DEPTH MAP FINGERPRINTING AND SPLICING DETECTION

Falko Matern[⋆] *Christian Riess*[†] *Marc Stamminger*[⋆]

[⋆] Visual Computing, Friedrich-Alexander Universität Erlangen-Nürnberg (FAU)
[†] IT Security Infrastructures Lab, Friedrich-Alexander Universität Erlangen-Nürnberg (FAU)

ABSTRACT

With the ubiquity of social networks, images have become crucial in todays exchange of information. Most of these images are taken by smartphones. For forensic approaches relying on fixed image formation pipelines, the capabilities of smartphones using computational photography pose new challenges. But these new capabilities also offer opportunities for forensic analysis. A growing amount of commodity devices are able to capture 3-D information using various technologies such as stereo imaging or structured light. Modern smartphones commonly save such 3-D information as depth maps alongside regular images.

In this work, we propose to use characteristic artifacts of depth reconstruction algorithms as trace for forensic analysis. The proposed method is able to infer the source algorithm of stereo reconstructions with an accuracy of up to 97%. We further demonstrate the applicability of the method to collected smartphone data. It is able to discriminate patches from different sources with an AUC of up to 0.88 and can be used for splicing localization in depth maps.

Index Terms— image forensics, source identification, forgery detection, depth, smartphone

1. INTRODUCTION

Images play a major role in todays communication and constitute an important source of information. Particularly the widespread use of smartphones enables to instantly capture high quality images and to share them across social networks. At the same time, powerful image editing tools and new manipulation methods, such as "deepfakes", reduce trust in image authenticity. As a result, methods for image authentication and manipulation detection become even more relevant. An overview of such methods can be found in text books on image forensics, e.g., [1, 2]. Recent works successfully propose deep-learning methods for forensic tasks [3, 4, 5, 6, 7, 8].

Some forensic methods explicitly target smartphone images [9, 10, 11, 12], but the increasing use of smartphone cameras [13] poses new challenges to traditional forensic approaches. In smartphones, images are increasingly formed from a variable software-stack and computational photography. Recent smartphones for example merge bursts of RAW images to increase resolution or to enable low-light photography of high quality [14]. But such new imaging capabilities can also lead to completely new traces for a forensic analyst.

This material is based on research sponsored by the Air Force Research Laboratory and the Defense Advanced Research Projects Agency under agreement number FA8750-16-2-0204. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory and the Defense Advanced Research Projects Agency or the U.S. Government.

Fig. 1. Image from an Apple iPhone 7+ in portrait mode. Left, the main image with bokeh effect is shown. The unblurred image (middle) and depth image (right) are saved within the image file for further processing and can be extracted for forensic purposes.

One of these capabilities is to capture 3-D information, which is for example used for face identification or to simulate bokeh effects in portrait photography. Modern smartphones commonly save this 3-D information as so called depth maps within the same file as the RGB image for further processing. Figure 1 shows an example of a RGB image and its corresponding depth map, where scene depth is encoded as intensity. This additional data, available through a "side-channel", offers a great opportunity for forensic applications. To our knowledge, the only other work that forensically targets depth data analyzes temporal noise in depth video streams from cameras that are similar to Microsoft's Kinect [15].

Depth information can be captured by various technologies such as stereo imaging, structured light or time-of-flight. Each approach has its individual advantages and limitations, and in particular introduces modality-specific characteristic artifacts into the depth image. From a forensic perspective, these artifacts are highly interesting traces that allow to link a depth image to a depth-calculation algorithm. Furthermore, making depth data subject to forensic analysis raises the effort for a manipulator, who is then forced to additionally doctor the depth image in a consistent way.

In this work, we propose to use depth maps from modern smartphones for camera fingerprinting and splicing detection. We train deep-learning models on the outputs of a variety of stereo reconstruction algorithms. These models can then be transferred to a dataset of smartphone images. To this end, a small smartphone dataset for fine-tuning and final evaluations is collected. The data and methods are described in detail in Sec. 2. The results and application of the method to splicing detection are presented in Sec. 3. We conclude the work in Sec. 4.

2. METHOD

The proposed method links patches of depth maps to their source algorithm and determines whether two patches stem from the same algorithm. First, in Sec. 2.1, we describe the deep-learning architecture and training of the network. In Sec. 2.2, we describe data collection, generation and its characteristics.

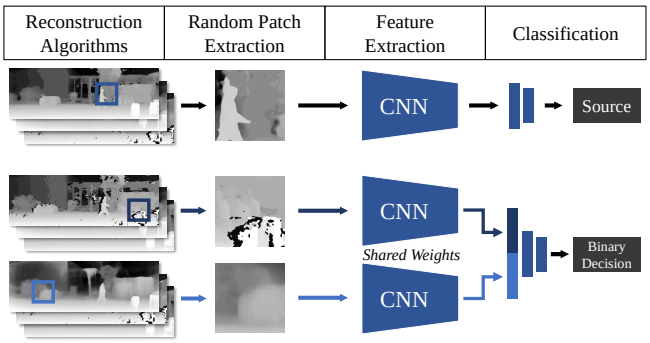

Fig. 2. Overview of the proposed method for algorithm fingerprinting (top) and patch discrimination (bottom).

2.1. Algorithm Fingerprinting and Patch Discrimination

We make the assumption that different modalities or algorithms for depth reconstruction produce characteristic, distinguishable artifacts. We propose a deep-learning method to infer the source algorithm of a depth map based on these differences. Similar to related methods [5, 7, 8], we work on patches, but only use depth maps as input. An overview of the method is shown in Fig. 2 on top.

We use images from different depth reconstruction algorithms. Random patches are extracted from these images to predict the reconstruction algorithm. Each training batch is balanced, such that each class contributes the same number of patches. To expedite the training, multiple patches from within the same image are extracted per batch. The random extraction of patches can be seen as data augmentation, and additionally the patches are randomly flipped horizontally and vertically. The input depth data is scaled from value range $[0, 255]$ to $[-1, 1]$. Completely flat patches, where all pixels have a constant value, are discarded.

A Convolutional Neural Network (CNN) as feature extractor, followed by a top part of several fully-connected layers (cf. Sec. 3.1) for classification is trained on these patches. The output layer consists of C neurons, where C is the number of the available reconstruction algorithms to distinguish. This layer uses a softmax activation function,

$$p_i = S(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \ , \tag{1}$$

where z_i are the node activations for i in $[1 \dots C]$. The softmax output p can be interpreted as class probabilities. For training, we use a categorical cross-entropy loss function

$$L(y, p) = -\sum_{i=1}^{C} y_i \log(p_i) \ , \tag{2}$$

where the ground-truth label y is encoded as one-hot vector.

We use this fingerprinting network also within a Siamese architecture (Fig. 2 (bottom)) to distinguish whether two patches originate from the same algorithm and for splicing detection. The input to the Siamese network consists of a pair of patches. The same underlying CNN is used as a feature extractor. The resulting features are concatenated and classified by a subsequent part of fully-connected layers (cf. Sec 3.1). For the binary decision whether both patches originate from the same algorithm, a single output node is followed by a sigmoid activation function

$$k = \sigma(z) = \frac{1}{(1 + e^{-z})} \ . \tag{3}$$

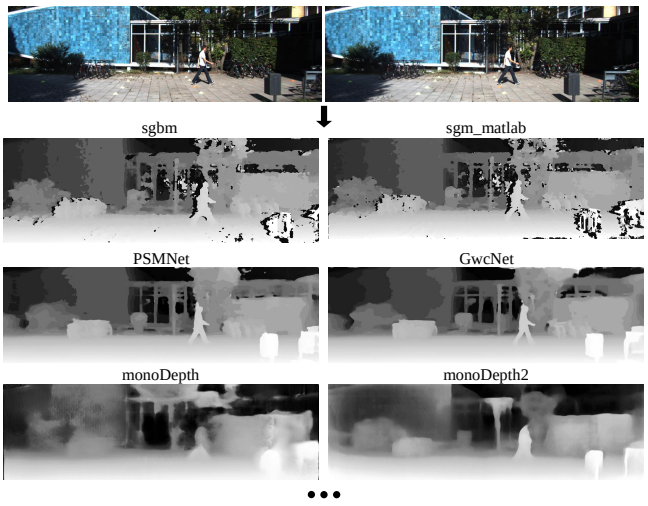

Fig. 3. The depth maps are reconstructed with twelve different algorithms, given the same RGB stereo pairs of the Kitti 3-D Object Detection dataset [16] as input (image contrast is enhanced for the purpose of visualization).

The network is trained by minimizing the binary cross-entropy loss

$$L_b(y, k) = -(y \log(k) + (1 - y) \log(1 - k)) \ . \tag{4}$$

The new top layers are randomly initialized for training. We first leave the CNN weights fixed and train only the top layers. In a second step, end-to-end learning can be used. Each training batch is balanced to contain the same amount of pairs from the same and different depth algorithms.

2.2. Data

Stereo reconstruction: We generate depth maps with different algorithms on 7481 RGB stereo pairs of the Kitti 3-D Object Detection dataset [16]. Distinguishing these algorithms serves as a proxy task for training, in order to later distinguish different devices. This dataset has the additional advantages of perfectly aligned image content across algorithm classes and full control over the processing pipeline. Depth maps are reconstructed with twelve different algorithms. Six algorithms use standard stereo reconstruction methods in OpenCV and Matlab. Here, bm and sgbm denote block matching and semi-global matching in OpenCV, and bm_matlab and sgm_matlab their Matlab counterparts. In OpenCV, we also postprocess samples with a Weighted Least Squares disparity map filter, denoted as wls_bm and wls_sgbm. Additionally, we create depth maps with three state-of-the-art deep-learning methods for stereo matching, namely PSMNet [17], hd3 [18], and GwcNet [19]. We also use three single-image depth reconstructions, namely monoDepth [20], monoDepth2 [21], and denseDepth [22]. Examples for six methods are shown in Fig. 3. All depth images are stored as 8-bit PNG images such that closer scene elements have larger depth map intensities.

Smartphone data: The smartphone depth maps are typically stored in the same image file as the color image. More specifically, Android devices store depth data within JPEG files as XMP metadata [23]. Apple devices use the High Efficiency Image File Format (HEIF), which directly supports addition of auxiliary images such as alpha channels or depth maps [24]. Figure 1 shows an example im-

Fig. 4. Example images of the four scenes of the collected smartphone dataset.

age from an Apple iPhone 7+ in portrait mode. The unblurred RGB image and depth map have been extracted from the same HEIC file.

We collect a small dataset with five different smartphones that provide depth data, namely an Apple iPhone X, Apple iPhone 7+, Google Pixel 3a, Motorola Moto X4, and Motorola Moto G6+. To that end, we set up four scenes and take five images per phone and scene. The scenes are shown in Fig. 4. During acquisition, the scene illumination is fixed. All phones are set to operate in portrait mode. Due to differences in the camera models, e.g., in focal lengths, the images between two devices cannot be exactly aligned. We hence capture scenes that only approximately show the same content. All depth maps are grayscale images, and directly extracted from the original files saved on the smartphones. Some smartphones assign larger intensities to scene points at larger distances. Depth map intensities from such devices are mirrored, such that closer scene elements always have larger intensities. The images are stored without further modification in JPEG format with quality 100. Some conditions in the collected dataset remain uncontrolled, as the internal processing pipelines are unknown and technical details such as resolution, stereo baseline or focal length vary across devices.

3. RESULTS

Details on the specific network architectures and their training are provided in Sec. 3.1. Algorithm fingerprinting on generated depth maps is evaluated in Sec. 3.2. The application to smartphone data and splicing detection is evaluated in Sec. 3.3.

3.1. Architecture and Training

The proposed model is trained on CNN architectures that proofed successful for related forensic tasks [6, 8, 5, 25]. More specifically, we consider four architectures that are referred to as Xception [26], ResNet-50 [27], MislNet [28] and MesoNet [25]. For Xception and ResNet-50, the top layers are Dense(1024), Dropout(0.5), Dense(512), Dropout(0.5), Dense(C) with ReLU activations. For MislNet and MesoNet, the original top layers by [28] and [25] are used. Global averaging is added between the CNN and the top layers to enable varying patch sizes.

For patch discrimination and splicing detection, all models use our proposed top layers but with a single output node. All models are trained with the Adam [29] optimizer, for the same number of epochs and with patches of size 128×128. The data is split into training, validation, and test data as described below. For all experiments, the checkpoint with lowest validation error is chosen for evaluation.

3.2. Evaluation with Generated Data

For algorithm fingerprinting, the 7481 images from the Kitti dataset are split into 5404, 954, and 1123 images for training, validation, and test. On the test images, we classify the center patch per class. The average classification accuracy is shown in the middle column of Tab. 1. Xcpetion CNN performs best with a remarkable accuracy of 0.97. Detailed classification accuracies per class are shown by the confusion matrix in Fig. 5. Here, confusion predominantly occurs between very similar algorithms such as GwcNet [19] and PSMNet [17], which is consistent with our assumptions. ResNet-50 and MislNet perform slightly worse with accuracies of 0.91 and 0.90. MesoNet only achieves an accuracy of 0.62. Upon closer examination, we found that MesoNet also confuses qualitatively similar classes. These results show that depth maps reliably fingerprint depth computation algorithms.

Table 1. Results for algorithm fingerprinting and patch discrimination. Evaluation on generated data with different network variants.

Architecture	Fingerprinting Accuracy	Discrimination AUC
Xception [26]	**0.97**	**0.997**
ResNet-50 [27]	0.91	0.989
MislNet [28]	0.90	0.983
MesoNet [25]	0.62	0.958

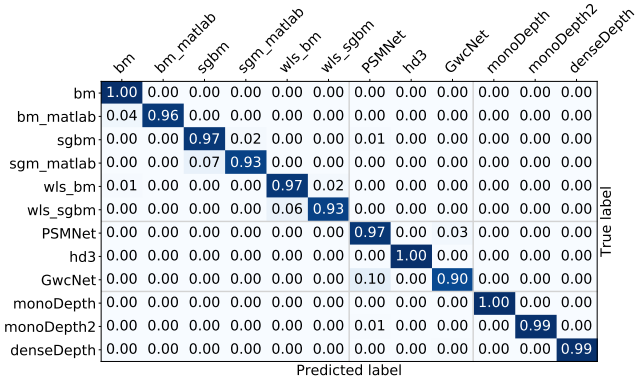

Fig. 5. Confusion matrix for algorithm fingerprinting with Xception architecture.

For patch discrimination, the trained network is integrated into a Siamese architecture. We only report results for retraining the new top layers, as additional end-to-end training did not further improve results. For evaluation, 13464 patch pairs are randomly chosen, so that 50% of the pairs comprise patches from the same algorithm, and the other 50% from different algorithms. The area under curve (AUC) of the receiver operating characteristic (ROC) curve are reported in the right column of Tab. 1. Consistent with the fingerprinting results, Xception performs best with an excellent AUC of 0.997. Overall, all networks perform very well with AUC values between 0.958 and 0.989.

3.3. Application to Smartphone Data

We use the smartphone data (cf. Sec 2.2) for patch discrimination. First, the models are evaluated without further training to test the generalization of the features learned from generated data. Then, we perform few-shot fine-tuning of the networks, by retraining the models with five images from a single scene of the dataset, shown in Fig. 4 (right). Three images per device are used for training, and two for validation. The images of the remaining three scenes are used as test data. For evaluation, we generate 3600 random samples from all

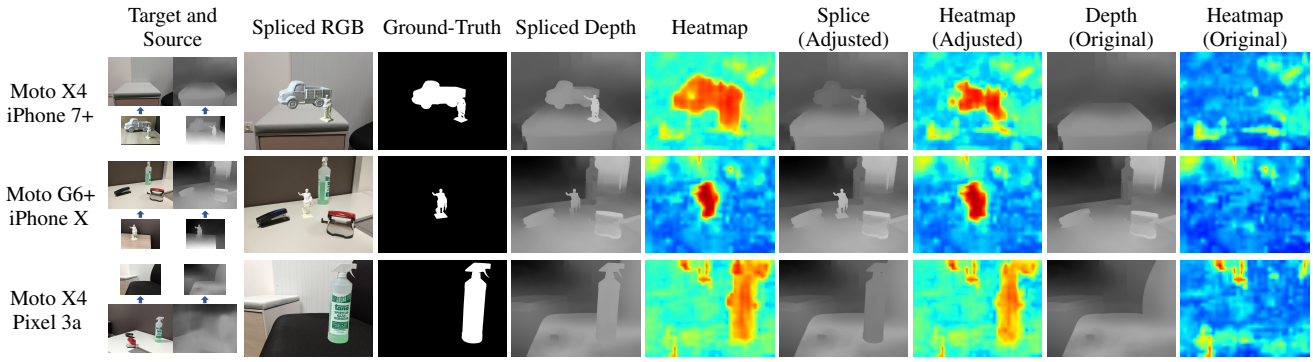

Fig. 6. The method can directly be applied to the task of splicing detection. The example depth map splices are created out of different smartphone sources. The heatmaps are generated analyzing just the depth maps with the fine-tuned Xception variant.

Table 2. AUC for ROC curve of discriminating patches from smartphone depth maps with different network variants.

Architecture	No fine-tuning	Fine-tuning top part	Fine-tuning end-to-end
Xception [26]	0.578	0.579	**0.883**
ResNet-50 [27]	**0.635**	0.607	0.828
MislNet [28]	0.585	**0.705**	0.841
MesoNet [25]	0.598	0.602	0.707

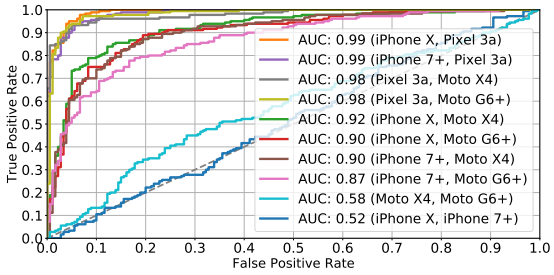

Fig. 7. ROC curves for evaluating samples of specific pairs of devices with the Xception variant.

devices. As before, half of the samples comprise patch pairs from the same device and the other half from different devices. The AUC values of the ROC curves are reported in Tab. 2. The experiment reveals a moderate generalization to the smartphone dataset with AUC values of up to 0.635 without further training. Retraining only the top layers has little impact on the performance, as shown in column three of Tab. 2. Interestingly, only MislNet improves its performance significantly to an AUC of 0.705. This architecture was specifically designed to extract high frequency features. By fine-tuning end-to-end, the performance of all models improves considerably to AUC values of up to 0.883, as shown in column four of Tab. 2. We analyze the performance in more detail, following the same evaluation scheme, but with samples from specific pairs of devices. The resulting ROC curves are shown in Fig. 7. The model cannot identify specific exemplars of the same device or reliably discriminate patches from smartphones of the same manufacturers. This is expected, since the depth reconstruction pipeline of these devices can be assumed to be quite similar, and we train on few images. Overall, the experiments indicate that the generalization of the models trained on generated data to smartphone data is limited, but they are well suited for fine-tuning with little data.

The method can directly be applied to splicing detection. We manually create example splices for a qualitative evaluation. For each splice, an object is selected and copied from the depth map of one device to the depth map of a different device without further editing. Additionally, we create examples, where the depth of the spliced object is manually adjusted by adding an offset to better fit the target, which is a likely step in creating a plausible spliced depth map. The RGB image is spliced for visualization only, as the method works on just the depth data. To analyze a depth map, patches of size 128×128 are extracted with a stride of 32. Each patch is compared against all others. The resulting predictions are averaged, yielding one prediction for each patch position. The final heatmaps are generated by scaling the predictions per patch position to image size. The examples and resulting heatmaps for spliced and original depth maps are shown in Fig. 6. The results demonstrate the performance of localizing splices in depth maps as a practical application of the proposed approach. Even further editing of the splice by adjusting depth values appears to have little impact on performance.

4. CONCLUSIONS

We propose to include depth maps into forensic analysis. Depth maps are commonly stored by smartphones as metadata of RGB images in common image file formats. We show that the artifacts from different modalities and different depth reconstruction algorithms can be a valuable forensic trace. First, we show the feasibility of using these artifacts for fingerprinting of depth maps from different algorithms. The proposed system achieves a fingerprinting accuracy of up to 97% given single depth patches. Further, we extend the system to a Siamese architecture. This network identifies patches from the same algorithm with a ROC AUC of up to 0.997. This model is successfully transferred to smartphone data, achieving AUC values of 0.883 using only a few images for training. The method can be directly applied to the task of splicing detection.

The use of auxiliary data such as depth maps is highly interesting as a "side-channel" for forensic analysis, forcing a manipulator to also consistently alter the depth maps. The impressive performance with generated data suggests that further improvements on smartphone images are possible. Future work will include larger scale training data and more extensive experiments. Additionally, we aim to investigate other uses of depth data for forensic analysis.

5. REFERENCES

[1] Judith Redi, Wiem Taktak, and Jean-Luc Dugelay, "Digital Image Forensics: A Booklet for Beginners," *Multimedia Tools and Applications*, vol. 51, no. 1, pp. 133–162, Jan. 2011.

[2] Hany Farid, *Photo Forensics*, The MIT Press, 2016.

[3] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi, "A Full-Image Full-Resolution End-to-End-Trainable CNN Framework for Image Forgery Detection," *arXiv preprint arXiv:1909.06751*, 2019.

[4] Davide Cozzolino and Luisa Verdoliva, "Noiseprint: a CNN-based camera model fingerprint," *IEEE Transactions on Information Forensics and Security*, 2019.

[5] O. Mayer and M. C. Stamm, "Forensic Similarity for Digital Images," *IEEE Transactions on Information Forensics and Security*, pp. 1–1, 2019.

[6] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner, "Faceforensics++: Learning to detect manipulated facial images," *arXiv preprint arXiv:1901.08971*, 2019.

[7] B. Hadwiger, D. Baracchi, A. Piva, and C. Riess, "Towards Learned Color Representations for Image Splicing Detection," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 8281–8285.

[8] Minyoung Huh, Andrew Liu, Andrew Owens, and Alexei A Efros, "Fighting Fake News: Image Splice Detection via Learned Self-Consistency," *European Conference on Computer Vision (ECCV)*, 2018.

[9] Qingzhong Liu, Peter A. Cooper, Lei Chen, Hyuk Cho, Zhongxue Chen, Mengyu Qiao, Yuting Su, Mingzhen Wei, and Andrew H. Sung, "Detection of JPEG double compression and identification of smartphone image source and post-capture manipulation," *Applied Intelligence*, vol. 39, no. 4, pp. 705–726, Dec 2013.

[10] J. R. Corripio, D. M. A. Gonzlez, A. L. S. Orozco, L. J. G. Villalba, J. Hernandez-Castro, and S. J. Gibson, "Source smartphone identification using sensor pattern noise and wavelet transform," in *5th International Conference on Imaging for Crime Detection and Prevention (ICDP 2013)*, Dec 2013, pp. 1–6.

[11] A. L. Sandoval Orozco, L. J. Garca Villalba, D. M. Arenas Gonzlez, J. R. Corripio, J. Hernandez-Castro, and S. J. Gibson, "Smartphone image acquisition forensics using sensor fingerprint," *IET Computer Vision*, vol. 9, no. 5, pp. 723–731, 2015.

[12] Patrick Mullan, Christian Riess, and Felix Freiling, "Forensic source identification using JPEG image headers: The case of smartphones," *Digital Investigation*, vol. 28, pp. S68 – S76, 2019.

[13] Ana Beln Gonzlez and Jose Pozo, "The Industrial Camera Modules Market," *PhotonicsViews*, vol. 16, no. 2, pp. 24–26, 2019.

[14] Bartlomiej Wronski, Ignacio Garcia-Dorado, Manfred Ernst, Damien Kelly, Michael Krainin, Chia-Kai Liang, Marc Levoy, and Peyman Milanfar, "Handheld Multi-frame Super-resolution," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 28:1–28:18, July 2019.

[15] Noa Privman-Horesh, Azmi Haider, and Hagit Hel-Or, "Forgery Detection in 3D-Sensor Images," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[16] Andreas Geiger, Philip Lenz, and Raquel Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[17] Jia-Ren Chang and Yong-Sheng Chen, "Pyramid Stereo Matching Network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5410–5418.

[18] Zhichao Yin, Trevor Darrell, and Fisher Yu, "Hierarchical Discrete Distribution Decomposition for Match Density Estimation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[19] Xiaoyang Guo, Kai Yang, Wukui Yang, Xiaogang Wang, and Hongsheng Li, "Group-wise Correlation Stereo Network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3273–3282.

[20] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 270–279.

[21] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow, "Digging into self-supervised monocular depth estimation," *arXiv preprint arXiv:1806.01260*, 2018.

[22] Ibraheem Alhashim and Peter Wonka, "High Quality Monocular Depth Estimation via Transfer Learning," *arXiv e-prints*, vol. abs/1812.11941, 2018.

[23] "Overview Depthmap Metadata Google Developers," `https://developers.google.com/depthmap-metadata/`, accessed 2019-10-08.

[24] "Capturing Photos with Depth Apple Developer Documentation," `https://developer.apple.com/documentation/avfoundation/cameras_and_media_capture/capturing_photos_with_depth`, accessed 2019-10-08.

[25] Darius Afchar, Vincent Nozick, Junichi Yamagishi, and Isao Echizen, "MesoNet: a Compact Facial Video Forgery Detection Network," in *2018 IEEE Workshop on Information Forensics and Security (WIFS)*, 2018.

[26] Francois Chollet, "Xception: Deep Learning With Depthwise Separable Convolutions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[28] B. Bayar and M. C. Stamm, "Constrained Convolutional Neural Networks: A New Approach Towards General Purpose Image Manipulation Detection," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2691–2706, Nov. 2018.

[29] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.