Abstract—Feature detection is the basis for much of computer vision. Traditionally, this field is still dominated by handcrafted detection algorithms. These algorithms work reliably on two-dimensional images, but generalizations to light field images have not been explored in detail yet. In this work, we present a learning approach for feature detection. We show that our method, using a fully convolutional neural network, is feasible for two-dimensional images and generalizes to the higher dimensional light fields.

I. INTRODUCTION

The ability to detect features in images quickly and reliably is necessary for a wide range of applications, including object recognition, change detection and 3D reconstruction. This project will focus on a learning-based approach to detecting features in order to obtain structure from motion (SfM).

There are challenges that are not addressed by commonly used feature extraction methods such as SIFT [1]. For example, SIFT cannot distinguish between Lambertian and reflective or refractive surfaces. This is a problem for 3D-reconstruction methods that use matches acquired from, e.g., SIFT, because matches resulting from non-Lambertian surfaces confuse geometric consistency check algorithms and are thus not suitable to infer structure.

In contrast to normal images, light fields are four-dimensional and contain richer information about the captured scene. Parallax in different views enables depth estimation, and a single light field can be sufficient to fully reconstruct a 3D model of the scene [2], [3]. Furthermore, light fields can be used to detect whether or not a surface is Lambertian [4]. Figure 1 shows the difference between refractive and Lambertian surfaces in epipolar plane images (EPIs).

For a given row in the light field sensor image, the EPIs show how the light field changes for different horizontal viewpoints. For Lambertian surfaces, the slope in the EPIs is proportional to the depth of the scene. This concept breaks for non-Lambertian surfaces, as can be seen in the second EPI. The crystal ball, a highly refractive object, leads to curved lines in the EPI.

Using this information, it is possible to use hand-crafted algorithms to remove features that have been detected on the crystal ball (Figure 2). With “hand-crafted”, we refer to algorithms that do not make use of learning techniques. In this work, a first step towards the direct detection of “good” features using a fully convolutional neural network (CNN) is presented. By designing a convolutional light field autoencoder and comparing the results from 2D and 3D convolutions, we show that convolutions can pick up on additional information in the light field in higher dimensions. We then build a CNN to detect SIFT features on 2D input images. We show that the model architecture can be adapted to work with higher-dimensional inputs.

II. RELATED WORK

Several research groups have used learning approaches for feature detection before. [5] and [6] used fully convolutional neural networks for their feature detectors. Both performed feature detection on 2D images, obtaining their ground truth from a multi-view geometric consistency check. Similar to their approach, we propose using COLMAP [7] for the geometric consistency check. Results of this will be shown in follow-up work. Other methods do not rely on multiple views of one scene, but instead use a self-supervised approach to perform the geometric consistency check [8].
Research on feature detection has been performed on light fields as well. In [9], a hand-crafted method for 3D feature extraction on light fields – with the third dimension being depth – is shown. Our work also introduces a model that extends to the third dimension of a light field, but unlike [9], we use one row of horizontal sub-aperture images as the third dimension. We detect features only in the center view of a light field, but use parallax information from surrounding views. While our model is trained exclusively on keypoints extracted from 2D rendered images using SIFT and Harris algorithms, we hope that the 3D architecture will be able to use parallax information to remove features resulting from occlusion boundaries or reflections when trained on more meaningful training data.

III. OBJECTIVE AND SCOPE

The main objective of this project is to motivate the use of 4D light fields in computer vision by developing a convolutional neural network for learning feature detection.

To accomplish this, we first classify a set of images into “good” patches based on features detected using SIFT and Harris, and “bad” patches sampled randomly from the image. (Further refinement of the good patches could be done using COLMAP 3D-scene matching and rejecting any unmatched features.) Once the patches are labeled, they are used to train a 2D CNN to classify and detect good features in 2D images. We then extend the model to train a 3D CNN for classification of 3D light field slices to show the robustness of our model.

While beyond the scope of the project, logical next steps would be to create a training dataset that benefits from using three or all four light field dimensions. Once feature detection is successful with light fields and show superior results compared to feature detection on 2D images, this work can be extended to feature description and matching.

IV. DATASET

The dataset contains 4251 light fields of indoor and outdoor scenes in 31 different categories. Each scene is captured from 3-6 different viewpoints. The complete dataset has a size of 212 GB. Each light field is composed of a $541 \times 376$ microlens array containing $14 \times 14$ pixels per microlens image. Additionally, the dataset contains one rendered high resolution image ($2022 \times 1404$) per light field. In the following, we refer to the different sub-aperture images with coordinates $s$ and $t$, and to the position within a sub-aperture image with coordinates $u$ and $v$.

V. PROOF OF CONCEPT: LIGHT FIELD INFORMATION

The first step is a proof of concept that there is relevant information in the $t$ and $s$ dimensions of a light field. To do this, two convolutional autoencoders have been developed and trained on images without noise. One autoencoder encodes 2D light field slices and the other one uses 3D light field slices, where $s$ is kept constant. The performance of both autoencoders has been measured by their ability to denoise an image, a task they were not trained for. The ability to denoise is therefore a measurement of how well the autoencoder can compress images to a subset of its most important features. The 3D autoencoder performs better on the reconstruction of a 3D light field slice than on a 2D slice. This is due to the fact that the 3D slice contains more information that can be used to reduce noise, (e.g., redundancy in the third dimension). Figure 3 shows a qualitative comparison of the denoising performance of the two autoencoders.

Quantitative results are shown in Figure 4.

![Fig. 4. Structural Similarity (SSIM) of the images shown in Figure 3 compared to the original image.](image)

The plot shows the structural similarity (SSIM) against the noise standard deviation of the additive Gaussian noise. The 3D autoencoder outperforms the 2D version, which is still superior to the naïve method of averaging 14 sub-aperture images, which reduces noise but instead introduces blur resulting from parallax between the views.

VI. TRAINING DATA ACQUISITION

We acquired training data by extracting keypoints from the rendered images in the dataset.

A. Train / Test Split

The images in the dataset has been split 80/20 into training and testing scenes. To make testing as fair as possible, different images of the same scene are contained in either the training or the testing subset, but never both.

B. Ground Truth Sampling

Good features were collected by extracting SIFT and Harris [10] features. For SIFT features, we used a constant peak threshold of 10 and edge threshold of 20 across all images in the dataset, which resulted in a selection of approximately 8.5 million examples. For Harris features, we additionally found the corresponding SIFT descriptor at a (heuristically chosen) scale of 2 for the top 500 keypoints. The descriptors will be useful once COLMAP is used for a geometric consistency check.

Bad features were collected by random selection. To detect bad features, a binary image was created with ones corresponding to detected feature locations. This image was dilated with a $21 \times 21$ pixel structuring element. Pixels not labeled were randomly chosen as 1 for the centers of bad
patches. The structuring shape of the dilation enforces a minimum (L1-)distance between the center of a good patch and the center of a bad patch.

Both Harris and SIFT features have been extracted on all images, but only SIFT features have been used to train the model in this report. We chose to evaluate these features first, because they seem to be more relevant for the task, and ran out of time before being able to test the algorithm on Harris corners as well.

C. Dataset Balancing

The dataset was balanced by acquiring the same number of good and bad features from each 2D rendered image. Furthermore, we kept the distribution of scales identical across the good and bad features by extracting one bad patch per good patch with the same scale as that of the good patch.

D. Geometric Consistency Check

Generation of good features can be improved using COLMAP, an open-source software used for matching features in a set of images that observe a particular scene from different angles. Considering that the goal of this project is to detect features in light fields, feeding in features filtered through COLMAP will enable the model to pick up on better features within a single light field. We used COLMAP on the 2D rendered images to illustrate the effect of feature matching. Figure 5 displays two images of the same scene from two varying angles superimposed with the matched features extracted using COLMAP.

![Feature comparison of two images in COLMAP](Image)

VII. CONVOLUTIONAL DETECTION MODEL

The detection model was created in analogy to the model presented in [6]. We simplified their approach by implementing only one branch of their multi-branch model. Thus, our model is not scale-invariant and can be trained on a single input scale only. The fully convolutional approach enables us to detect features of a single scale on input images of arbitrary size.

A. Model Architecture

We therefore developed a convolutional neural network to classify and detect features in 2D rendered images as well as in 2D and 3D light field slices. Figure 6 shows the model architecture.

Input patches are fed into two convolutional layers (C1 and C2), each followed by a pooling step (P1 and P2). Layer C1 is also followed by a normalization step. After the convolution and pooling steps, the tensors are fed into two convolutional layers (C3 and C4). These two layers have a kernel size of $1 \times 1$, and could be replaced with dense layers for classification. For detection, it is important to maintain a fully convolutional network, which is necessary for the input shape invariance of the model. For classification, the model outputs the probabilities for each class (good or bad). For detection on images that are larger than an input patch, the output of the convolutional layers (C3 and C4) forms a heatmap which can be further processed to locate features.

The architecture for patches from 2D and 3D light field slices is very similar, except for an extension of the convolutional kernels to the third dimension. Since the resolution of the light field slices is much lower than the resolution of the rendered images, we decided to adapt the model to use $16 \times 16$ slices (2D LF) or $16 \times 16 \times 12$ slices (3D LF) instead of $32 \times 32$ slices which have been used for the 2D model on rendered images. Table I below summarizes the details of the architecture.

B. Feature Detection from Heatmaps

Each pixel in a heatmap represents the probability of a good feature at the corresponding position in the input image. We performed a series of processing steps to extract local maximas that represent CNN-detected features from the heatmap. First, the heatmap was rescaled to input image size using linear interpolation. Next, pixels that represent a regional maxima (using 8-connected neighborhoods) were kept. Finally, the image was binarized using a chosen threshold value which acts as a quality control over detected features. The binarized image contains ones at detected feature positions and can be used to compare against SIFT features in the image. Figure 8 shows a rendered image with CNN and SIFT detected features overlaid.

![Fig. 5. Feature comparison of two images in COLMAP](Image)

3Evaluation script by Vincent Sitzmann
C. Training Details

The model has been trained on 1 million good and 1 million bad patches from the training data (due to memory limitations) and evaluated on 500,000 testing patches. Training was performed on a Titan X GPU in about 10 min per epoch. The model has been implemented in tensorflow and uses the Adam optimizer and binary cross-entropy loss:

\[ L(y, \hat{y}) = \sum_i y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i), \]

where \( y \in \{0, 1\} \) is the ground truth and \( \hat{y} \in [0, 1] \) is the model’s output probability for each training example.

D. Model Input Pipeline

The inputs to the CNN model are patches that are cropped around the location of a good or bad feature according to the keypoint location acquired during training data acquisition. These patches have been chosen to have the same size as the area that SIFT uses for description. The patch size is therefore dependent on the scale of the corresponding SIFT feature. Since the model is capable of handling one scale only, all patches are resized to a fixed size. For classification of patches extracted from 2D rendered images, the fixed size is \( 32 \times 32 \). This size corresponds to a SIFT scale of 2.67, which is the dominant value in the distribution of scales across all features in the dataset.

For the 2D LF model we chose the center view \( (s = t = 7) \) of a given light field, while for classification of 3D light field slices, we chose 12 of the 14 sub-aperture images \( (s = 7 \text{ and } t = 2, \ldots, 13) \). The keypoint coordinates (centers of the patches) have been translated from the rendered image size to the light field image size by scaling with an appropriate factor. In theory, this is only accurate for the center view of a light field, but the parallax within the images is small enough to make this a good approximation for more extreme views. All patches from the 2D and 3D light fields are resized to a fixed size of \( 16 \times 16 \) or \( 16 \times 16 \times 12 \) respectively, corresponding to a SIFT scale of 1.33. This is motivated by the smaller size of light field slices compared to rendered images.

The resized and labeled patches from the 2D rendered images and the light fields are fed into the model for training and testing our keypoint classifier. After classification, the previously learned weights were used to perform detection on full size input images.

VIII. EXPERIMENTAL RESULTS

Several experiments involving classification of input patches and detection of features in input images on the different models have been performed.

A. CNN Evaluation: Classification of Patches

First, we tested how well the model can classify patches obtained from the rendered images. We found that the 2D model does not need much training data to distinguish between good and bad patches, as can be seen from the learning curve shown in Figure 9. 10,000 patches are enough to achieve an accuracy of over 95% on the test set. As the
number of training examples increases, the training and test accuracy converges to a value of 98.6%. While this is a good result, it means that the training data is quite polarized, and good patches are easily distinguishable from bad patches.

The training and test accuracy for the 2D and 3D CNN show that our classification model is very robust (Table II) against resolution changes (2D LF) and input shape variation (3D LF). The model performs best on 2D rendered images because our training dataset is based on 2D SIFT features extracted from those images. The accuracy of classification of 3D light field slices is comparable to the classification of 2D light field slices and even to patches from the rendered images.

### TABLE II

<table>
<thead>
<tr>
<th>Model</th>
<th>Specifications</th>
<th>Train Acc.</th>
<th>Test Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D Class.</td>
<td>Hi-res 2D images</td>
<td>98.7 %</td>
<td>98.6 %</td>
</tr>
<tr>
<td>2D Class. LF</td>
<td>2D light field slices</td>
<td>96.0 %</td>
<td>94.7 %</td>
</tr>
<tr>
<td>3D Class. LF</td>
<td>3D light field slices</td>
<td>95.8 %</td>
<td>94.3 %</td>
</tr>
</tbody>
</table>

B. Feature Detection in Test Images

Table III shows the recall and precision values for 2D detection in rendered and light field images. Recall refers to subset of SIFT detected-features that match a CNN-detected feature. Precision refers to the subset of CNN-detected features that match a SIFT-detected feature. Due to the fixed scale of the input images, only SIFT features with a scale of 2.67 ± 1 and 1.33 ± 1 for the rendered and light field images, respectively, were used for comparison initially. These scales correspond to the 32 × 32 and 16 × 16 areas that SIFT uses when creating a descriptor, and have been chosen with respect to the 32 × 32 and 16 × 16 patches used as inputs to the model.

For detection in rendered images, the peak was set to 10 and the edge threshold was set to 20 for consistency between SIFT features used for extracting the training data. For 2D light field detection, because the images are smaller and yield fewer features, the peak threshold value was lowered to 1.

Next, we validated our choice of scale value for the SIFT features. Figure 10 shows the precision and recall values for the 2D rendered images when compared against SIFT features at varying scales (±1). Precision is highest at a scale of 2.67, as we expect most features to be around that scale. Recall does not follow a trend, and may be more sensitive to other parameters such as the range of SIFT scale values used for feature matching. The same behavior is observed for the 2D light field slices (not shown).

### TABLE III

<table>
<thead>
<tr>
<th>Model</th>
<th>Specifications</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D Detection</td>
<td>32 × 32 scale</td>
<td>15.8 %</td>
<td>27.6 %</td>
</tr>
<tr>
<td>2D Detection LF</td>
<td>16 × 16 scale</td>
<td>12.2 %</td>
<td>14.7 %</td>
</tr>
</tbody>
</table>

The detection values are expected to be low due to the aforementioned polarized classification of good and bad patches during acquisition of training data. Good patches consist only of SIFT features and bad patches are restricted to regions a certain distance away from the good patches. There are no training patches that include a SIFT feature that is only slightly off-centered. In a full image, these areas are difficult for our model to classify correctly. This further
motivates the use of COLMAP to filter the SIFT features into ones that yield a match in COLMAP (good) and those that do not (bad). Even better would be the inclusion of patches in the training data that correspond to bad features but look very similar to a SIFT feature. These patches can be created by running SIFT on rendered images as before, but then using information from the light field to determine whether those features result from texture of a Lambertian surface (good) or from reflections, refractions or occlusion boundaries (bad).

Another reason for poor detection is the fixed scale of the input patches. While input patches are sampled across all sizes during the acquisition of training data, they are rescaled to a fixed size before being fed into the model (see Figure 7). Accounting for features detected at multiple scales in an image would improve the number of matches between CNN and SIFT features. The easiest way to do this is by feeding the model the same image at different scales and generating a heatmap for each scale. In order to extract the relevant features across all scales, a 3D non-maxima suppression method would be required. Another approach would be to adjust the training architecture to allow input patches of different scales, as used in [6].

Finally, we considered the effect of the multiple pooling and stride steps on the layer resolution in our CNN. To reduce the amount of downsampling within the CNN, we added additional convolutional layers and removed some of the pooling. However, the detection precision and recall did not improve for the 2D rendered test images. This may be due to the finer resolution of detected features, which make exact matching with SIFT features more difficult.

IX. CONCLUSIONS

Overall, this project was very instrumental for learning to use CNNs to accelerate feature detection in light fields. First, through the results of the autoencoder, we demonstrated that light fields contain useful information in the additional dimensions, motivating the idea of using this information to extract better features. We then discussed our training data acquisition process and our model architecture for both classification and detection of features in light fields. Our results indicate that the CNN model can classify patches as good or bad with very high accuracy of above 94%. Lastly, our model detects some relevant features in light field images. Our results on feature detection in light fields can be further improved with more meaningful training samples and with the development of a model architecture that is capable of detecting features at multiple scales.

X. FUTURE WORK

To further improve the detection of features in the light fields, the model needs to be trained on less polarized data. Hence, we plan to train the model with good features being those that are filtered through COLMAP and bad features being those that are not filtered through COLMAP, but instead pass a “bad-patch-finder” that makes use of, for example, information about specularities found in a light field. This will improve the training dataset and, hence, feature detection in light field images.

To allow the model to detect features at multiple scales, we can scale the input images to detect features at different scales. Another alternative for distinguishing features at multiple scales is to create branches for inputs of different sizes to resemble a scale space within the model [6]. The ability of the model to identify features at multiple scales will increase its robustness and improve its performance.

Lastly, the model architecture in this report uses 3 dimensions for feature detection, u, v, and t with fixed s. To utilize all the information in the 4D light fields, the model can be modified to take a 3D input volume with the dimensions being u, v, and depth. Since depth depends on both s and t, a model that incorporates depth takes into account information from all four light field dimensions which should result in improved performance.

XI. CONCLUSIONS

L.J. designed the autoencoder and implemented the 2D models including data parsing (data_parser.py). F.M. performed feature matching in COLMAP and wrote the code for SIFT feature detection (SIFTFeatureExtraction.m) and 3D CNN model. K.G. wrote the code for Harris feature detection (harrisdetect.m), heatmap evaluation (heatmap.m), and 3D CNN model. All authors interpreted the results and prepared the poster and report.

XII. ACKNOWLEDGMENTS

The authors would like to thank Jayant Thatte, Vincent Sitzmann, and Dr. Donald G. Dansereau for their mentorship, contributions, and helpful discussions during the project.

REFERENCES