3.39) \]

enforces the smoothness of \( U \) alone the disparity direction by means of tensor

\[ D_{\nu} = \nu(I - dd^T) + \frac{3 - \nu}{2} dd^T. \quad (3.40) \]

The constant \( \nu \) defines the degree of anisotropy. With small values, it encourages smoothing in the direction of the disparity field \( d \). Similar to expression (3.39) we define the regularization term \( J_{xs} \). For more information about anisotropic total variation we refer to the work by Kolev et al. [74].

The specular component cannot be smoothed along the disparity direction because its movement depends on local curvature of the surface and should be modeled separately. Thus, we optimize the specularity for each view independently according to the data term \( F \).

Second, for each view, there is a regularizer \( J_{st} \), and as in the basic framework in [41], we use a simple total variation term for efficiency. In future work, we intend to move to something more sophisticated here.

Albedo and shading are independent of view point, thus their values should not vary between views. We want \( A^{\log} \) and \( S^{\log} \) to be constant in the direction of \( d \), except at disparity discontinuities. We also regularize both components within each individual view as noted previously. The complete regularizer can thus be written as

\[ J(A^{\log}, S^{\log}) = \mu J_{xs}(A^{\log}, S^{\log}) + \mu J_{yt}(A^{\log}, S^{\log}) + \lambda J_{st}(A^{\log}, S^{\log}), \quad (3.41) \]

where \( \lambda, \mu > 0 \) are user-defined constants which correspond to the amount of smoothing on the separate views and EPIs, respectively. The objective is convex, so that we achieve global optimality. For details and the actual optimization algorithm, we refer the reader to [41].

The problem (3.37) is now solved in three steps:

1. Pre-compute all data necessary for albedo, shading, and specularity terms: specular and shadow masks that will be discussed in the following sections, spatial neighborhoods and chromaticity.

2. Initialize albedo \( A^{\log} \) and shading \( S^{\log} \) variables using values obtained from Color Retinex [46], and specularity \( H^{\log} \) with zeros.

3. Iteratively solve the problem within the framework proposed by Goldluecke and Wanner [41] for inverse problems on light fields:
   - Optimize \( F \) using gradient descent with respect to all intrinsic variables \( A^{\log}, S^{\log} \) and \( H^{\log} \) for every \((s, t) \in \Pi\).
   - Enforce angular consistency of \( A^{\log}, S^{\log} \) by optimizing \( J_{yt} \) and \( J_{xs} \) with subgradient descent.
   - Regularize \( A^{\log} \) with total variation (TV), and \( S^{\log} \) with total generalized variation (TGV) for every sub-aperture view \((s, t) \in \Pi\).

To speed up convergence, we recompute the step size of the gradient descent in each iteration such that the cost is minimized as quickly as possible. We then use it to adapt the remaining regularization steps.

All of the values for the parameters that we use in our work are given in Table 3.1.
Chapter 3 Variational Models for Intrinsic Light Field Decomposition

### 3.7 Results and Comparisons

We validate our decomposition methods on ground truth synthetic datasets and real world light fields. Our main goal is to achieve superior results for the center view of the light field, thus we present decomposition and evaluations only for the center view. Overall, our approaches make use of a light field with $9 \times 9$ views. Because estimated disparity is less accurate for the edge views, we solve the final optimization problems (3.6) and (3.9) for a cross-hair shaped subset of 13 views from a light field with $7 \times 7$ views, where edge views are excluded.

We present three synthetic datasets (Fig. 3.9, Fig. 3.11, and Fig. 3.10) and four real world light fields (Fig. 3.15, Fig. 3.12, Fig. 3.13, and Fig. 3.14). Here, we will briefly recap the main aspects of the two models that are being evaluated and compared with the other methods. We also mention some design choices used in the implementations of both models.

- **Model 1, Section 3.2.1**
  - Decomposition into three components: albedo, shading, and specularity.
  - Simple shading is used, Section 3.5.1.
  - Specular mask is estimated with Steps 3 and 4 only, without frequency analysis, Section 3.3.2.
  - Illumination color is estimated from the center view with the algorithm by Yang et al. [140]. The light field is then transformed to one that is illuminated with a white light source.
  - The resulting shading and specularity components are multiplied by the estimated illumination color.

- **Model 2, Section 3.2.2**
  - Decomposition into four components: albedo, direct shading, indirect shading and specularity.
  - Direct shading can be understood as the shading that the object would have had if it is alone in the scene, Section 3.5.3.
  - Indirect shading describes cast shadows and inter-reflections, Section 3.5.4.
  - A specular mask is computed with the full procedure, Section 3.3.2.
  - Calculations are performed under assumption of a white light source.

<table>
<thead>
<tr>
<th>Energy</th>
<th>$\lambda_{spec} = 0.5$</th>
<th>$\lambda_{sparse} = 0.5$</th>
<th>Specular Mask</th>
<th>$k_{spec} = 60$</th>
<th>$\gamma_d = 10$</th>
<th>$\gamma_d = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_{spec}$</td>
<td>$\lambda_{d} = 1$</td>
<td>$\tau_f = 0.1$</td>
<td>$\tau_f = 0.1$</td>
<td>$\gamma_d = 10$</td>
<td>$\gamma_d = 10$</td>
<td>$\gamma_d = 10$</td>
</tr>
<tr>
<td>$\lambda_{sparse}$</td>
<td>$\lambda_{d} = 1$</td>
<td>$\gamma_a = 10$</td>
<td>$\gamma_a = 10$</td>
<td>$\gamma_d = 10$</td>
<td>$\gamma_d = 10$</td>
<td>$\gamma_d = 10$</td>
</tr>
<tr>
<td>$\lambda_{retinex}$</td>
<td>$\lambda_{d} = 1$</td>
<td>$\gamma_a = 10$</td>
<td>$\gamma_a = 10$</td>
<td>$\gamma_d = 10$</td>
<td>$\gamma_d = 10$</td>
<td>$\gamma_d = 10$</td>
</tr>
<tr>
<td>$\lambda_{chroma}$</td>
<td>$\lambda_{d} = 1$</td>
<td>$\gamma_a = 10$</td>
<td>$\gamma_a = 10$</td>
<td>$\gamma_d = 10$</td>
<td>$\gamma_d = 10$</td>
<td>$\gamma_d = 10$</td>
</tr>
<tr>
<td>$\lambda_{normal}$</td>
<td>$\lambda_{d} = 1$</td>
<td>$\gamma_a = 10$</td>
<td>$\gamma_a = 10$</td>
<td>$\gamma_d = 10$</td>
<td>$\gamma_d = 10$</td>
<td>$\gamma_d = 10$</td>
</tr>
</tbody>
</table>

Table 3.1: Main parameters of the variational models 3.2.1 and 3.2.2 that we use in the implementation
Chapter 3 Variational Models for Intrinsic Light Field Decomposition

Ground truth

<table>
<thead>
<tr>
<th>Model 2 sec: 3.2.2</th>
<th>Model 1 sec: 3.2.1</th>
<th>Garces et al. [39]</th>
<th>Chen/Koltun [25]</th>
<th>Jeon et al. [64]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Albedo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LMSE=0.0021</td>
<td>LMSE=0.0074</td>
<td>LMSE=0.0076</td>
<td>LMSE=0.0065</td>
<td>LMSE=0.0198</td>
</tr>
<tr>
<td>GMSE=0.0025</td>
<td>GMSE=0.0118</td>
<td>GMSE=0.0154</td>
<td>GMSE=0.0141</td>
<td>GMSE=0.0272</td>
</tr>
<tr>
<td>Shading</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LMSE=0.0064</td>
<td>LMSE=0.0116</td>
<td>LMSE=0.0192</td>
<td>LMSE=0.0152</td>
<td>LMSE=0.0290</td>
</tr>
<tr>
<td>GMSE=0.0110</td>
<td>GMSE=0.0262</td>
<td>GMSE=0.0846</td>
<td>GMSE=0.0325</td>
<td>GMSE=0.0941</td>
</tr>
<tr>
<td>Specularity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LMSE=0.0006</td>
<td>LMSE=0.0015</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMSE=0.0014</td>
<td>GMSE=0.0019</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>center view</td>
<td>estimated disparity</td>
<td>estimated normals</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.9: Quantitative comparison on synthetic data rendered with Blender. The scene illustrates how the algorithms can cope with hard shadows, where chromaticity might vary in the shaded area. The disparity range is from $-1.5$ to $1.5$, light field size is $9 \times 9 \times 512 \times 512$. We show the estimated disparity and normal maps as well as the center view of the input light field. Since estimation of the intrinsic components relies a lot on the geometry of a scene, good quality of disparity map is essential for the accurate intrinsic light field decomposition. We find that our proposed model 2 outperforms the other approaches in all measures. The simplified version, model 1, reaches competing quality of the results compared to the other approaches, but fails to recover strong cast shadows. In the visual comparison, we see that approach described in model 2 successfully removed shadow from albedo component.

Note that because there is currently no ground truth available for intrinsic light field decomposition, and benchmark datasets for intrinsic image decomposition presented in [15, 21, 46] are not applicable to our setting, we have created our own ground truth data. Synthetic datasets were generated with Blender using the Cycles rendering engine and a light field plugin [60]. Internally, Blender combines direct shading $D$, indirect shading $I$ and object color $C$ for diffuse and glossy reflection separately by evaluating $(D_{\text{diffuse}} + I_{\text{diffuse}})C_{\text{diffuse}} + (D_{\text{glossy}} + I_{\text{glossy}})C_{\text{glossy}}$. The separate components can be stored individually to use as ground truth for evaluation. Albedo corresponds to the diffuse color, shading is the sum $D_{\text{diffuse}} + I_{\text{diffuse}}$ of direct and indirect illumination components, and specularity is the difference $H = L - AS$ between input light field and product of albedo and shading.

Our definition of direct and indirect shading differs from that used in Blender; thus, for quantitative evaluations, we compute resulting shading $S = s_dS_d$ and then compare to the ground truth shading $S_{\text{ground truth}} = D_{\text{diffuse}} + I_{\text{diffuse}}$.

We compare our methods to the two RGB+D approaches proposed by Chen and Koltun [25] and Jeon et al. [64], as well as recent method for light fields proposed by Garces et al. [39]. For quantitative evaluations, we selected two error measures. The first is local mean-squared error (LMSE) [46] computed patch-wise. The idea is to reduce scaling ambiguity by adjusting the brightness of the image patch, such that it corresponds to that of the ground truth, and we then compute MSE. In our experiments, we use rectangular overlapping patches with a size of 20% of the total image size. Although the LMSE is a more reasonable measure than pure MSE, it can sometimes produce non-sensible results in shaded areas.
### Chapter 3 Variational Models for Intrinsic Light Field Decomposition

<table>
<thead>
<tr>
<th>Ground truth</th>
<th>Model 2 sec: 3.2.2</th>
<th>Model 1 sec: 3.2.1</th>
<th>Garces et al. [39]</th>
<th>Chen/Koltun [25]</th>
<th>Jeon et al. [64]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Albedo</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LMSE=0.0032</td>
<td>LMSE=0.0038</td>
<td>LMSE=0.0041</td>
<td>LMSE=0.0033</td>
<td>LMSE=0.0053</td>
</tr>
<tr>
<td></td>
<td>GMSE=0.0076</td>
<td>GMSE=0.0082</td>
<td>GMSE=0.0082</td>
<td>GMSE=0.0078</td>
<td>GMSE=0.0085</td>
</tr>
<tr>
<td><strong>Shading</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LMSE=0.0213</td>
<td>LMSE=0.0231</td>
<td>LMSE=0.0208</td>
<td>LMSE=0.0262</td>
<td>LMSE=0.0377</td>
</tr>
<tr>
<td></td>
<td>GMSE=0.0607</td>
<td>GMSE=0.0624</td>
<td>GMSE=0.0636</td>
<td>GMSE=0.0682</td>
<td>GMSE=0.0975</td>
</tr>
<tr>
<td><strong>Specularity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>center view</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>estimated disparity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>estimated normals</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.10: Synthetic dataset generated with Blender using Cycles rendering engine. Disparity range is \([-1.5, 1.3]\). The following two datasets have identical scene geometry. However, the first scene contains an object with pure Lambertian reflection, the second an object with specular reflection. Here, we present the Lambertian reflection. With this example, we want to evaluate how algorithms perform on a scene without specularity. For our algorithms, we assign zeros to the specularity confidence mask \(h\), so that they in effect run without specularity component detection. We observe that in Lambertian case, the proposed methods still have some advantages; for instance, cast shadows are successfully removed with the model 2. Results obtained with model 1 have smooth shading without too much texture left.

Thus, we also compute global mean-squared error (GMSE) that adjusts the test image brightness value to the ground truth for the whole image. With GMSE, if shaded regions are present in the albedo image, then the influence of their error will be more reasonably reflected in the error measure—MSE would otherwise be very small for dark regions. Because aggregated quantitative evaluation can only give a partial picture of the performance, we also visualize all decomposition results for qualitative evaluations. The resulting direct and indirect shading described in Section 3.5, as well as specular, shadow and its boundary masks for the above datasets can be observed in Figs. 3.16 and 3.17. To assess individual contributions of proposed priors, we present results where some prior are excluded by setting corresponding weights to zero, see Fig. 3.18.
Figure 3.11: Geometrically, the same as the previous example, see Fig. 3.10, but the object now has not only complicated geometry but also specular reflection. Again, we perform much better on removing the cast shadow. Both proposed approaches are able to find specularity.

Figure 3.12: Qualitative comparisons on the real world light field from figure 3.2 captured with Lytro Illum plenoptic camera. This scene has highly specular saxophone and almost Lambertian koala toy. Background and koala have similar pallets, which creates difficulties in albedo estimation. Our advanced method successfully removes cast shadows, especially on the wall behind the koala and from saxophone. In both approaches the specular component contains all highlights that appear on saxophone.
Figure 3.13: Qualitative comparisons of the real world dataset captured with Lytro Illum plenoptic camera, 9×9 views, resolution 434×625. Method based on model 2 outperforms the previous approaches, especially albedo component has much less cast shadows, and shading is more smooth and contains less texture. Method by Garces et al. [39] outputs almost perfect albedo, but shading contains too much structure, compared to our models and work by Chen and Koltun [25], where shadows are still present in albedo, but shading image contains little discontinuities caused by texture.

Figure 3.14: We present one more scene with koala and saxophone. This scene contains more shadows and less specularity. Also, cast shadows have stronger boundaries. We conclude that both models performs good on this dataset, compared to the other methods. In both cases shadows are completely removed from albedo component. The difference in color of the specularity comes from the fact that model 2 was solved under the assumption of white light source. In case of model 1 we first estimated light source color with [140] and then transformed input light field into one that is illuminated with white light source.
Chapter 3 Variational Models for Intrinsic Light Field Decomposition

Chapter 3 Variational Models for Intrinsic Light Field Decomposition

### Model 2

[Image]

### Model 1

[Image]

### Albedo

[Image]

### Shading

[Image]

### Specularity

[Image]

---

**Figure 3.15:** Real world dataset captured with industrial camera mounted on gantry. The size of the light field is $9 \times 9 \times 497 \times 710 \times 3$, with the disparity range $[-1.5, 1.0]$. We present a dataset with non-Lambertian object, illuminated with approximately white light. This scene was captured with the industrial camera mounted on a gantry. We observe that the method based on model 1 could detect more specularity compared to the method based on model 2. In this case additional frequency analysis removed small regions from the specular mask, thus the algorithm could not assign values for the specular components in those regions. For the specular and shadow mask illustration, we refer to Fig. 3.17.

---

**Figure 3.16:** The first column shows the center views of three synthetic light fields introduced in Figs. 3.9, 3.10 and 3.11 respectively. The second and third columns illustrate direct and indirect shading components. The fourth column shows a final shadow score, where brighter regions correspond to the higher chance of having a cast shadow in that area. The fifth column represents a shadow boundary. The last column illustrates specular masks. We observe that direct and indirect shading are well-separated and almost all cast shadows are identified. Also, we observe that intense specular regions are present in the specular mask.

---

center view direct shading indirect shading shadow score shadow boundary specular mask

---

###sec: 3.2.2

###sec: 3.2.1

###Garces et al.

[39]

###Chen/Koltun

[25]

###Jeon et al.

[64]
Figure 3.17: The components are the same as in Fig. 3.16 but are computed for the real-world light fields. Note, that for this type of light fields disparity estimation is extremely challenging task due to complicated lightning condition, geometry and camera noise. Although our shading components rely on the estimated disparity, we conclude that the proposed methods cope well with the real scenes. We observe that cast shadows and inter-reflections are correctly recovered. Direct shading is shadow-free as designed and specular mask captures all strong specularity.
Figure 3.18: The left-hand column shows the original result of the second model with experimentally determined parameter settings as specified in table 3.1. The other columns show the consequences of excluding certain priors from the model. We can see that local direct and indirect shading priors \( \lambda^d \) and \( \lambda^i \), respectively, influence the smoothness of the shading component, while the global indirect shading prior \( \lambda^g \) controls the amount of cast shadows left in the albedo component. The specularity prior \( \lambda_s \) is required to prevent over-smoothing of albedo and shading.
3.8 Conclusions

In this chapter, we proposed the first approach towards solving the intrinsic 4D light field decomposition problem, while leveraging the disparity-induced structure on the epipolar plane images.

In contrast to existing intrinsic image algorithms, the dense collection of views in a light field allows us to define an additional specular term in the decomposition model, so that we can optimize over the specular component, and also albedo and shading by minimizing a single variational functional.

Another main contribution is a novel shading term that describes cast shadows and inter-reflections. By means of this term, we recover consistent shading components in both hard and soft shadows.

Moreover, we improve albedo and specularity estimation by embedding specularity information in the albedo prior, which makes the albedo less affected by highlights. Because the inverse decomposition problem is embedded in a recent framework for light field labeling [41], we can ensure that albedo and shading estimates are consistent and use information from all views.

Our experiments demonstrate that we outperform both a state-of-the-art intrinsic image decomposition method employing additional depth information [25], [64] and a light field based method for specular removal [120], [39] on challenging non-Lambertian scenes.

We show that the advanced model described in Section 3.2.2, which explicitly models cast shadows and inter-reflections, outperforms the first model with simple assumptions for the shading component described in Section 3.2.1. We introduce ground truth evaluations on synthetic light fields, where we qualitatively and quantitatively show that our method significantly outperforms existing algorithms. In addition, we perform qualitative evaluations for the real world examples captured with a Lytro Illum plenoptic camera and gantry.
Deep Autoencoder for Light Fields

We present a fully convolutional autoencoder for light fields, which jointly encodes stacks of horizontal and vertical epipolar plane images through a deep network of residual layers. The complex structure of the light field is thus reduced to a comparatively low-dimensional representation, which can be decoded in a variety of ways. The different pathways of upconvolution that we currently support are used for disparity estimation and separation of the lightfield into intrinsic components. The architecture allows to decompose a light field into diffuse and specular components or perform full intrinsic light field decomposition into albedo, shading and specularity. The key idea is that we can jointly perform unsupervised training for the autoencoder path of the network and supervised training for the other decoders. Consequently, we are able to find features that are both tailored to the respective tasks and generalize well to datasets for which only example light fields are available. Another use case of the proposed architecture is to recover a high-resolution light field from its low-resolution version in angular and spatial domains. We provide an extensive evaluation of the synthetic light field data, and show that the networks yield good results on previously unseen real world data captured by a Lytro Illum camera and various gantries.

4.1 Related Work

Here, we list the relevant machine learning papers for light field analysis. We also mention the recent works on the following topics: light field encoding, reflection separation, intrinsic images, disparity estimation, and super-resolution tasks. Although we concentrate on deep learning approaches, we also introduce some state-of-the-art methods. For each research direction, we selected several publications with the ideas that support the design choices we made for the proposed architecture. Given that the literature review on the conventional algorithms for intrinsic images is covered in the previous chapter, we only mention deep learning methods here.

4.1.1 Neural networks for light field analysis

Deep neural networks are employed for all of the above tasks including light field analysis. Wang et al. [129] aim at material classification. They explore different light field representations that can be used to train a convolutional neural network. Meanwhile, Heber and Pock [54, 56] apply an encoder-decoder architecture on 2D EPIs and later 3D EPI stacks to estimate depth. Kalantari et al. [69] and Srinivasan et al. [108] introduce view synthesis algorithms, which recover light fields from a sparse set of images or a single view. Similarly, [48] obtain compressive light field reconstructions from single coded 2D images using a joint autoencoder and 4D-CNN architecture. Recently, deep networks were also successfully applied for inverse rendering and intrinsic image problems [76, 103]. In contrast to these approaches, our architecture is not limited to a single task but can be trained to perform several of these jointly by implementing different decoder chains.
4.1.2 Encoding light fields

From the first introduction of light fields for image-based rendering [44, 81], light field compression has been an important topic due to the huge amount of data which needs to be stored. Early on, it has been noted that estimating disparity is necessary to exploit the redundancies in the different viewpoints [84]. This can be turned around, and sparse coding actually been used as a tool for disparity estimation—similar in spirit to what we are proposing here. In [53], the authors use the idea of redundancy of sub-aperture views and used sparsity of the RPCA as a new matching term. Likewise, [89] employ sparsity ideas to model light field patches as Gaussian random variables conditioned on its disparity value. They construct a patch prior and can estimate disparity by finding the nearest PCA subspace. In [67], EPI patches are encoded with a dictionary of patches with known slope, such that the coding coefficients give a disparity estimate. Notably, this method can recover disparity for multiple layers of a scene. Sparse coding is also used for compressive light field photography [86], which reduces the amount of data to be captured. Both sparse coding and low-rank constraints are also key to modern light field compression schemes [24, 65].

However, the idea of the autoencoder that we employ in this work is in some sense the exact opposite to sparse coding: instead of finding an overcomplete basis and represent patches with a sparse vector in a high-dimensional space, we want to directly find the best low-dimensional coding.

4.1.3 Disparity estimation

The problem of disparity estimation benefits considerably from the multiple views that are available in a light field; for an overview of the various algorithms, we refer the reader to the recent work of Johannsen et al. [66]. Deep learning approaches lead to significant improvement in this task. Heber et al. [56] recover a 4D depth field from the light field using CNN followed by convex optimization to refine point-wise predictions from the deep network. Srinivasan et al. [108] synthesize a 4D light field from a single image with two neural networks: the first estimates the disparity and renders Lambertian light field, and the second predicts occluding rays and models non-Lambertian effects. In [7], we jointly solve disparity estimation and reflection separation tasks with a fully-convolutional encoder-decoder network.

4.1.4 Reflection separation

The dichromatic reflection model proposed by Shafer [101] decomposes an input scene into diffuse and specular components. Based on this, [141] considers specularity removal as an image denoising problem and solves it with bilateral filtering. In [114, 116], Tan and Ikeuchi devise a method based on pure chromaticity analysis without any geometrical information. Kim et al. [72] used the fact that the dark channel can provide an approximately specular free image. In [3], Akashi and Okatani use sparse non-negative matrix factorization to jointly estimate body color and separate reflection components.

What makes reflection separation from a single image particularly difficult is that specularity is a view dependent phenomenon and can hardly be recognized from a single view point. With multiple views available, changes in object appearance can be tracked with respect to the viewing angle, which significantly simplifies the task of reflection separation. The behavior of specularity in static scenes with a moving camera is described by Swaminathan et al. [112] These authors show how motion of specularity depends on object geometry and light source position, and propose a technique for specularity extraction from an image sequence. Another work by Lin et al. [83] uses color analysis and multibaseline stereo to recover both depth and reflection components.

Recent works by Gryaditskaya et al. [47] and Sulc et al. [110] explore the light field structure to edit appearance of specularity and estimate diffuse and specular components. Tao et al. [120] adapt the
dichromatic reflection model to light fields and propose a depth estimation and specularity removal algorithm. Finally, Criminisi [30] studies the behavior of diffuse and specular components in EPIs and proposes several reflection separation techniques.

4.1.5 Intrinsic images

For an overview of intrinsic decomposition algorithms, we refer the reader to the work of Bonneel et al. [19], where the authors discuss priors used for modeling intrinsic components. Among the deep learning approaches, Narihira et al. [91] was the first to introduce CNN for recovering relative lightness that was trained on human judgments on relative reflectance [15]. They later developed a regression CNN-based model that predicts albedo and shading components. Shi et al. [103] introduced a mirror-like, U-shaped architecture that solves non-Lambertian intrinsic decomposition from a single image. Janner et al. [62] developed a self-supervised (RIN) model, which predicts reflectance, shape, and lighting conditions given a single image.

4.1.6 CNN-based single image super-resolution

Methods based on convolutional neural networks are widely employed for inverse problems, such as deblurring [111, 138], image denoising [85, 100] and image super resolution [49, 50, 61, 85, 105]. Mao et al. [85] introduce a fully convolutional autoencoder architecture with skip connections to deal with single image restoration, including denoising and single image super-resolution. Skip connections are proven to be very useful to guide the decoding process and increase detail. In our approach, we consequently employ 3D skip connections between our encoder and decoder. Shi et al. [105] propose a sub-pixel convolutional layer that maps low-resolution (LR) image to high-resolution (HR) output. Their research shows that strided upconvolution can produce checkerboard patterns in the generated high-resolution images, which we avoid by first upscaling the features spatially by the means of bicubic interpolation, thus removing the need to apply strides. Ledig et al. [78] employ a generative adversarial network to force the network produce more natural high-resolution images. As a modification, we introduce our DiffGAN, which not only takes the pixel value of the generated images and the ground truth but also the derivatives of them. Consequently, DiffGAN helps to enhance the details of the generated images. Some other network architectures employ unique units to enhance or refine the HR output, such as the backprojection unit [50] and the information distillation units [61]. Instead of a straightforward deep CNN architecture, Han et al. [49] attempt to use a recurrent neural network for single image super-resolution.

4.1.7 Light field super-resolution

Due to the need to sacrifice resolution to sample angular coordinates, the light fields usually suffer from the lower spatial resolution compared to standard images. Bishop et al. [16] introduce a variational Bayesian framework for light field super-resolution, closely related to classical approaches [9]. Shi et al. [104] present the light field signal reconstruction in the frequency domain. Among recent studies, Wanner and Goldluecke [133] and Pujades et al. [96] propose a variational super-resolution framework using estimated high-accuracy depth maps from epipolar plane images and solve novel view synthesis as an inverse problem. Likewise, Mitra and Veeraraghavan [89] make use of depth information of the scene to construct an inference model with a Gaussian mixture model prior. Rossi and Frossard [99] adopt a multi-frame approach with a graph-based regularizer.

With the continuing success of deep learning, CNN-based light field super-resolution methods have become common. Yoon et al. [143] train three networks of the same architecture for spatial resolution
and finally combine them to achieve angular super-resolution. Farrugia and Guillemot [33] upscale the whole light field spatially by utilizing a low-rank approximation restored by a CNN.

In this work we use an epipolar volume convolutional autoencoder to process light fields. The autoencoder, unlike the straightforward neural network architectures, shrinks the size of the input data in the encoding process, and thus leaves more space for the number of features and batches, especially for high-dimensional training data such as the light fields.
In general, the architecture that we propose the pathways of our deep encoder-decoder network are organized in six groups of three residual blocks each. The first two blocks in each encoder group keep depth and resolution the same, the last block reduces resolution, while increasing feature depth. Depending on the task and GPU memory available, number and content of groups can be changed. The decoder paths are exact mirrors of this chain. The viewpoint dimension of the shape in 2D decoders is removed. The network supports any number of 2D and 3D decoders. If the decoders are closely related they can share the features on a certain layers. To not overly clutter the figure, the visualization does not show that the encoder and 3D decoders actually operate on two EPI stacks in parallel, the horizontal and vertical one. The feature output of these is briefly joined on the bottom layer, and then decoded again into two separate chains.
4.2 General Architecture

The key idea is to build the network around an autoencoder, so it can be trained without supervision using just raw light fields. However, we add multiple pathways to decode the latent representation, which can be trained jointly with the autoencoder in a supervised manner, depending on which data is available in the current training example. Due to the combination of supervised and unsupervised training, we ensure that the latent representation is both tailored to the desired tasks, such as depth reconstruction or intrinsic component representation, but can also generalize well to datasets for which no training information is available for these tasks. When the network is deployed, all of the decoder chains can be evaluated using just the light field data.

4.2.1 Encoder pathway

The input to the network is a pair of epipolar volumes, one sliced horizontally and the other sliced vertically, see Fig. 4.2. Input patches are $48 \times 48$ RGB with a depth of nine views. Larger light fields are segmented into these patches, so that our network can deal with lightfields of any shape. The basic ingredient for the encoders and decoders are residual blocks. To decrease resolution, we employ strided convolutions instead of max-pooling, so the network is fully convolutional. See [52, 107] for justifications of this architecture. The residual blocks have a very simple structure and allow direct pass-through of the (batch normalized) input, see figure4.3. In the encoder pathway, the residual blocks are chained together. Some of them reduce the patch resolution via strided convolution while increasing feature depth, with the overall goal of gradually reducing dimensionality. Another just applies convolution with stride one, which keeps the resolution but updates the features. The final output has shape $3 \times 3 \times 3 \times N$, where $N$ is the number of features in the bottleneck; see Fig. 4.1. Horizontal and vertical epipolar volumes are encoded separately. Because they have the exact same structure, we have them share the same filter kernels to reduce the number of network parameters. Given that pathways like depth reconstruction require information from both horizontal and vertical epipolar volumes, their feature output at the representation level is concatenated. This is the final output of the encoder and it is also the bottleneck of the network.

4.2.2 Decoder pathways and output

After passing the bottleneck, the low-dimensional representation is decoded again by a chain of residual layers. The latent variables enter different decoder pathways. All decoder pathways use transpose convolutions to exactly revert the encoder on the corresponding level. However, the only link between them is through the latent representation, see Fig. 4.1. Another possibility to link the encoder and decoder is a skip connection, which can be enabled for the decoder if necessary.

Various components can be reconstructed for the $17 = 9 + 9 - 1$ views in a crosshair around the center view, see Fig. 4.2. Later, we show the application of the architecture to the following tasks: disparity estimation and reflection separation, intrinsic light fields with a separate decoder for the disparity, super-resolution with magnification factors $x2$ and $x4$. The disparity map is computed for the center view only.

To incorporate physical models of image formation we employ the dichromatic reflection model [101], whose adaptation to light fields was discussed in detail in the previous chapter and [120]. According to this model, the specular component is assumed to be independent from the diffuse component, which justifies the use of two separate decoder chains. The diffuse component can be further decomposed into albedo and shading. However, all of the intrinsic components should sum up to the input light field. To let the network better cope with this constraint, we share features of the corresponding decoders on certain layers to ensure that the network learns the dependency between decoders. Because the disparity
Chapter 4 Deep Autoencoder for Light Fields

Figure 4.2: Visualization of horizontal (left) and vertical (right) EPI stacks used as input to our network. To achieve the actual spatial input resolution of $48 \times 48$, they need to be cut out from the above epipolar volumes. Note that although both stacks are three dimensional, they use images along different directions of view points. In effect, those two stacks assemble a crosshair of views around the center view, which is thus the only view present in both stacks.

Figure 4.3: Single residual block of the encoder and decoder networks. After batch normalization, a first path leads through a (possibly strided) convolution or up-sampling layer and a leaky ReLU. A second path either keeps the input, or passes it through a strided (transposed) convolution in case it needs to be resampled. Both paths are added together to produce the final output. In the decoder block with the skip connection option, the output is concatenated with the corresponding encoder features. It then passes through the $1 \times 1$ convolution with stride one to bring number of features to the original size. The idea is that it is much easier for such blocks to learn the identity transformation, or perform only small modifications to the input \cite{52}, which helps the encoder-decoder paths to gradually add details.

output is only 2D, we reduce the filter shape by the respective dimension. When tiling the output back together, we use overlapping patches and extract only its central part because data closer to the center is more accurate.

4.3 Loss Function

In the classical supervised machine learning, we need training data together with a label or target to train the model. By observing the random variable $X$ and its label $Y$ in the training set, supervised learning tries to fit the model $f_\theta$ to the training samples $(x_0, y_0), \ldots, (x_n, y_n)$. The test data has similar structure as training data, but the network never sees it. The training process can be considered as learning the mapping $f_\theta$ to predict label $y^*$ from the new instance $(x^*, y^*)$ of the test data via

$$f_\theta(x^*) = y^*$$  \hspace{1cm} (4.1)
Here, $\theta$ denotes set of trainable parameters of the model that are learned to minimize the difference between estimated function values and targets. A typical cost function for image restoration problems is the mean squared error (MSE)

$$E_{MSE}(X, Y, \theta) = \frac{1}{N} \sum_{n=1}^{N} ||f_{\theta}(x_n) - y_n||^2_2$$

(4.2)

This MSE cost is the so-called $L^2$ loss. Therefore, the goal is to minimize the loss to train the model

$$\hat{\theta} = \operatorname{argmin}_{\theta} E(X, Y, \theta)$$

(4.3)

To predict the correct $y$ from new instance of the test data, the model $f_{\theta}$ must have the ability to generalize. However, there are only finite number of samples in the training data and an infinite number of test pairs that could be completely different; that is, one only rarely observes the precise relationship $y = f_{\theta}(x)$. Instead, there is also a measurement error. Under such conditions, we assume that the training and test pairs are independently sampled from the same joint distribution $Pr(x, y)$, and the measurements follow a normal distribution over the correct values

$$Pr(y|x, \theta) = \text{Norm}[f_{\theta}(x), \sigma^2]$$

(4.4)

Thus, our goal can be formulated as maximizing the probability to see the outputs that we sampled from the training data given the inputs from the training data

$$\hat{\theta} = \operatorname{argmax}_{\theta} P(y|x, \theta) = \operatorname{argmax}_{\theta} \prod_{n=1}^{N} P(y_n|x_n, \theta)$$

$$= \operatorname{argmax}_{\theta} \left\{ \prod_{n=1}^{N} \text{Norm}_{y_n}[f_{\theta}(x_n), \sigma^2] \right\}$$

$$= \operatorname{argmax}_{\theta} \left\{ \prod_{n=1}^{N} \frac{1}{2\pi\sigma^2} e^{\frac{- (f_{\theta}(x_n) - y_n)^2}{2\sigma^2}} \right\}$$

$$= \operatorname{argmin}_{\theta} \left\{ -\log \left( \sum_{n=1}^{N} \frac{1}{2\pi\sigma^2} e^{\frac{- (f_{\theta}(x_n) - y_n)^2}{2\sigma^2}} \right) \right\}$$

$$= \operatorname{argmin}_{\theta} \left\{ \sum_{n=1}^{N} \left( -\log \frac{1}{2\pi\sigma^2} + \frac{(f_{\theta}(x_n) - y_n)^2}{2\sigma^2} \right) \right\}$$

$$= \operatorname{argmin}_{\theta} \left\{ \frac{1}{2\sigma^2} \sum_{n=1}^{N} (f_{\theta}(x_n) - y_n)^2 \right\} \sim \operatorname{argmin}_{\theta} L^2(f_{\theta}(x), y)$$

(4.5)

This is called the maximum-likelihood estimation, which is proven to be the best asymptotic estimation in terms of convergence rate if the number of training samples goes to infinity. As the deep learning is a data-driven approach, with the increasing number of training samples, maximum-likelihood estimation seems to be the first choice; however, according to Equation 4.5, the maximum-likelihood estimation leads to almost the same result as the $L^2$ loss. So it is reasonable to use the $L^2$ loss for the image restoration tasks as the cost function.

Unlike supervised learning, the training data for unsupervised learning usually does not contain the label or target value because unsupervised learning tries to extract useful structure from the training data; for example, principal component analysis and clustering. In our work we use model driven cost function, where we ensure that recovered components satisfy underlying physical model of the image formation.
g, thus we minimize the energy of the form

$$E_{\text{model}}(X, \theta) = \frac{1}{N} \sum_{n=1}^{N} \|g(f_{\theta}(x_n)) - x_n\|_2^2$$

(4.6)

The final loss function is a weighted combination of supervised and unsupervised costs

$$E_{\text{CNN}} = E_{\text{MSE}}(X, Y, \theta) + \lambda E_{\text{model}}(X, \theta).$$

(4.7)

In the following section, we will introduce the key ingredients of CNN that are used in our architecture: convolution and its transposed version, activation function, skip connection, and Adam optimizer.

### 4.4 Building Blocks

In deep learning, a convolutional neural network is a class of feed-forward networks and is most commonly applied to computer vision tasks. CNN is inspired by biological processes because the pattern of connections between neurons resembles the organization of the animal’s visual cortex. Individual neurons only respond to stimuli in restricted areas of the receptive field. As a feed-forward network, the CNN is composed of multiple layers in which the main operations are convolutions. In addition, other operations work together with convolution for different purposes, which will be briefly introduced in this section. For an extensive description of CNN architectures and deep learning theory, we refer the reader to the book by Goodfellow et al. [42].

#### 4.4.1 Convolution and convolution transposed

From the perspective of image processing and CNN, the convolution is usually done in the spatial domain of the image. Let $K$ denote a $N \times M$ kernel and $I$ a grayscale image with resolution $H \times W$, we can imagine the convolution as sliding the kernel $K$ onto each pixel of the image $I$, calculating the element-wise multiplication of its $N \times M$ neighboring pixels and the corresponding element of the kernel, and finally adding them up as the value of that pixel. In mathematical terms, the 2D convolution can be expressed as follows

$$\left( I * K \right)_{(h,w)} = \sum_{n=1}^{N} \sum_{m=1}^{M} I_{(h-n,w-m)} K_{(n,m)}$$

(4.8)

The output of the convolution is called its feature. The values in the kernel are called weights and they are constant during the operation and are only updated after an iteration of gradient descent. This is called weight sharing and it only requires one set of parameters to be learned in a layer instead of a set of parameters for each position in an image.

A fundamental property of convolution is that it is a linear operation. If the $3 \times 3$ convolution kernel $K$
and the $4 \times 4$ image $I$ are row-first expanded into column vectors, then the convolution operation (4.8)

$$I \ast K = \begin{pmatrix}
K_{(0,0)} & 0 & 0 & 0 \\
K_{(0,1)} & K_{(0,0)} & 0 & 0 \\
K_{(0,2)} & K_{(0,1)} & 0 & 0 \\
0 & K_{(0,2)} & 0 & 0 \\
K_{(1,0)} & 0 & K_{(0,0)} & 0 \\
K_{(1,1)} & K_{(1,0)} & K_{(0,1)} & K_{(0,0)} \\
K_{(1,2)} & K_{(1,1)} & K_{(0,2)} & K_{(0,1)} \\
0 & K_{(1,2)} & 0 & K_{(0,2)} \\
K_{(2,0)} & 0 & K_{(1,0)} & 0 \\
K_{(2,1)} & K_{(2,0)} & K_{(1,1)} & K_{(1,0)} \\
K_{(2,2)} & K_{(2,1)} & K_{(1,2)} & K_{(1,1)} \\
0 & K_{(2,2)} & 0 & K_{(1,2)} \\
0 & 0 & K_{(2,0)} & 0 \\
0 & 0 & K_{(2,1)} & K_{(2,0)} \\
0 & 0 & K_{(2,2)} & K_{(2,1)} \\
0 & 0 & 0 & K_{(2,2)}
\end{pmatrix}
\begin{pmatrix}
I_{(0,0)} \\
I_{(0,1)} \\
I_{(0,2)} \\
I_{(0,3)} \\
I_{(1,0)} \\
I_{(1,1)} \\
I_{(1,2)} \\
I_{(1,3)} \\
I_{(2,0)} \\
I_{(2,1)} \\
I_{(2,2)} \\
I_{(2,3)} \\
I_{(3,0)} \\
I_{(3,1)} \\
I_{(3,2)} \\
I_{(3,3)}
\end{pmatrix} \tag{4.9}
$$

can be written out as matrix multiplication.

Note that we obtain the final image by reshaping the result of the matrix multiplication, which is a $4 \times 1$ vector.

The 3D convolution is an extension of the 2D convolution. The kernel can be imagined as a cube and it moves around in the input volume. The pixel value convolved with the kernel is not only computed with its spatial neighbors but is also computed within the temporal dimension of the kernel.

Equation 4.9 shows that it is valid to write the convolution operation as matrix multiplication. To better understand the convolution transpose, we give another simple example in 1D: let a $1 \times 4$ input convolve with a $1 \times 3$ kernel with stride 1 and 0-padding

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}^T \begin{bmatrix} a & b & c & d & 0 \end{bmatrix} = \begin{bmatrix} 0x + ay + bz \\ ax + by + cz \\ bx + cy + dz \end{bmatrix} = \begin{bmatrix} x & y & z & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \end{bmatrix} = K I \tag{4.10}
$$

If we transpose $K$ and multiply it with output, we obtain

$$K^T \begin{bmatrix} p \\ q \\ r \\ s \end{bmatrix} = \begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \end{bmatrix} \begin{bmatrix} p \\ q \\ r \\ s \end{bmatrix} = \begin{bmatrix} px \\ py + qx \\ pz + qy + rx \\ qx + ry + sz \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}^T \begin{bmatrix} p \\ q \\ r \\ s \\ 0 \end{bmatrix} \tag{4.11}
$$
Here, the multiplication of the transposed kernel matrix and the output of the original convolution can be written as another convolution, where the new kernel \([z, y, x]\) is the flipped \([x, y, z]\), which is why it is called convolution transpose.

For a 2D case, recall that in equation 4.9 we transform a \(3 \times 3\) convolutional kernel into a \(4 \times 16\) matrix for a \(4 \times 4\) input and finally obtain a \(2 \times 2\) output after reshaping. Now, if we expand the output to a \(4 \times 1\) vector and right-multiply with the transposed kernel matrix, we will have a \(16 \times 1\) vector result. By reshaping it, we upscale the result of the original convolution. Even though it is called the transposed convolution, this does not mean that we take some existing convolution matrix and use the transposed version; that is, the actual weight values in the transposed matrix do not have to come from the original kernel matrix. It is important that the weight layout is transposed from that of the kernel matrix.

4.4.2 Activation functions

Activation functions (Fig. 4.4) are one of the most important ingredients in the network. On the one hand, they filter out the non-critical information and only let the key information in the features flow into the next layers. This simulates the working principle of the neurons in a biological network, which is why they are called "activation functions". On the other hand, they provide non-linearity, which significantly improves the representational ability of the network, because convolution is a linear operation. There are several popular activation functions available for deep learning tasks.

Sigmoid

\[
\sigma(x) = \frac{1}{1 + \exp(-x)} \quad (4.12)
\]

Squeezes the input into the 0 to 1 interval that is consistent with the range of probabilities, which is why sigmoid is very popular in classification tasks; see Fig. 4.4 a. The shape of \(\tanh(x)\) is similar to sigmoid, except that \(\tanh\) projects the inputs into the interval \([-1, 1]\); see Fig. 4.4 b. These two activation functions have a common problem in that the activated values are very easy to become saturated due to their value range because once the input \(|x| \geq 10\), they remain constant after the activation and thus the gradients will be close to zero. Consequently, no change will happen during gradient descent—that is, there is no learning for the weight in the network—, which is called the gradient vanishing problem.

ReLU = \(\text{max}(0, x)\) is the most commonly used activation function because of its simplicity during back-propagation and it is computationally not expensive, see Fig. 4.4 c. ReLU handles the gradient vanishing problem that frequently occurs in sigmoid and tanh, it also converges more quickly than some other activation functions. By contrast, ReLU’s value range \([0, \infty)\) leads to the so-called "dead neuron" problem; that is, the value of the input that continues to be negative is always zero, thus the weights related to them are never updated.

Leaky ReLU = \(\text{max}(\alpha x, x)\) is an extension of ReLU, see Fig. 4.4 d. It has all properties of ReLU, plus it will never have the "dead neuron" problem. We can consider different multiplication factor \(\alpha\)
to form different variations of Leaky ReLU. Parametric leaky ReLU is a further extension of Leaky ReLU in which the multiplication factor is treated as a trainable parameter instead of being pre-defined in usage.

In addition to the activation functions introduced earlier, there are some other choices, such as: Max-out, Exponential Linear Unit (ELU), to name but a few. Most of these activation functions overcome the weaknesses of the previous functions but usually require more computational effort.

4.4.3 Batch normalization

It is common practice to split the training data into batches to proceed the training with stochastic gradient descent (SGD). However, the distribution of the values in a batch might be completely different from that of the entire dataset. As we mentioned in the previous subsection, the activation functions are sensitive to their input. Thus, if we do not change the value distribution in the batch and occasionally have an inappropriate initial weight, the unstable gradient or "dead neuron" problems will easily occur.

To deal with this situation, batch-normalization is applied before the activation function in each layer such that the activation function is fed with a batch of proper distribution and one does not have to worry about how to initialize the parameters.

Consider a batch $B = (x_1, ..., x_n)$ with $N$ sample points. Batch normalization proceeds in four steps:

\[ \mu_B = \frac{1}{N} \sum_{i=1}^{N} x_i \]  
\[ \sigma^2_B = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu_B)^2 \]  
\[ \hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma^2_B + \epsilon}} \]  
\[ y_i = \gamma \hat{x}_i + \beta = BN_{\gamma, \beta}(x_i) \]

First, it computes the mean $\mu_B$ and variance $\sigma^2_B$ over values of $x \in B$. It then additionally learns the scale and offset parameters $\gamma$ and $\beta$. Therefore, each layer is trained on normalized input which makes the training robust to "internal shift". Note that the batch normalization should be disabled during testing because true mean $\mu_{data}$ and variance $\sigma^2_{data}$ are desired.

4.4.4 Skip connections

Mao et al. [85] introduce a fully convolutional autoencoder architecture with skip connections to deal with single image restoration, including denoising and single image super-resolution. Skip connections are applied between encoder and decoder to form the residual learning paths. The network architecture is shown in Fig. 4.5.

First, convolution is used to extract features. After several convolution layers, the image features are obtained with the noise being filtered out, thereby achieving the purpose of noise reduction. Subsequently, the features extracted are used by deconvolution to perform image reconstruction. In this network, convolution extracts features, and deconvolution up-samples them, thus completing the transition from image to feature and then from feature to image. Second, skip connection is employed in an encoder-decoder network; that is, linking the convolutional layer in the encoder with the corresponding decoder layer via skip connection. This has been mentioned in [52] that as the network deepens, the problem of unstable
gradient will occur. Skip connection can be used to form a path for residual learning, thus effectively solving the gradient vanishing or exploding problem.

The authors also compare the performance between the models with and without the skip connection. It is shown that the unstable gradient problem rarely appears and has less influence on the loss when the number of layers is small. Once the network grows deeper, the performance without skip connection is poor. Therefore, skip connections are proven to be very useful to guide the decoding process and increase detail in image restoration problems by encoder-decoder networks. Consequently, in our approach, we employ 3D skip connections between our encoder and decoder.

4.4.5 Gradient Descent and Adaptive Moment Estimation (ADAM)

After the back-propagation computes the gradients in a network, one needs the gradient descent algorithm to update the parameters to minimize the cost. Let $\nabla_{\theta} E(x, \theta)$ be the gradient of the loss function $E(x, \theta)$ with respect to the parameters $\theta$. Let $\tau \in \mathbb{R}$ and $\theta_0$ serve as step size and the initial guess of the parameter, respectively. The gradient descent then proceeds as follows

$$\theta_{k+1} = \theta_k - \tau \nabla_{\theta_k} E(x, \theta_k)$$  \hspace{1cm} (4.17)

where $k$ represents the number of iteration. The gradient $\nabla_{\theta_k} E(x, \theta_k)$ points out the steepest direction in which the value of the cost function goes at the current step $k$. By negating the gradient, the cost decreases oppositely to that direction. The step size $\tau$, known as the learning rate in the area of machine learning, controls how far it goes to reach the next point $\theta_{k+1}$ in each iteration. As the gradient descent algorithm proceeds with the iteration, we get closer to a minimum of the cost function and thus obtain a candidate of the best parameter $\hat{\theta}$.

Here we gave the basic principle of the gradient descent algorithm. In practice, however, its use is not guaranteed to find the global minimum. First, the cost function in deep learning, especially for image restoration problems, is high-dimensional and often non-convex; that is, there exists multiple local minima. The gradient algorithm itself cannot recognize which one is the global minimum. Second, the learning rate $\tau$ needs to be chosen carefully, otherwise the loss function will not converge. Furthermore, the initial value of the parameter $\theta_0$ decides where the searching starts for the optimal $\hat{\theta}$. Combined with an inappropriate learning rate, it can lead the searching to an awkward situation.

Consider the simple example that is illustrated in Fig. 4.6. Assume that we have a non-convex cost function $E(\theta)$ in which there is a local minimum, and a global minimum. The gradient descent algorithm is applied. With a small learning rate, the descent process lasts a long time. In addition, if the initial value of the parameters sets the start point near the local minimum, then the search gets stuck in the local
Figure 4.6: The search for the global minimum using gradient descent algorithm with inappropriate learning rates $\tau$. Left: Too small a step size leads to a very slow search and gets stuck in the local minimum. Middle: Too large a step size causes drastic updates, such that the global minimum is skipped. Right: Relatively too large a step size lets the search travel back and forth around the minimum, or even gives a divergent behaviour.

minimum and, therefore, loses the opportunity to find the global minimum. By contrast, if the learning rate is too large, with a start point near the global minimum, then the descent step would have the chance to skip the goal and jump to the local one or some points that are even more inappropriate. Relatively too large a step size could also encounter the situation that the searching gets stuck since the descent step goes back and forth on two sides of a minimum but never reaches it.

As mentioned earlier, the maximum-likelihood estimation, which is equivalent to the $L^2$ loss, tries to maximize the probability of seeing the target value given the training samples. If we feed the network with the entire training dataset, then it is highly possible that we obtain the real probability distribution of the dataset thus have the best performance of the network. In fact, this is close to impossible to fit in all the data for the training due to the limit of the hardware. To proceed the gradient descent, we randomly draw a subset, namely a batch from the training data, and update the parameter according to the gradient within the batch. This technique is the so-called stochastic gradient descent (SDG). However, the gradient from the batch can be noisy because the probability distribution of the batch cannot represent that of the entire dataset. Furthermore, recall that the high-dimensional non-convex cost function has multiple local minima. If the Hessian matrix\(^1\) of the cost function is "badly conditioned"; that is, the ratio of largest to smallest absolute eigenvalue is very large, we can imagine in this case a long, steep valley in the graph of the function, where the SGD struggles as the searching has the "zig-zag" route thus moves very slowly towards the minimum.

Fortunately, there are several extensions of SDG to solve the problems mentioned earlier, basically by accumulating gradients over time, averaging over the recent history and then slowly reducing the contributions of previous gradients again. The Adaptive Moment Estimation (ADAM) \[73\] is the most popular among these extensions. Specifically, the algorithm calculates the exponential moving averages of the gradient and the squared gradient, as well as the hyper-parameters $\beta_1$ and $\beta_2$ that control the decay rate

\(^1\)Hessian matrix is a square matrix of second-order partial derivatives of a multivariate function, describing the local curvature of the function. It is used to identify the local minimum of the function by observing its eigenvalues. For more detail about the Hessian matrix we refer the reader to Chapter 4 of Deep Learning by Ian Goodfellow \[42\].
Figure 4.7: Example center view together with ground truth disparity of light fields from the benchmark [60]. Ground truth disparity is provided for the center view only. Most of the light fields have disparity range $[-1.5, 1.5]$.

of these moving averages:

1. $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$ \hspace{2cm} Compute the gradient at current iteration
2. $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ \hspace{2cm} Update biased first moment
3. $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ \hspace{2cm} Update biased second moment
4. $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ \hspace{2cm} Compute bias-corrected first moment \hspace{2cm} (4.18)
5. $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ \hspace{2cm} Compute bias-corrected second moment
6. $\theta_t \leftarrow \theta_{t-1} - \tau \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ \hspace{2cm} Update parameters

where $g_t^2$ is the element-wise square $g_t \odot g_t$, $\tau$ the learning rate and $\epsilon$ a constant ($10^{-8}$) to prevent dividing by zero. $\beta_1^t$ and $\beta_2^t$ donate the $\beta_1$ and $\beta_2$ to the power of $t$, respectively. The initial values of the moving averages for moment estimation and $\beta_1$, $\beta_2$ are close to 1 (recommended as $\beta_1 = 0.9$ and $\beta_2 = 0.999$). Steps 4 and 5 of the bias correction are taken into account if they are initialized with zero vectors. The ADAM not only calculates the adaptive learning rate based on the first-order moment $m$, but also makes full use of the second-order moment $v$ of the gradient. The learning rate $\tau$ is rescaled by dividing the term $\sqrt{\hat{v}_t} + \epsilon$ since $v$ is proportional to the magnitude of the gradient, the parameters receiving large updates get a smaller step size, while the those having small updates will have larger steps. As the iteration goes on, the term $1 - \beta_2^t$ grows and thus gradually reduces the contribution of the previous gradients.

4.5 Data Preprocessing

As input data for our algorithm, we use a variety of publicly available datasets [60, 126, 132, 136], as well as synthetic scenes that were specifically created for the intrinsic images task. In this section, we introduce the data generation process and what kind of ground truth we create for each task. We discuss the procedure of creating training patch and augmentation strategy that we use to prevent overfitting.

4.5.1 4D light field benchmark [60]

The light field benchmark [60] offers 28 light fields rendered with Blender with ground truth disparity available, see Fig. 4.7 for an example. Their composition varies substantially, with many different
4.5.2 New light fields rendered with Blender

We generate data for specular and diffuse separation and full intrinsic light field decomposition using the Blender addon provided with [60], see Fig. 4.8 for an example of the generated scenes. By randomizing scenes, we can generate a (theoretically) infinite amount of different light fields to ensure a large variety of data. We designed multiple scenes containing up to five objects of different scales and geometric complexity. We chose the texture, the reflective properties and the environment map for lighting at random. Additionally, we randomly change the position and rotation of all objects and rotate the environment map to prevent overfitting to certain geometries and lighting conditions. To ensure that the network can also deal with purely Lambertian materials, a certain percentage of objects have purely diffuse material. In total we used 36 pre-built scenes, 321 textures and 109 environment maps collected from different public sources. The 3D models that we use are selected from Chocofur\(^2\) and The British Museum\(^3\). We adapted the material properties to fit our needs and only used the mesh data.

Lightfields are rendered with the Cycles engine, and we adapted the addon [60] such that it can output the intrinsic components. For both diffuse and specular passes, Cycles outputs the three different components color, direct lighting, and indirect lighting. Adding the direct and indirect light and multiplying it by the color yields the desired ground truth separation. Data is stored in high dynamic range to circumvent

---

\(^2\) [http://www.chocofur.com](http://www.chocofur.com)

\(^3\) [https://sketchfab.com/britishmuseum](https://sketchfab.com/britishmuseum)
problems with saturated specularities. The size of these light fields is also $9 \times 9 \times 512 \times 512$. The 175 light fields we use for training contain around 160,000 patches.

### 4.5.3 Real-world light fields

We have four sources for real world light fields, for which no ground truth data is available; see Fig. 4.9. First, we use light fields captured with the Lytro Illum light field camera, calibrated and rectified using the light field toolbox from [31]. The size of the light fields is $9 \times 9 \times 434 \times 625$. We used 11 light fields for training and two for testing, which results in 10,175 training examples. Second, we use a dataset built from the Stanford Light Field Archive [126] with six training datasets which is 6,816 patches, and with two light fields held back for testing. Third, we captured a light field using an industrial camera mounted on a gantry we assembled ourselves. The size of the light field is $9 \times 9 \times 497 \times 710$ with a disparity range of $[-1.5, 1]$. The light field illustrates a non-Lambertian object, illuminated with approximately white light. Finally, we use five real world light fields from the HCI benchmark [132] and we keep one for testing, which results in 16,016 more training patches.

### 4.5.4 Augmentation

To prevent overfitting, we use the following augmentation strategy. As discussed earlier, we divide training data into patches and select samples for minibatch at random. If we do not have ground truth for the data sample $x$, then we simply apply random color perturbation

$$
a = \frac{\text{Norm}[0, 1]}{k} + 1
$$

$$
b = \frac{\text{Norm}[0, 1]}{k}
$$

$$
x' = a x + b
$$

(4.19)

to each color channel of $x$. We then scale $x$ such that it is in the desired color range. The parameter $k$ controls the amount of perturbation.

In case of decoding intrinsic components, we need to augment them to still satisfy physical principles of image formation. To augment ground truth albedo component $x_a$ of sample $x$, we use simple augmentation procedure described earlier (4.19). Thus, albedo or true color of objects in the scene is slightly changed. For shading component $x_{sh}$ we only scale its RGB values without introducing an offset. By doing this, we simulate the slight change of the light source color and its intensity. Specular component $x_s$ is scaled proportionally to the shading because if the light intensity changed than specularity might
become brighter or darker. To obtain the input patch $x$, we use intrinsic light field model described in the previous chapter

$$x' = x'_a x'_{sh} + x'_s$$  \hspace{1cm} (4.20)

Due to color randomization, the network never sees the same patch again; see the example of augmentation in Fig. 4.10. We do not augment the disparity and do not apply any geometric or affine transformations to the data because these transformations will break the epipolar structure of the light field.

We introduced all necessary CNN background and general framework. We will now show some applications of deep autoencoder for light fields to state-of-the-art computer vision tasks. We start with reflection separation and disparity estimation, we then proceed with full intrinsic light field decomposition, which is more challenging due to its computational complexity. We conclude the chapter with a super-resolution network.

### 4.6 Reflection Separation and Disparity Estimation CNN

As discussed in the previous chapter 3, according to the dichromatic reflection model proposed by Shafer [101] and adopted for light fields by Tao et al. [120],

$$L(r) = D(r) + H(r)$$  \hspace{1cm} (4.21)

the light reflected from a scene point has two independent components: light reflected from the surface body $D(r)$ and at interface $H(r)$. Body reflection is known as the diffuse component and it is independent of viewing direction, while interface reflection is the specular component $H$, which is view-dependent. The diffuse component can be further decomposed into albedo $A$, which represents a color of an object independent of illumination and camera position, and shading $S$, which describes intensity changes due to illumination, inter-reflections, and object geometry.

Diffuse specular separation is an ill-posed problem that is still an active field of research in the computer vision community. In the work by Sulc et al. [110], we show that specularity appearance in a light field depends on the scene geometry and constant along the specular flow directions, while the Lambertian component remains constant alone the disparity direction. Thus, it feels natural to use a geometrical information encoded in light field to perform reflection separation.

We adopt the proposed architecture from Section 4.2 to perform together disparity estimation and reflection separation tasks. This project is described in the research paper [7]. In extensive experiments, we show that our network outperforms state-of-the-art reflection separation algorithms and single image CNN methods.
Figure 4.11: The pathways of our deep encoder-decoder network are organized in six groups of three residual blocks each. The first two blocks in each encoder group keep depth and resolution the same, the last block reduces resolution (shown on bottom, viewpoint × spatial coordinates), while increasing feature depth (shown on top) by 32. The decoder paths are exact mirrors of this chain. Disparity is only a 2D decoder, where the viewpoint dimension of the shape is removed.

4.6.1 3D reflection separation and 2D disparity estimation architecture

In the encoder pathway, 18 residual blocks are chained together. Every third one reduces the patch resolution via strided convolution while increasing feature depth, with the overall goal of gradually reducing dimensionality. The final output has shape $3 \times 3 \times 3 \times 192$, for an overall reduction of the input to around 8.3% of its original size, see figure 4.11.

Light field, diffuse and specular components are reconstructed for the $17 = 9 + 9 - 1$ views in a crosshair around the center view. The disparity map is only computed for the center view. We employ the dichromatic reflection model [101], see Equation (4.21). According to this model, the specular component is assumed to be independent from the diffuse one, which justifies the use of two separate decoder chains. However, they should also sum up to the input light field. To let the network better cope with this constraint, we append specular features to the diffuse ones, and vice versa, but only for the input to the final layer. Because the disparity output is only 2D, we reduce the filter shape by the respective dimension.

4.6.2 Network implementation and training strategy

From the training data, we set aside 5% for a validation set. Several light fields are also completely held back, and used only for testing (see above for details). We implement the network using Tensorflow in Python3, and train on an Intel Core i9 system with four nVidia Titan Xp, with the encoder/decoder chains distributed to different GPUs to satisfy memory requirements for training. All decoders are trained with
Chapter 4 Deep Autoencoder for Light Fields

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$L^2$-loss times 100, validation data</th>
<th>$L^2$-loss times 100, training data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AE diffuse specular disparity</td>
<td>AE diffuse specular disparity</td>
</tr>
<tr>
<td>Synthetic</td>
<td>0.860 – – 6.114 0.816 – – 5.964</td>
<td>0.568 1.456 1.393 1.419</td>
</tr>
<tr>
<td>Benchmark [60]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>0.610 1.577 1.511 1.620</td>
<td>0.568 1.456 1.393 1.419</td>
</tr>
<tr>
<td>Real-world</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lytro Illum</td>
<td>0.606 – – –</td>
<td>0.574 – – –</td>
</tr>
<tr>
<td>Stanford [126]</td>
<td>1.045 – – –</td>
<td>0.919 – – –</td>
</tr>
<tr>
<td>HCI [132]</td>
<td>1.230 – – –</td>
<td>1.150 – – –</td>
</tr>
<tr>
<td>Average</td>
<td>0.8702 1.577 1.511 3.867</td>
<td>0.8054 1.456 1.393 3.6915</td>
</tr>
</tbody>
</table>

Figure 4.12: Network losses for different groups of datasets at convergence. The datasets most difficult to fit for the autoencoder are the ones from gantries, perhaps due to minimally uneven sampling of viewpoints which has not been properly corrected. Depth reconstruction on our own synthetic dataset is surprisingly easier than for the benchmark datasets, although it has much stronger specularity. However, the geometry of our objects is also substantially simpler, and the datasets have large regions of easy to fit planes. Overall, disparity MSE on the benchmark validation is around the current benchmark average, which is 6.29. However, our model is not specifically optimized for depth reconstruction, and in particular trained for non-Lambertian scenes, on which it can perform much more robustly than competing methods, see figure 4.13.

Reconstruction of a single pathway during evaluation requires roughly 7 seconds on this system for a light field with a center view resolution of $512 \times 512$, including tiling of the input light field, all transfers from CPU to GPU and back, and reassembling the output from the patches. The complete specular/diffuse decomposition with disparity estimation takes 19 seconds. We will verify the quality of reflection separation and disparity estimation in detail in the next section.

4.6.3 Qualitative and numerical evaluations

We compare our reflection separation with two algorithms designed for light fields. The first, by Sulc et al. [110], performs reflection separation based on specular flow. The second one is drawn from our work [6], which performs intrinsic light field decomposition presented in the previous Chapter 3. In addition, we compare to the network proposed by Shi et al. [103], which uses a deep autoencoder for intrinsic images. However, it only works for standard 2D images. To compare to the full decomposition [6], where we decompose the input light field into albedo, shading and specularity, we compute the diffuse component by multiplying albedo and shading [46].

For quantitative results, we evaluate reflection separation on synthetic scenes and report the local mean-squared error (LMSE) [46], which we compute patch-wise. This error is scale invariant because the brightness of the patches is adjusted to the ground truth. In our experiments, we use rectangular overlapping patches with a size of 20% of the total image size. To evaluate the errors that might be canceled by LMSE, we also compute global mean squared error (GMSE) that adjusts the brightness value for the whole image. We also measure the structural similarity index (SSIM). See Fig. 4.28 for an overview of
Figure 4.13: We compare our results for disparity on challenging synthetic scenes that feature strong specularities and regions of little texture against state-of-the-art methods for depth estimation. Especially in regions where the specularity dominates the texture, the other EPI based methods fail, while ACC due to its strong regularization can still yield pleasing (albeit oversmoothed) results. With respect to MSE, our approach outperforms the other methods significantly.

<table>
<thead>
<tr>
<th></th>
<th>LMSE ×100</th>
<th>GMSE ×100</th>
<th>SSIM ×100</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Proposed</strong></td>
<td>0.15</td>
<td>0.28</td>
<td><strong>80.08</strong></td>
</tr>
<tr>
<td>Alperovich [6]</td>
<td><strong>0.12</strong></td>
<td>0.22</td>
<td>74.98</td>
</tr>
<tr>
<td>Sulc et al. [110]</td>
<td><strong>0.12</strong></td>
<td>0.23</td>
<td>75.43</td>
</tr>
<tr>
<td>Shi et al. [103]</td>
<td>0.34</td>
<td>0.5</td>
<td>63.02</td>
</tr>
</tbody>
</table>

Figure 4.14: Comparison of different error metrics for specular and diffuse components. Numbers show the average over nine previously unseen test datasets. See Section 4.6.3 for a description of the metrics. Because Shi et al. [103] does not perform decomposition for the background, we multiply all results and ground truth with object mask before measuring the errors.

all numerical results, and Figs. 4.18 and 4.15 for a visual comparison. We also compare performance of disparity map estimation for specular scenes to different other algorithms in Fig. 4.13.

As an ablation study, we performed two experiments. In the first case we only trained the network for center view without any disparity information from sub-aperture views, in the second case we reduced spatial patch size to 24 × 24. Both experiments lead to decrease in performance compared to the original network, see Fig. 4.16 for the comparisons on the same datasets that are used in Figs. 4.18, 4.28, 4.13.

Finally, results of our method on different datasets that are commonly used in the light field community [60, 126, 132] can be found in Fig. 4.17.

We show the performance of our network visually by presenting more results for the real and synthetic scenes. To demonstrate the large variety of data that our method works on, we evaluate on real world Lytro light fields, see Fig. 4.24, and our own synthetic datasets created for this project.

We present the results for both seen (see figures 4.19, 4.20) and unseen light fields (see Figs. 4.21, 4.22, 4.23). Given that the ground truth is only available for our synthetic scenes and [60], we show quantitative evaluations only for those datasets. For visual evaluation, we show center views of diffuse and specular components, estimated disparity, and center view of reconstructed light field.
4.6.4 Conclusions

In this work, we propose a generative encoder-decoder architecture for patches taken from light field epipolar volumes. Using different decoder paths, we can achieve both intrinsic decomposition and disparity estimation with a unified network. Thanks to joint training of autoencoder and the supervised pathways, we can transform the input light field into a latent representation, which is both much smaller and well adapted to the desired tasks. To our knowledge, this is the first time that these problems have been attacked with a single network. In addition, it is easy to include decoders for more tasks if training data can be provided.

Our method outperforms recent light field based methods [6, 110], and a single image deep network approach for intrinsic image decomposition [103]. For instance, compared to the variational approach [6] that was presented in Chapter 3, our CNN based method performs visually and numerically better and requires much less time to compute the decomposition (less than a minute versus one hour).

Although we have only average performance in depth reconstruction on datasets from the benchmark [60], in contrast to other methods, we still recover reliable depth in the presence of strong specularity. We also generalize well to real-world light fields captured with the Lytro Illum plenoptic camera or a gantry, although we do not have ground truth training data available for these. Despite being trained only on soft reflections, experiments with highly specular light fields show that we are robust against strong non-Lambertian effects. As the structures in epipolar volumes are both relatively characteristic and contain
Chapter 4  Deep Autoencoder for Light Fields

<table>
<thead>
<tr>
<th></th>
<th>LMSE $\times 100$ diff.</th>
<th>GMSE $\times 100$ diff.</th>
<th>SSIM $\times 100$ diff.</th>
<th>MSE (depth) $\times 100$ scene 1</th>
<th>scene 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>0.25 0.19</td>
<td>0.64 0.62</td>
<td>66.66 72.75</td>
<td>5.9 4.6</td>
<td></td>
</tr>
<tr>
<td>48 x 48</td>
<td>0.33 0.33</td>
<td>0.74 0.73</td>
<td>56.67 59.07</td>
<td>192.7 167.9</td>
<td></td>
</tr>
<tr>
<td>9 x 24 x 24</td>
<td>0.28 0.35</td>
<td>0.69 0.85</td>
<td>57.72 62.87</td>
<td>55.87 19.31</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.16: Ablation study: Quantitative comparison of separation over nine previously unseen test datasets, and depth estimation for the two scenes from Figure 4.13. Note that we compute error for the whole center view, without object mask.

more information, we only require a relatively few training examples (around 200 light fields), compared to single image approaches which use several millions of images.
Figure 4.17: Results on unseen light fields from various sources. We show center views of the light fields with diffuse and specular components and estimated disparities. Top: lightfield from the Stanford dataset [126], where we have chosen the most challenging case with respect to reflection separation and disparity estimation. Our network, while being trained on synthetic scenes, is able to generalize to real-world examples with complicated geometry and reflection. Middle: synthetic scene from light field benchmark [60], where we have selected an object with small specular regions, to evaluate how the network will cope with it. Specularity is successfully removed from the diffuse part, while preserving texture. Bottom: an example dataset from HCI benchmark [132].
Chapter 4 Deep Autoencoder for Light Fields

Figure 4.18: Comparison for a synthetic dataset with two non-Lambertian objects with almost no texture, which is typically challenging for reflection separation. Both modeling approaches [6] and [110] fail to separate the specular component from the diffuse one. The CNN-based approach [103] successfully separates reflection components, but the diffuse one has some artifacts. In addition, the method requires an object mask, thus its application is limited to objects well separated from the background, which are rarely found in real world scenes.

Figure 4.19: Results of our network for the synthetic dataset generated with Blender using Cycles rendering engine. We show the ground truth and our results of the center view, disparity, diffuse and specular components. The light field resolution is $9 \times 9 \times 512 \times 512$ with disparity range $[-2.55, 1.44]$. The dataset contains a glossy object with complicated geometry, and a mixture of planar and curved surfaces. The foreground object has almost no structure. This light field was used to train the network. Error measure LMSE: diffuse = 0.2, specular = 0.05. Error measure GMSE: diffuse = 0.24, specular = 0.08. Error measure SSIM: diffuse = 67.58, specular = 84.26. Error measure MSE: disparity = 3.92.
Figure 4.20: Another synthetic training example. The dataset has an object with soft reflection. The background contains some texture with clear pattern. We illustrate how our network recovers disparity from the object with large specular highlight. This example supports the idea that disparity estimation may benefit from the knowledge about diffuse and specular components. Error measure LMSE: diffuse = 0.11, specular = 0.12. Error measure GMSE: diffuse = 0.14, specular = 0.17. Error measure SSIM: diffuse = 56.16, specular = 83.23. Error measure MSE: disparity = 3.45.

Figure 4.21: Synthetic dataset that is used for testing. With non-trivial illumination, the scene exhibits lots of specularity. The geometry of the object is complicated with multiple fine details that create challenges for disparity estimation. We show that our network copes well with non-trivial geometry and lighting conditions. Error measure LMSE: diffuse = 0.11, specular = 0.13. Error measure GMSE: diffuse = 0.13, specular = 0.15. Error measure SSIM: diffuse = 84.25, specular = 77.85. Error measure MSE: disparity = 18.42.
Chapter 4  Deep Autoencoder for Light Fields

Figure 4.22: Unseen synthetic light field generated in Blender. This dataset shows an object with fine details and curved surfaces. The color pallet of background and foreground is similar, which is a complicated case for both disparity estimation and reflection separation tasks. Error measure LMSE: diffuse = 0.19, specular = 0.3. Error measure GMSE: diffuse = 0.29, specular = 0.37. Error measure SSIM: diffuse = 68.57, specular = 51.52. Error measure MSE: disparity = 5.82.

Figure 4.23: The last synthetic dataset used for testing that we show in our experiments. This light field contains an object with almost diffuse reflection and very soft specularity. With this example, we want to show how our network copes with almost Lambertian scenes. Error measure LMSE: diffuse = 0.11, specular = 0.25. Error measure GMSE: diffuse = 0.25, specular = 0.37. Error measure SSIM: diffuse = 77.28, specular = 84.5. Error measure MSE: disparity = 3.54.
Figure 4.24: Real world light fields captured with Lytro Illum plenoptic camera. The scenes are captured under various illumination conditions, indoor and outdoor. Most datasets have a single foreground object. Objects consist of different materials, thus they exhibit various specularity ranging from the soft specular reflection of the fur to strong reflections from the metal surfaces. For all scenes we illustrate the same components as for the synthetic examples. Since it is a real world light fields, the ground truth is not available.
Chapter 4 Deep Autoencoder for Light Fields

4.7 Deep Intrinsic Light Field Decomposition CNN

With the growing popularity of data-driven methods, our work [5] focuses on intrinsic light field decomposition and disparity estimation with an encoder-decoder end-to-end deep neural network. These tasks can benefit from each other in a sense that estimated disparity contains information about the geometry of the scene, which is strongly related to shading and specularity. Moreover, albedo, shading, and specular flow are useful cues for disparity estimation.

Similar to the previous network example, see Section 4.6, where we only recover disparity, diffuse and specular components, we also use horizontal and vertical 3D volumes as an input to our network. Contrary to the previous network, we perform full intrinsic decomposition, design an architecture that is capable of processing twice larger patches as an input and introduce skip connections from the encoder to corresponding decoder parts that improve the reconstruction quality of decoders. We substitute 3D convolutions with a 2D sequence, which acts in spatial and angular domains. This design choice decreases the number of parameters in the network and speeds up the training process. As in the previous case, this network can process light fields where a ground truth is not available.

We evaluate the network on synthetic data generated with Blender and real world examples taken with Lytro Illum plenoptic camera. We perform comparisons with a single image CNN-based algorithm and recent methods for light fields.

4.7.1 Underlying model

According to the dichromatic reflection model [101], the total radiance $I$ of the reflected light is the sum of two independent parts: the radiance $I_d$ of the reflected light at the surface body and radiance $I_s$ at interface. More precisely,

$$ I(\lambda, n, l, v) = I_d(\lambda, n, l, v) + I_s(\lambda, n, l, v), \quad (4.22) $$

where $\lambda$ is the wavelength of the light, $n$ is the surface normal, $v$ is the viewing direction, and $l$ direction to the light source. In particular, it is a complex function of the local surface geometry.

As discussed in detail in [119], we consider the case where the diffuse component is modeled as Lambertian, and can rewrite (4.22) for a light field $L$ as

$$ L(v) = m_d(n, l)c_d(\lambda) + m_s(n, l, v)c_s(\lambda), \quad (4.23) $$

where $m_{d,s}$ is a geometric scale factor, $c_{d,s}$ is the spectral power distribution. In short, we can represent a light field as a sum of diffuse and specular $H$ component. By further breaking down diffuse component into albedo $A$ and shading $S$, we arrive at the intrinsic light fields

$$ L(r) = A(r)S(r) + H(r) \quad (4.24) $$

model described in Section 3.2.1 and in our research paper [4].

We can see that according to our model, specular and diffuse components behave quite differently in EPIs [30]. While the albedo and shading have constant color along the projections of 3D points given by disparity, specular reflection moves in a non-rigid way and depends on local surface geometry [110, 112]. Based on these definitions and observations, we can now start to build our network.
Figure 4.25: Encoder-decoder residual network with 12 convolution layers in the encoder part and corresponding up-sampling layers in the decoder. The network has four 3D decoding pathways: albedo, shading, specularity and light field itself, and one 2D pathway for disparity. To keep the diagram readable, we do not illustrate skip connections. The arrow in the last decoding layer for intrinsic components illustrates that albedo, shading and specularity shares their features to better cope with modeling cost (4.24). Numbers on top describe the output dimension of a tensor after each layer. Numbers on bottom illustrate the size of feature maps after each block of operations. Block color corresponds to its kind, for instance for encoder and decoder pathways purple is spatial downscaling, blue is dimension preserving, and orange is angular downscaling.

4.7.2 Intrinsic light field CNN architecture

Similar to the previous work [7, 62, 103] we use a U-shaped mirror-like architecture [98] to build the network, see Fig. 4.25 for the detailed explanation of the network structure. The key idea is to extract a small number of features, 3.8% of the input patch size, and then upscale them back to the input light field, intrinsic components and disparity.

The input to the network is a pair of $9 \times 96 \times 96 \times 3$ horizontal and vertical 3D slices of the light field which have an overlap of 32 pixels. Compared to the original architecture from Section 4.2, here we use twice larger size of patches because decomposition of $9 \times 48 \times 48$ could be ambiguous due to lack of information in the patch. It especially applies to shading component. Because we increase the spatial resolution of the input patch, our network becomes bigger in terms of parameters and memory required to store the graph. To perform training, we need to find a trade off between the size of the network, the speed of computation and the accuracy of the results.

The concept of separable convolutions is now widely used in machine learning [26]. The main idea is to split the 2D convolution into a spatial kernel, which acts on each slice of the features volume and 1D convolution that acts depth-wise. This design choice aims to decrease the number of parameters in the network and speed up the training process. The obvious drawback is that the dependency between spatial dimension and feature depth will be lost, thus the network would give less accurate results. In our case,
we apply similar strategy to 3D convolutions. We replace 3D kernels with the sequence of 2D filters: the first filter acts on the epipolar images in the angular domain, and the second filter acts in the spatial domain. Spatial convolution is applied to each sub-aperture view independently.

Every patch passes 12 convolution layers, where each layer is represented by the residual block; see Fig. 4.26. This is similar to the one that we introduced in general architecture, see Fig. 4.3, but instead of 3D convolutions it has 2D. After batch normalization, we duplicate the output tensor and successively apply 2D convolution on the EPI followed by 2D spatial convolution to one copy. We then pass the output through the Exponential Linear Units (ELU) layer [28]. Another copy is kept unchanged or resampled to have the same shape as the first copy. The output of the block is the sum of two copies.

Based on the strides used, the residual blocks are divided into three groups: spatial downscaling, angular downscaling and dimensions preserving; see Fig. 4.25.

To improve the accuracy of decomposition, we add skip connections [62, 103] by copying encoder features to the corresponding outputs of the decoder layers. We then pass a new tensor through a \(1 \times 1 \times 1\) convolution to preserve its original shape. Note that there are no skip connections in the pure autoencoder.

The decoder consists of five pathways. To ensure that intrinsic components follow the model (4.24), we concatenate their features in the last decoding layer and then up-sample to the output albedo, shading, and specularity.

We use scale invariant loss [32, 91]

\[
\alpha = \frac{yy^*}{y^2} \quad (4.25)
\]

\[
E_{scale} = \lambda \|y - y^*\|^2 + (1 - \lambda)\|\alpha y - y^*\|^2
\]

for albedo \(A\) and shading \(S\) to reduce the ambiguity caused by the product of those components in the intrinsic decomposition, see model (4.24). Here, \(y^*\) denotes the ground truth value and \(y\) denotes the output of the network, \(\lambda\) is a weight that alternates between MSE and scale-invariant loss function. For all of the other decoders, we use standard MSE loss function.
Chapter 4 Deep Autoencoder for Light Fields

Proposed

|----------------|-----------|-------------|

Figure 4.27: Intrinsic decomposition for real world $9 \times 9 \times 434 \times 625 \times 3$ light field captured with a Lytro Illum plenoptic camera. Rows from top to bottom: albedo, shading, and specularity. Note, that method by Garces et al. [39] performs only albedo/shading decomposition. We use space for the specular image to illustrate the center view. The single image CNN [103] does not perform decomposition for the background, thus it appears black in the visualization. We conclude that our method copes well with soft shadows and highly specular materials. It also preserves all the structure compared to over-smoothed results from our modeling approach presented in the research paper [6] and described in Chapter 3.

4.7.3 Experiments

Most of the training data is generated with the Blender addon provided by [60]. For unsupervised training, we use the light field benchmark [60] with only disparity ground truth available and real world light fields [108, 126, 134, 136] without any ground truth.

We train the network with batch size 10 for $160K$ iterations starting from learning rate $10^{-4}$ and dropping it every $40K$ iterations till it reaches $10^{-5}$. The evaluation of the trained network takes about 11s for $3D$ pathway and 2.6s for $2D$ on a machine with an Intel(R) Core i7-4790 CPU 3.60GHz and an NVIDIA GeForce GTX 1080Ti.

Because there are no algorithms for light fields that jointly solve intrinsic decomposition and disparity estimation, we perform separate comparisons for these tasks.

For intrinsic decomposition, we select three methods for comparison. The first method is a modeling approach for the light fields that we proposed [6] and described in detail in Chapter 3, where we model priors according to their physical properties. The second method is proposed by Garces et al. [39], where the authors decompose the input light field into albedo and shading components with extended Retinex theory. The third method is a single image CNN-based method by Shi et al [103], where the authors develop a deep network for non-Lambertian intrinsic decomposition. See Figs. 4.27, 4.33, 4.34 for results on the real world data, and Fig. 4.30 for evaluations on the synthetic test data.

Note that the method by Garces et al. [39] uses the whole light field as an input, thus we perform comparisons only for the real world data captured with a Lytro Illum plenoptic camera, where we have all 81 views available. Our variational approach [6] use horizontal and vertical slices of the light field,
Figure 4.28: Numerical evaluation of intrinsic decomposition. The numbers represent average LMSE, GMSE, and DSSIM over nine synthetic test datasets. Because the single image CNN [103] does not perform decomposition for the background, we multiply all results and ground truth with object mask before measuring the errors. We conclude that the use of light fields improves intrinsic decomposition, compared to [103]. Our method outperforms modeling approach [6], especially on the DSSIM measure.

and Shi et al. [103] use single image plus object mask. This information is available in our synthetic data.

For quantitative evaluations, we select three error metrics: local mean-squared error (LMSE) [46] computed patch-wise with the size of 40% of the image size, global mean-squared error (GMSE) [6], which is similar to LMSE, but computed for the whole image, and DSSIM index [25] which is defined as $(1 - SSIM)/2$ and measures structure dissimilarity. See Table 4.28 for numerical evaluations on the center view over 9 test datasets.

For disparity estimation, we select four algorithms for comparison. Our deep network [7], which is described in the Section 4.6, jointly decomposes input light field into diffuse and specular components and estimates the disparity. Johannsen et al. [67] employ dictionary learning to recover disparity. Strecke et al. [109] estimate disparity with occlusion-aware focal stack symmetry with additional normals refinement. The last is the method proposed by Wanner and Goldluecke [131], which is based on orientation of EPI patches. We use the standard MSE metric for the comparisons, see Fig. 4.31 for visual and Table 4.29 quantitative evaluations.

We illustrate more results of the network on datasets from different sources. Fig. 4.33 shows the decomposition of the outdoor scene captured with Lytro Illum plenoptic camera. This scene is a part of flowers benchmark [108]. We also introduce the results from the Stanford benchmark [126], see Fig. 4.34.

We use overlapping patches to compute the decomposition as the weighted average, assuming that pixels that are close to the patch center are more accurate. Thus, we exclude border pixels from the final results.

Figure 4.29: Quantitative evaluations of disparity map for three synthetic scenes. We conclude that the proposed method outperforms by large margin state-of-the-art disparity estimation algorithms and recent CNN for light field. Thus, we conclude that use of large input patches bring enough information for the accurate disparity estimation, even for specular objects with little structure.
4.7.4 Conclusions

We propose a novel architecture that outperforms recent methods for disparity estimation and intrinsic decomposition. The key idea is to replace 3D convolutions with the sequence of 2D angular and spatial convolutions, which decreases the number of parameters in the network. The resulting architecture is more computationally efficient and allows larger patch sized compared to the previous architecture [7] described in Section 4.6. Consequently, we are able to train the network with four 3D and one 2D decoders.

From our experiments, we conclude that with a larger patch size the disparity estimation is much more accurate, especially on the large specular surfaces, see Fig. 4.31 and Table 4.29. Although we loose some angular-spatial information due to replacement of 3D convolutions with the sequence of 2D convolutions, the results confirm the advantage of this strategy. Our experiments show that we have visually and numerically better results if we increase the size of the input patch. This behavior can be explained by the fact that it is not enough structure to correctly estimate disparity in small patches.

While ground truth is available only for the synthetic scenes, the proposed architecture still generalizes well to the real world light fields, it successfully removes most of the shading from the albedo component, and it correctly detects and separates specularity.

Given the difficulty of the tasks, our network achieves superior performance, but there is still plenty of work ahead. The estimated shading component contains some structure from the albedo, thus additional post-processing might be needed. In addition, our current architecture can deal only with soft shadows because training scenes are illuminated with environmental maps, which results in soft lighting.
Ground Truth | Proposed | Alperovich [6] | Shi [103]
--- | --- | --- | ---
![albedo](image1) | ![proposed_albedo](image2) | ![alperovich_albedo](image3) | ![shi_albedo](image4)
![shading](image5) | ![proposed_shading](image6) | ![alperovich_shading](image7) | ![shi_shading](image8)
![specularity](image9) | ![proposed_specularity](image10) | ![alperovich_specularity](image11) | ![shi_specularity](image12)

Figure 4.30: Comparison on two synthetic datasets generated with Blender. The light field size is $9 \times 9 \times 512 \times 512 \times 3$. We conclude that proposed network outputs more accurate albedo and shading components compared to single image CNN [103]. Compared to the modeling method by [6], the albedo is much sharper and specularity is more intense. The shading component produced by our method still contains some texture compared to shading by [6], we also do not train the network on examples with strong cast shadows, thus shadows are not fully removed from albedo. Due to ambiguity between albedo and shading, we observe the difference between scaling of the ground truth and decomposition results.
Chapter 4 Deep Autoencoder for Light Fields

Example 1  Example 2  Example 3

Center view

Ground Truth

Proposed

Alperovich [7]

Strecke [109]
Chapter 4  Deep Autoencoder for Light Fields

Example 1  Example 2  Example 3

Johannsen [67]

Wanner et al. [131]

Figure 4.31: Ground truth and estimated disparity for three synthetic datasets generated with Blender. The disparity range is $[-2.12, 2.51]$. Compared to [7] our method improves disparity estimation in large specular regions. We explain this behavior by increase of input patch size. Other methods also fail to estimate correct disparity on specular and structureless surfaces. Although we use similar loss function for the disparity, due to the skip connections, the new disparity is sharper than in [7]. Numerical evaluations from the table 4.29 support qualitative results, MSE error is smaller than for the competing methods.

center view  Proposed  Alperovich [7]


Figure 4.32: Disparity estimation for the real world light field from Fig. 4.15. The estimated disparity range is $[-2.44, 1.14]$. This example is particularly difficult for disparity estimation because it has highly specular object. Based on visual comparison we conclude that our method produces similar quality results compared to CNN-based and modeling approaches. Strecke et al.’s [109] method produces visually more accurate results on the saxophone.
Figure 4.33: The center view result on real world flowers dataset [108] taken with Lytro Illum plenoptic camera. The light field size is $9 \times 9 \times 376 \times 541 \times 3$, with the estimated disparity range $[-1.19, 0.57]$. The albedo component is well separated from shading, which is especially visible on the flower. Specular component contains all highlights and does not have any leftover information from the albedo and shading. All intrinsic components appear sharp compared to the reflection separation network 4.6, see figure 4.24 for the results of the network on the same dataset.
Figure 4.34: Another real world example captured with a gantry from Stanford benchmark [126]. This scene is particularly complicated case for intrinsic images because the object exhibit strong specularity and is semi-transparent. Although we do not model transparency and we did not have any semi-transparent objects in our training data, we see that the network copes well with this case. The network output acceptable results, but there is lot of space for the improvement, especially in separation of albedo and shading components. Compared to the previous approach 4.6 described in the previous section, the disparity component is sharper and less noisy.
4.8 Generative Adversarial Network for Light Field Super Resolution

When capturing the light field of a scene, one typically faces a trade-off between more spatial or more angular resolution. Fortunately, light fields are also a rich source of information for solving the problem of super-resolution. Contrary to single image approaches, where high-frequency content has to be hallucinated to be most likely source of the downscaled version, sub-aperture views from the light field can help with an actual reconstruction of those details that have been removed by downsampling. In this paper [145], we propose a 3D generative adversarial autoencoder network to recover the high-resolution light field from a low-resolution light field with a sparse set of viewpoints. We require only three views along both horizontal and vertical axis to increase angular resolution by a factor of three, while at the same time increasing spatial resolution by a factor of either two or four in each direction, respectively.

4.8.1 Introduction

The problem of super-resolution, where one wants to recover a high-resolution image from one or more low-resolution versions, is one of the central challenges of computer vision. Typically, this is a highly ill-posed problem. Meanwhile, the classical approach that solves an inverse problem requires carefully constructed image priors [9]. It is well-known that having multiple low-resolution images with slightly different viewpoints is both crucial and successful in recovering sharp details in the reconstructed image [34, 35, 105, 133]. Such a dense collection of viewpoints is described by the light field of a scene. However, there is typically a conflict between either having more views (angular resolution) or more spatial resolution within the views. Therefore, light field super-resolution aims to increase both the spatial and the angular resolution, where the latter amounts to generating novel views of the scene.

Although optimization-based approaches [99, 104, 133] can provide results of a good quality, their parameters usually need to be fine-tuned and they often require a lot of time to converge. Recently, inverse problems have been successfully attacked just by brute-force learning a prediction of the desired result given the observed inputs. This is due to the rapid development of deep learning techniques in the last decade, which was made possible by the dramatic increase in computational power of GPUs and the availability of large amounts of training data. Architectures based on deep encoder-decoder convolutional neural networks (CNN) have become very powerful for super-resolution, even of single images [85,124], where they zoom a single low-resolution image to impressive high-resolution quality. However, fine detail is necessarily hallucinated and reproduced from natural image statistics, as the downsampling removes the high-frequency content. Multiple input images are required to be able to really recover a truthful high-resolution image.

Deep learning methods are nowadays also popular in the light field community. In recent work, the rich information inherent in the light field was successfully used to built deep networks for diverse tasks, such as disparity estimation, view synthesis, reflection separation and intrinsic image decomposition [5, 7, 108]. We have already illustrated two use-cases in the previous sections. The super-resolution problem was also addressed with recent CNN architectures [34, 142].

One of the biggest challenges in light field super-resolution is the high dimensionality of the problem. The light field itself is a 4D data structure, its upscaled version is two or four times larger and it requires huge amounts of GPU memory to process it with adequate batch size and a sufficient number of feature maps. Thus, the plain architecture where the input image is upscaled to the desired resolution and then refined with a convolutional network is not applicable to light fields. Another challenge is that the light field is 4D and cannot be directly used in the current deep learning frameworks where convolution operations are performed only in 2D and 3D spaces. Finally, capturing a ground truth light field of good quality for training is difficult because, for example, the consumer Lytro Illum plenoptic camera produces
blurry images that suffer from noise, while the higher quality Raytrix camera is of plenoptic type 2.0 and thus sub-aperture views are unavailable. Although it is possible to use a high-quality camera mounted on a gantry, which is a time-consuming approach that can hardly be performed in the wild and is only applicable to static scenes, so the training data will necessarily be limited.

**Contributions.** In this work, we address the problem of light field super-resolution; that is, obtaining a light field with larger spatial and angular resolution from only five sub-aperture views of a low-resolution light field. The proposed approach uses the information from neighboring views and benefits from the dense redundant structure of the light field. A fully convolutional asymmetrical encoder-decoder is built as the first network architecture to transform the 4D structure of the light field to its upscaled version. To enhance the sharpness of the reconstructed light field, we propose a novel WGAN loss that penalizes the difference between angular and spatial derivatives of the generated light field and its ground truth. Contrary to the original GAN based architectures for super-resolution [78], where the discriminator takes generated high-resolution image and the ground truth, we also feed the discriminator network with derivative information, which is more simple and sparse than the original light field because it only contains edge information. To avoid artifacts, we use Wasserstein distance in the discriminator and adversarial losses proposed by Arjovsky et al. [8] instead of the original GAN by Goodfellow et al. [43]. Those design choices make WGAN training very stable and avoid any unexpected distortions that we have otherwise observed. We perform both spatial and angular super-resolution with scale factors two and four, given only a sparse set of sub-aperture views. Our network achieves results with competitive quality for both artificial and real-world light fields compared to state-of-the-art conventional methods and single image SRGAN.

### 4.8.2 Outline of model and losses

In this section, we present an overview of the general structure of our deep neural network model and the key formulas that were used to construct the loss functions of the network. We start by briefly reviewing notation and basic definitions of the light field structure, and we then discuss our modifications to the typical encoder-decoder model and discriminator loss as introduced in [8, 43].

**Encoder-decoder model for light fields**

Let $L_l$ be the low-resolution version of the light field $L$. With the convolutional encoder network $E$, we project $L_l$ onto the latent variable space $Z$ to obtain the latent representation $z = E(L_l)$. Using $z$ as an
Figure 4.36: Proposed network architecture. The input of the network consists of three views from the vertical and horizontal stacks in the cross-hair that are framed in red and green, respectively. All views are split into $48 \times 48$ patches with 16 overlapping pixels with each of their neighbors to decrease the dimensionality of the data. The volumes of size $3 \times 48 \times 48$ are downscaled spatially to $3 \times 3 \times 3$ when they reach the latent space. The latent features are decoded and upscaled separately by nine 2D decoders to achieve the angular super-resolution. In the upscaling phase, the volumes of the decoded sub-aperture views are again spatially upscaled to twice and four times the size of the input; that is, $9 \times 96 \times 96$ and $9 \times 192 \times 192$ to obtain our output. Finally, we introduce the DiffWGAN to distinguish the output images and the ground truth by their pixel values, as well as the spatial and angular derivatives, and thus improve the details in the super-resolved images.

input, we generate the output high-resolution light field $U(z)$ with the convolutional decoder network $U$. We term the concatenation of encoder and decoder as generator $G$, which generates high-resolution light fields from low-resolution light fields. Its output should be the same as the high-resolution ground truth $L_h$ provided to the network.

**DiffWGAN discriminator for light fields**

On top of the output of the generator, we model the discriminator network $D$. Overall, we use a deep WGAN architecture that was originally proposed by Arjovsky et al. [8], with a few modifications. First, we use epipolar volumes instead of plain images, thus the generator and discriminator networks use 3D convolutions. Second, we propose an additional input to the WGAN. Besides the generated patch, we also feed the discriminator with the angular and spatial derivatives. We believe that for the super-resolution task, the high-frequency components are very helpful to obtain good quality and aesthetically pleasing results. In previous work [78], it was shown that GANs help with enhancing details in the high-resolution version of the image; however, it might create some unwanted distortions and hallucinate details that are not present in the original image. Because in theory the light field has more information about the scene due to its sub-aperture views, we want to focus the attention of the discriminator on the sharpness of the reconstructed details.

For any light field $L'_h$, our discriminator $G(L'_h)$ actually takes as additional input all of the derivatives of $L'_h$ explicitly; that is,

$$G(L'_h) = G(L'_h, \partial_s L'_h, \partial_t L'_h, \partial_y L'_h, \partial_x L'_h).$$

(4.26)

We omit this explicit dependency in the following to avoid cluttering the notation. The discriminator loss is built so that the target output for generated light fields is one, while the target output for real light fields is zero. Thus, for a single low-/high-resolution training pair $(L_l, L_h)$, the discriminator loss becomes

$$E_{\text{wgan}}(D) = \langle D(L_h) - D(G(L_l)) \rangle.$$  (4.27)

By $\langle \cdot \rangle$ we denote the mean value. In addition, the generator $G$ models the outputs to ensure that they are similar to ground truth according to the discriminator $D$. The generator tries to take the output of the
discriminator to zero, and thus minimizes
\[ E_{\text{wgan}}(G) = D(G(L_l)). \] (4.28)

Note that in the actual implementation, the WGAN losses are split up into a sum of contributions for horizontal and vertical epipolar volumes. Minimization steps for discriminator and generator are performed in an alternating fashion.

**Loss functions of our network**

As the main loss function of the network, we use \( L^2 \)-loss between the network’s output and the high-resolution ground truth. We have found that WGAN training becomes more stable if we introduce a weighted pixel-wise \( L^2 \)-loss. The weight is chosen according to the brightness of the pixel in the ground truth epipolar volume. It penalizes dark regions more because dark regions are usually more noisy and the \( L^2 \)-loss is less sensitive. To further enhance the detail of the output, we also calculate the \( L^2 \)-loss between derivatives of the output and the ground truth. Again, for a single training light field, the total reconstruction loss from directly comparing the generated to the ground truth high resolution light field reads
\[ E_{\text{rec}}(G) = (1 - \exp(-L_h/0.5)) \| G(L_l) - L_h \| + \| \nabla G(L_l) - \nabla L_h \|, \] (4.29)

where the gradient is computed spatially.

To improve the quality of the output and to speed up the training process, we additionally employ a perceptual loss [68]. For this, we pass the output and the ground truth through the convolutional layers of a pre-trained Inception-v3 network [113] and accumulate the \( L^2 \)-loss of the features in all the layers to compute \( E_{\text{perceptual}}(G) \). Finally, we add the loss of the DiffWGAN for both vertical and horizontal stacks to our generator network, as described in the previous subsection. The total loss of the generator network for a single light field becomes
\[ E_{\text{total}}(G) = E_{\text{rec}}(G) + E_{\text{perceptual}}(G) + E_{\text{wgan}}(G). \] (4.30)

In the next section, we detail the architecture of the subnetworks, which we follow with a detailed explanation of the training data and the strategy that we use to minimize the loss and prediction error.

**4.8.3 Detailed network architecture**

The light field has a very rigid structure that is linked to the scene geometry, which gives rich and redundant information about how to precisely match sub-aperture views, as required for super-resolution. To efficiently take full advantage of this structure, we propose a network architecture with a tailored 4D autoencoder, which is shown in Fig. 4.36. The inputs to our network are patch-wise vertical and horizontal epipolar volumes from the low-resolution light field ”crosshair”, a cross-shaped subset of views around a reference view. We can significantly reduce computational cost by using the autoencoder structure and the patch-wise input instead of the whole image. The vertical volume consists of the top and bottom views and the center view which are marked by red frames in the left-hand side of Fig. 4.36, while the horizontal volume contains the left- and right-most views, and also the center view, which are all framed in green. The high-resolution crosshair encompassing all views is fed as the ground truth.

**Architecture of the encoder**

The architecture of the encoder is based on the general model described in Section 4.2. Each epipolar volume has a spatial resolution of \( 48 \times 48 \). The horizontal volume is spatially transposed so that the images exhibit the same view point motion as the vertical patches. Consequently, the vertical and the horizontal volumes can share the convolution kernels in the whole network. The 4D encoder has nine
Chapter 4 Deep Autoencoder for Light Fields

Figure 4.37: Residual blocks in the encoder and decoder. Left: The residual block of the encoder performs batch normalization, $3 \times 3 \times 3$ convolution and a leaky ReLU with $\alpha = 0.2$. For the residual connection, the input features will be transformed with a $1 \times 1 \times 1$ convolution, if necessary with stride, to achieve the correct output size before addition. Right: The residual block of the 2D decoder also employs batch normalization. Bicubic interpolation will upscale the features, followed by a $3 \times 3$ deconvolution with stride 1 and "VALID" padding to avoid the fabricated pixels on the border of the image. Likewise, if the input features in this block are rescaled spatially by the left chain, then they are convolved with a $1 \times 1$ kernel with respective stride to allow addition.

layers, applying the residual block [52] to the features. This block is detailed in the left-hand part of Fig. 4.37. The odd layers gradually increase the number of features, while the even layers downscale the features spatially with stride-2 convolution in the residual block. In the latent space, the vertical and horizontal volumes are downscaled from $3 \times 48 \times 48$ finally to $3 \times 3 \times 3$.

Architecture of the decoder

The first part of the 4D decoder has nine layers and generates a light field that is spatially the same size but has already increased angular resolution. The features are spatially upsampled in nine decoding pathways, each pathway decodes one sub-aperture view. The right side of Fig. 4.37 shows the structure of the decoding residual blocks. The detailed version is presented in Fig. 4.38. To spatially upscale the features, we grow the spatial size of them by the means of bicubic interpolation and apply unstrided transpose convolution to avoid the checkerboard artifacts in the spatial domain caused by strides [105]. At this point, the features are angularly super-resolved to the target number of nine views, both in the vertical and horizontal directions.

The features of each layer in the encoder have only three vertical or horizontal views. Consequently, to prepare the skip connections for the decoder, we simply concatenate the encoded volume alone the last dimension to obtain the 2D features instead of 3D volumes. In addition, we leave twice as many features in the decoded volume as in the skip connection volume; otherwise, the skip connection volume will dominate the decoding process.

Spatial upscaling part of the decoder

On top of the output with increased angular resolution, we stack a spatial upscaling network. This network increases the spatial resolution of the features—first to twice the size of the input, then to four times—to obtain the super-resolved output. All of the spatial upscaling operations are carried out by a decoder residual block, see Fig. 4.38. In the second scaling phase, the features are spatially upsampled by the decoding residual block once and then subsequently passed through two decoding residual blocks, without their spatial resolution being changed. Afterwards, $1 \times 1 \times 1$ convolution is applied to the volumes to finally bring the number of features to the same as the input. The upscaling phase for scale factor 4 follows the same procedure, with its input coming from the intermediate results of scale factor 2 just before its $1 \times 1 \times 1$ convolution; see Fig. 4.36.

93
Chapter 4 Deep Autoencoder for Light Fields

Figure 4.38: Residual blocks in detail. Top: Left chain of the residual block in the 2D decoder. The input features are first batch-normalized. Bicubic interpolation will be applied to the features for the spatial upsampling. Afterwards, the features pass through a $3 \times 3$ transpose convolution. We add two pixels in the spatial output size and apply stride-1 with “VALID” padding. To obtain the final output of this block, we discard again one pixel from each side of the features. Bottom: Left chain of the residual block in the upscaling phase. The input features are first batch-normalized. After the bicubic interpolation, the features pass through a $3 \times 3 \times 3$ transpose convolution. We add two pixels in the spatial and two views in the angular output size and apply stride-1 with “VALID” padding. To obtain the final output of this block, we discard again one pixel from each side of the features and the additional views generated at the boundaries.

Architecture of the WGAN discriminator

To enhance the details of the output, we add the DiffWGAN discriminator on top of our network. The discriminator input consists of a stack of concatenated light fields together with their derivatives, as previously detailed in Section 4.8.2. The structure is similar to the encoder but we combine two subsequent downsampling and feature expansion layers into a single layer, making the network more shallow. In fact, the discriminator has a much easier task than the encoder, so it is reasonable to substantially reduce capacity. The discriminator distinguishes the super-resolved output from the ground truth using a Wasserstein distance on top of the features of the modified encoder chain.

4.8.4 Training strategy

The training data is generated using the Blender add-on provided with [60] and follows the same procedure as in the previous network examples for generating random light fields from a number of template scenes, objects and textures. We generated a total of 750 of these random light fields for training. Although placing random textures on objects leads to unrealistic objects that are usually not encountered in the real-world, we believe that this is not a drawback for small patches and for the task of super-resolution because the network will probably not learn at object level. Resolution and disparity range were chosen to allow us to downscale the spatial component of the light fields by a factor of 2 and 4, respectively, while still keeping the disparities readily distinguishable. In addition, we use publicly available data from the HCI database [134], the Stanford multi-camera array [126], and light fields captured with the Lytro Illum plenoptic camera.
We randomized the process of generating scenes to ensure a large variety of different light fields for training. We created template scenes with up to 13 placeholders, to ensure that scenes have varying complexity and different scales of geometry. Each placeholder is replaced by one of 42 objects, which are rotated at random. The texture, reflective properties and the environment map for lighting are chosen at random. High resolution textures are used to ensure that no interpolation artifacts are learned with the training data.

In our approach, we chose the YCbCr color space, where the luminance component $Y$ contains most of the spatial detail of the image. The channels $Cb$ and $Cr$ encode the blue-difference and red-difference chroma components, which are typically of low frequency. Therefore, we can reduce the memory requirements drastically, by following Timofte et al. [124] and Yoon et al. [143] in their observation that the YCbCr color space is proven to lead to the best results.

Our training was performed on an Intel Core i9 with 128 GB of RAM and 4 nVidia TITAN Xp GPUs. The optimization is performed with the Adam optimizer, with initial learning rate set to $1e^{-4}$, batch size is 5. The WGAN turned out to be easy to train and will easily dominate the training [8] to produce artifacts, so we assign a small weight of $1e^{-3}$ to the DiffWGAN loss to keep it balanced with respect to the other loss components. To satisfy the Lipschitz constraint, we keep the discriminator weights in $[-5e^{-3}, 5e^{-3}]$ range. The loss becomes stable after approximately 10 hours, after which we dropped the learning rate to $1e^{-5}$ and trained for another 8 hours. We then stopped training because the loss did not significantly decrease any further.

### 4.8.5 Experiments

Because our network is trained with patch-wise data, the output patches are reassembled to the complete high-resolution $Y$ channel of the image. We compute PSNR and SSIM after the images are converted back to RGB color space. We evaluate our network both on publicly available light field datasets, which
Figure 4.40: Visualization and quantitative evaluation of the results of scale factor 4. The images are super-resolved center views of Benchmark Cotton, HCI Maria and real-world data Hedgehog.

are synthetically rendered or captured with a gantry [60, 126, 134], as well as the data that we captured with a Lytro Illum plenoptic camera. None of the datasets that we use in our evaluation has been seen during training.

Our approach super-resolves the light fields angularly from three views to nine views and spatially, with scale factor 2 and scale factor 4. For comparison, we picked the recent works for single image SRGAN [78] with scale factor 4, the variational approach VarSR [96], which is improved upon [133], and the graph-based method GB-SQ [99] with spatial scale factor 2.

In Fig. 4.39 and in Fig. 4.40, we visualize the super-resolved center views of scale factor 2 and scale factor 4 of the Benchmark Cotton, HCI Maria and real-world data Hedgehog. The PSNR and SSIM of each approach are reported below the images. One can observe over-smoothing of VarSR [96] in the scale factor 2 results, while some severe high-frequency noise appears in their scale factor 4 results. Bilinear interpolation provides rapid single image super-resolution with acceptable PSNR and SSIM, but for scale factor 4 it is significantly more blurry than the other methods. The graph-based approach GB-SQ [99] gives brilliant results in HCI Maria and real-world data Hedgehog, and is also visually very sharp in Benchmark Cotton. However, its computational time means that it is next to inapplicable in practice. In our experiments, it took around 8 hours to compute a $9 \times 9$ light field for scale factor 2, compared to two to three minutes on average using our approach. SRGAN seems to sometimes hallucinate additional structure; for example, in the zoomed-in area of the Hedgehog, the fur around the nose is sharp but apparently does not have the same shape as the ground truth. Because we use only few views to reconstruct the whole light field, our results for scale factor x4 are generally worse that state-of-the-art, but scale factor x2 reach competing quality and even outperform the other methods in some cases. Complete numerical results can be observed in Tables 4.1 and 4.2. Note that the other methods use all sub-aperture views as an input, thus they perform only spatial super-resolution. For a fair comparison, we only compute PSNR and SSIM for those views that were in the input, thus we evaluate quality of the spatial super-resolution. Additionally we illustrate numerical results for all views (newly generated and present in the input). As an ablation study, we show the results of training without adversarial loss.
### Table 4.1: Comparison of PSNR (SSIM) for different methods, for super-resolution scale factor two and over a wide range of datasets from various sources. Note that we could not compute some results for GB-SQ (marked with an asterisk) due to excessive run-time, so we took them from their paper.

<table>
<thead>
<tr>
<th>Lightfield</th>
<th>scale factor $\alpha = 2$</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bilinear</td>
<td>GB-SQ [99]</td>
<td>VarSR [96]</td>
<td>Our Spatial</td>
<td>Our Full</td>
<td>No WGAN</td>
</tr>
<tr>
<td>antinous</td>
<td>28.43 (0.95)</td>
<td>27.83 (0.96)</td>
<td>27.29 (0.94)</td>
<td>39.70 (0.97)</td>
<td>37.07 (0.95)</td>
<td>32.27 (0.95)</td>
</tr>
<tr>
<td>bicycle</td>
<td>27.36 (0.83)</td>
<td>30.73 (0.90)</td>
<td>28.58 (0.86)</td>
<td>27.04 (0.86)</td>
<td>26.32 (0.80)</td>
<td>25.95 (0.81)</td>
</tr>
<tr>
<td>tomb</td>
<td>29.59 (0.91)</td>
<td>31.81 (0.94)</td>
<td>30.65 (0.91)</td>
<td>37.27 (0.91)</td>
<td>36.15 (0.87)</td>
<td>31.61 (0.86)</td>
</tr>
<tr>
<td>bedroom</td>
<td>27.21 (0.86)</td>
<td>26.66 (0.91)</td>
<td>26.73 (0.87)</td>
<td>32.06 (0.88)</td>
<td>30.97 (0.84)</td>
<td>30.03 (0.84)</td>
</tr>
<tr>
<td>herbs</td>
<td>30.61 (0.83)</td>
<td>33.61 (0.90)</td>
<td>31.45 (0.86)</td>
<td>30.16 (0.86)</td>
<td>28.23 (0.75)</td>
<td>27.69 (0.76)</td>
</tr>
<tr>
<td>cotton</td>
<td>27.33 (0.95)</td>
<td>27.87 (0.96)</td>
<td>27.45 (0.95)</td>
<td>40.68 (0.97)</td>
<td>39.64 (0.96)</td>
<td>33.14 (0.95)</td>
</tr>
<tr>
<td>platonic</td>
<td>33.39 (0.90)</td>
<td>38.42 (0.96)</td>
<td>34.53 (0.92)</td>
<td>34.13 (0.92)</td>
<td>32.2 (0.84)</td>
<td>29.84 (0.82)</td>
</tr>
<tr>
<td>rosemary</td>
<td>32.24 (0.94)</td>
<td>37.26 (0.98)</td>
<td>33.75 (0.96)</td>
<td>32.25 (0.95)</td>
<td>30.74 (0.92)</td>
<td>29.52 (0.92)</td>
</tr>
<tr>
<td>maria</td>
<td>30.05 (0.86)</td>
<td>37.25 (*)</td>
<td>32.78 (0.91)</td>
<td>33.23 (0.91)</td>
<td>32.61 (0.90)</td>
<td>30.69 (0.88)</td>
</tr>
<tr>
<td>owl2</td>
<td>36.21 (0.97)</td>
<td>41.04 (0.98)</td>
<td>37.93 (0.97)</td>
<td>35.76 (0.95)</td>
<td>35.12 (0.94)</td>
<td>30.08 (0.87)</td>
</tr>
<tr>
<td>flowers</td>
<td>34.23 (0.96)</td>
<td>36.98 (0.98)</td>
<td>36.03 (0.97)</td>
<td>34.33 (0.91)</td>
<td>34.27 (0.94)</td>
<td>29.25 (0.83)</td>
</tr>
<tr>
<td>owl-str</td>
<td>32.14 (0.94)</td>
<td>36.46 (0.97)</td>
<td>32.86 (0.95)</td>
<td>32.65 (0.95)</td>
<td>31.02 (0.90)</td>
<td>29.36 (0.91)</td>
</tr>
<tr>
<td>origami</td>
<td>28.90 (0.93)</td>
<td>32.03 (0.95)</td>
<td>29.86 (0.94)</td>
<td>29.65 (0.95)</td>
<td>29.34 (0.94)</td>
<td>27.74 (0.91)</td>
</tr>
<tr>
<td>hedgehog</td>
<td>34.41 (0.95)</td>
<td>39.07 (0.98)</td>
<td>34.98 (0.95)</td>
<td>33.61 (0.95)</td>
<td>32.27 (0.92)</td>
<td>29.39 (0.88)</td>
</tr>
<tr>
<td>eucalyptus</td>
<td>33.98 (0.96)</td>
<td>39.09 (*)</td>
<td>34.93 (0.95)</td>
<td>33.35 (0.88)</td>
<td>29.62 (0.84)</td>
<td>27.04 (0.67)</td>
</tr>
<tr>
<td>truck</td>
<td>36.06 (0.96)</td>
<td>41.57 (*)</td>
<td>36.94 (0.96)</td>
<td>35.47 (0.95)</td>
<td>32.45 (0.90)</td>
<td>29.18 (0.86)</td>
</tr>
</tbody>
</table>

In addition we show qualitative results of our method on a synthetic and two real world scenes taken with Lytro Illum plenoptic camera, see Figs. 4.41, 4.42, 4.43. In addition, we illustrate angular consistency of the results by showing some EPIs taken at random from the output light fields 4.44. For each dataset, we show the ground truth qualitative evaluations for scale factors x2 and x4. To obtain the scale factor x2, we downscale the original light field twice and then run our network to obtain the result with magnification factor x2. We then apply a similar procedure to compute light fields with magnification factor x4. For each dataset, we show horizontal and vertical EPIs for magnifications factors x2 and x4. Because light fields from benchmarks and those captured with the Lytro Illum plenoptic camera have a very small baseline, the disparity range is quite small—usually within $[-2, 2]$. Downscaling such a light field by four times might result in discarding the epipolar structure. Thus, all sub-aperture views will look the same with only miniscule view point change. One of the particular use cases of light field super resolution is to upscale noisy examples captured with the Lytro Illumm. We show the magnification factor x4 of the light field that was downscaled only two times. Thus, we do not have a ground truth comparisons in this showcase but we present the real application of our method.

### 4.8.6 Conclusions

We present an efficient approach to spatial and angular light field super-resolution based on an encoder-decoder architecture with a novel WGAN loss. Our proposed method adopts insights from both 2D [78, 124] and 3D [34, 143] CNN-based approaches to arrive at an architecture with very competitive performance. We demonstrate this in numerous experiments on public datasets [126, 134], as well as on real world light fields captured with a Lytro Illum plenoptic camera. Our architecture is simple and powerful in the sense that it fully utilizes the rich information inherited from the light field and proves that even very few sub-aperture views are sufficient to reconstruct a high-resolution dense light field of a good quality. Competing state-of-the-art light field super-resolution algorithms require many more input views.
Table 4.2: Comparison of PSNR (SSIM) for different methods, for super-resolution scale factor four and over the datasets presented in the table 4.1.

<table>
<thead>
<tr>
<th>Lightfield</th>
<th>Bilinear</th>
<th>GB-SQ [99]</th>
<th>VarSR [96]</th>
<th>Our Spatial</th>
<th>Our Full</th>
<th>No WGAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>antinous</td>
<td>28.03 (0.92)</td>
<td>33.81 (0.91)</td>
<td>26.86 (0.88)</td>
<td>25.75 (0.92)</td>
<td>25.62 (0.91)</td>
<td>24.33 (0.90)</td>
</tr>
<tr>
<td>bicycle</td>
<td>23.68 (0.63)</td>
<td>23.82 (0.63)</td>
<td>24.06 (0.67)</td>
<td>20.38 (0.61)</td>
<td>20.28 (0.59)</td>
<td>20.80 (0.60)</td>
</tr>
<tr>
<td>tomb</td>
<td>29.22 (0.83)</td>
<td>30.62 (0.67)</td>
<td>28.45 (0.75)</td>
<td>28.02 (0.79)</td>
<td>27.82 (0.78)</td>
<td>24.96 (0.76)</td>
</tr>
<tr>
<td>bedroom</td>
<td>26.02 (0.74)</td>
<td>28.47 (0.72)</td>
<td>24.76 (0.68)</td>
<td>25.41 (0.69)</td>
<td>25.38 (0.68)</td>
<td>24.38 (0.68)</td>
</tr>
<tr>
<td>herbs</td>
<td>27.28 (0.69)</td>
<td>26.80 (0.67)</td>
<td>27.41 (0.72)</td>
<td>21.08 (0.66)</td>
<td>21.01 (0.63)</td>
<td>21.46 (0.64)</td>
</tr>
<tr>
<td>cotton</td>
<td>27.34 (0.92)</td>
<td>34.83 (0.90)</td>
<td>26.91 (0.89)</td>
<td>27.47 (0.91)</td>
<td>27.35 (0.90)</td>
<td>25.60 (0.89)</td>
</tr>
<tr>
<td>platonic</td>
<td>29.07 (0.72)</td>
<td>27.38 (0.63)</td>
<td>29.63 (0.77)</td>
<td>23.60 (0.61)</td>
<td>23.52 (0.59)</td>
<td>21.09 (0.57)</td>
</tr>
<tr>
<td>rosemary</td>
<td>27.59 (0.84)</td>
<td>27.58 (0.85)</td>
<td>27.64 (0.86)</td>
<td>22.60 (0.80)</td>
<td>22.47 (0.78)</td>
<td>22.30 (0.79)</td>
</tr>
<tr>
<td>maria</td>
<td>26.45 (0.69)</td>
<td>26.08 (0.65)</td>
<td>27.55 (0.73)</td>
<td>24.64 (0.68)</td>
<td>24.56 (0.66)</td>
<td>23.62 (0.67)</td>
</tr>
<tr>
<td>owl2</td>
<td>29.74 (0.89)</td>
<td>29.26 (0.87)</td>
<td>30.68 (0.90)</td>
<td>22.12 (0.71)</td>
<td>22.02 (0.70)</td>
<td>19.69 (0.66)</td>
</tr>
<tr>
<td>flowers</td>
<td>28.64 (0.87)</td>
<td>29.51 (0.85)</td>
<td>29.54 (0.89)</td>
<td>20.72 (0.58)</td>
<td>20.61 (0.56)</td>
<td>18.09 (0.51)</td>
</tr>
<tr>
<td>owl-str</td>
<td>26.48 (0.79)</td>
<td>25.35 (0.75)</td>
<td>26.66 (0.81)</td>
<td>22.46 (0.75)</td>
<td>22.24 (0.72)</td>
<td>20.96 (0.72)</td>
</tr>
<tr>
<td>origami</td>
<td>24.09 (0.78)</td>
<td>22.27 (0.76)</td>
<td>24.18 (0.76)</td>
<td>20.87 (0.71)</td>
<td>20.81 (0.70)</td>
<td>19.47 (0.68)</td>
</tr>
<tr>
<td>hedgehog</td>
<td>28.31 (0.81)</td>
<td>26.24 (0.76)</td>
<td>28.10 (0.82)</td>
<td>20.96 (0.68)</td>
<td>20.84 (0.65)</td>
<td>19.88 (0.64)</td>
</tr>
<tr>
<td>eucalyptus</td>
<td>30.35 (0.90)</td>
<td>30.48 (0.84)</td>
<td>29.87 (0.86)</td>
<td>19.97 (0.43)</td>
<td>19.8 (0.42)</td>
<td>18.64 (0.37)</td>
</tr>
<tr>
<td>truck</td>
<td>31.52 (0.90)</td>
<td>31.34 (0.87)</td>
<td>27.99 (0.75)</td>
<td>22.71 (0.74)</td>
<td>22.61 (0.72)</td>
<td>19.93 (0.68)</td>
</tr>
</tbody>
</table>

and/or take a lot more time to compute. In addition we propose the new DiffWGAN loss that improves the visual quality of the results. From the experiments, we show that our method does not hallucinate missing information compared to the original SRGAN network.
Figure 4.41: Real world dataset owl2. The upper part shows the results for magnification factor x2 (left) and magnification factor x4 (right). The bottom part shows interpolated input and our results of upscaling with magnification factor x4. This example illustrates the possible application of our network, where light field is lacking spatial and angular resolution.
Figure 4.42: Synthetic dataset tomb from the light field benchmark [60]. The light field resolution is $9 \times 9 \times 512 \times 512 \times 3$. We show the whole images with two zoomed-in parts to illustrate the quality of our results. We observe very sharp results for the magnification factor x2 and quite blurry output for the magnification factor x4. This can be explained by the fact that if we downscale light field with very small baseline 4 times, then we loose epipolar structure, thus the network cannot find any additional information in the angular domain.
Chapter 4 Deep Autoencoder for Light Fields

Figure 4.43: Real world dataset koala. By this example we want to show again how our method can be used in the real-world applications. The bottom part shows the light field upscaled with the magnification factor $x_4$ from its twice downscaled version. In the zoomed-in parts we illustrate how our network copes with fine structure like fur.

Figure 4.44: Example of horizontal and vertical EPIs for magnification factors $x_2$ and $x_4$. The left column illustrates the ground truth and the right column shows our results. The first row shows EPIs from koala dataset 4.43, second row illustrates tomb dataset 4.42, and the last row presents owl2 light field 4.41. We observe that the epipolar structure is well preserved for the magnification factor $x_2$ light fields. For the light fields with magnification factor $x_4$ we see that in some regions the epipolar structure is broken due to the lack of information in the input light field.
4.9 Summary

In this chapter, we introduce the first convolutional encoder-decoder network for light fields that is suitable for multi-task learning. In this work, we demonstrate the application of this architecture to three state-of-the-art computer vision problems. The first example jointly solves disparity estimation and reflection separation tasks, see Section 4.6. The second example performs full intrinsic light field decomposition and disparity estimation, described in Section 4.7. The last application is light field super-resolution in angular and spatial domains with magnification factors x2 and x4, see Section 4.8. The key idea is to use 3D convolutions to compute features integrated over the whole range of both vertical and horizontal epipolar volumes. Later on, we upsample those features into separate decoding pathways: disparity, reconstructed input, diffuse and specular, albedo and shading, high-resolution light field.

Because the accurate ground truth is not easy to get, for instance for the real-world scenes, our network performs both supervised and unsupervised learning. For unsupervised training, we use a physical model of image formation to ensure that intrinsic components sum up to the input light field. According to the previous research, diffuse components and disparity correspond to the same projections of the same 3D point, thus they share the same pattern. However, the specular component behaves differently because it follows the specular flow, which depends on the local surface geometry and view point change. Thus, we assume that different tasks can benefit from each other and we train them together in one network.

We propose a mirror-like architecture that consist of several convolutional layers followed up by up-sampling layers for each decoding pathway. Each layer is represented by the residual block, which consist of batch normalization, convolution and activation layers. The light field itself, reflection and intrinsic components are modeled as 3D decoders, while disparity is 2D. In the last up-sampling layer, intrinsic decoders share the features. If ground truth for the input patch is available, then the network is trained in the supervised manner; otherwise, only the computational cost that comes from the dichromatic model is used.

We suggest a novel adversarial loss function for light field super-resolution. Compared to the original GAN loss, it takes into account the derivative information together with the light field itself. This design choice brings network attention to sharpness of details in the output light fields. Thus, our network is able to perform together angular and spatial super-resolution and achieve a competing quality of the results.

To train the network, we generate light fields with ground truth disparity and intrinsic components in Blender software with Cycles rendering engine. In addition to the generated examples, we use various real-world and synthetic benchmark datasets [60, 126, 134].

With extensive experiments, we show that the outputs of our networks outperform or are on par with state-of-the-art variational methods and recent deep learning approaches, both visually and numerically.
This work covers several research problems in computer vision and particularly in light fields, which are closely related to each other. We start with disparity estimation, which is a well-established application of light fields. Numerous algorithms have been proposed to compute disparity labels, but few of them evaluate its quality for some applications where geometrical information is needed.

We design a disparity estimation algorithm that is particularly suitable for modeling intrinsic components by jointly optimizing disparity labels and normal maps. The resulting disparity map is used to reconstruct a 3D representation of a scene, which in turn becomes a basis for modeling intrinsic components. We propose a variational intrinsic light field decomposition mode; where, to leverage light field structure, we enforce albedo and shading to stay constant along the disparity directions. To identify specular surfaces, we use the property of a highlight to be view-dependent. We analyze the projections of a 3D point in sub-aperture views and judge its type of reflection.

We improve our variational model by introducing better prior for shading component by further breaking down it into direct and indirect contributions. Direct shading is modeled based on scene geometry, while indirect shading that captures cast shadows and inter-reflections is designed based on geometrical and color information. We illustrate the advantage of the new model on a wide range of natural and synthetic light fields.

With the growing popularity of deep learning methods, we develop a neural network architecture that is particularly designed for light fields. To exploit the rich but redundant light field structure, we design an encoder-decoder network where we extract a small set of features from angular and spatial dimensions, and then use these features to solve different tasks simultaneously. Under the assumption that these tasks are closely related to each other, we upsample the extracted set of features into separate decoding pathways. Because it is quite challenging to get a ground truth intrinsic components or disparity for the real world scenes, we designed our network such that it can be trained both supervised and unsupervised. If the ground truth is available (for synthetic scenes), then the standard loss function penalizes the difference between the output of the network and ground truth example. For the natural scenes, without ground truth, the network ensures that the output components satisfy the underlying physical model of image formation. As a proof of concept, we designed several multi-task networks: disparity estimation and reflection separation, intrinsic images and disparity, and super-resolution with magnification factors x2 and x4.

However the proposed architecture shows promising results, there is still space for improvement. Currently, we incorporate the simplest model in the network architecture, where we penalize the intrinsic components to sum up to the input light field. However, there are more complicated and accurate models [29, 93] of image formation in the literature that could be used to improve the quality of decoded components. In the variational formulation of intrinsic light fields, we introduce several priors for the intrinsic components. We believe that those priors can be incorporated into network architecture as an additional loss function. Including this type of guidance will make network explainable and less dependent on the quality of ground truth data. In theory, if the network uses only costs that are derived for the variational model, then it can be trained completely without supervision.
Another interesting continuation of this work is to analyze the limits of the proposed architecture. Given sufficient computation power, one can explore how many tasks can be performed with a single network. Theoretically, all of the problems that have been discussed can be combined in one large encoder-decoder network. This experiment would show the advantage of using light fields over other data types for a large number of state-of-the-art computer vision problems.

One more research direction is to study the "redundancy" of the bottleneck. In all experiments, we used the whole set of features for every decoder. However, a particular subset may be able to do a similar or even better job. Another question that is closely related to the discussed tasks, especially to super-resolution and disparity estimation, is view synthesis—where a new view is created from a small subset of input light field. One could approach this with a LSTM network combined with the proposed autoencoder.

The combination of deep learning and modeling approaches open a new direction in light field research, where the ability of neural networks to approximate complex physical phenomena strengthens with mathematical modeling, which brings the whole concept towards explainable AI. Rich light field structure gives rise to 3D or 4D neural networks, where spatial and angular information is taken into account. With this work, we make a first step towards understanding the power of light fields for non-Lambertian scene analysis. Although our algorithms are specifically designed for light fields, they can be extended to many kinds of multi-view data and can applied in industrial applications, such as in the oil and gas industry, or for the analysis of 3D tomography images of rocks [51].
Bibliography


