Deep learning applied to quad-pixel plenoptic images

Guillaume Chataignier, Benoit Vandame, Jérôme Vaillant

Deep learning applied to quad-pixel plenoptic images

Guillaume Chataignier[a], Benoit Vandame[a], and Jerome Vaillant[c]

[a]Interdigital, 975 Avenue des Champs Blancs, Cesson-Sévigné, France
[b]CEA-Leti, 17 Avenue des Martyrs, Grenoble, France

ABSTRACT

In recent years, we have seen the development of integrated plenoptic sensors, where multiple pixels are placed under one microlens. It is mainly used by cameras and smartphones to drive the autofocus of the main lens, and it often takes the form of dual-pixels with 2 rectangular sub-pixels. We study the evolution of dual-pixels, the so-called quad-pixel sensor with 2x2 square sub-pixels under the microlens. As it is a simple light field capturing device, we investigate the computational photography abilities of such sensor. We first present our work on pixel-level simulations, then we present a model of image formation taking into account the diffraction by the microlens. Finally, we present new ways to process a quad-pixel images based on deep learning.

Keywords: Quad-Pixel, Plenoptic, Sensors, Deep, Learning, Demosaicing, Diffraction, Depth

1. INTRODUCTION

Pixel sensors are covered with microlenses to ensure that photons converge on the photodiodes located at the middle of pixels. With the growing number of pixels, new pixel design are experimented : microlenses can be designed to cover a fix number of pixels. In this case, the matrix of pixels becomes an integrated plenoptic sensor. Dual-pixel sensors are a simple kind of plenoptic sensors where a microlens is shared between two rectangular sub-pixels. Canon and Samsung are using dual-pixel[1,2] to drive the autofocus in their cameras or smartphones and Sony is advertising quad-pixel[3] sensors where a microlens is shared between 2x2 square sub-pixels for better autofocus. Google is using dual-pixel to create a depth map and to produce synthetic bokeh for the portrait mode of their smartphones.[4–6] An integrated plenoptic sensor produces one image per sub-pixel, the Sub-Aperture Images (SAI's), which are shifted relatively to each other depending on the amount of defocus. The final image is the sum of the 4 SAI's. Computational photography algorithms are based on the shifts between SAI's and allows refocusing, correction of main lens aberrations or passive depth estimation.

In a previous work[7] we have analysed a quad-pixel sensor at pixel level using electromagnetic simulations. Then, we have modified a ray-tracing renderer software (PBRT[8]) to take into account the angular responses of such quad-pixel thus creating more accurate synthetic images.

Using this tool we have generated a dataset composed of 35 images, declined in multiple versions: ideal rendering with perfect thin lens model ; extended rendering with geometrical aberrations ; complete rendering adding diffraction obtained with electromagnetic simulations. Thanks to the raytracer, we also produce ideal depth maps. While we are waiting for a prototype coming this year, this aricle presents our work on image processing based on deep learning. To our knowledge, this is the first paper which applies deep learning algorithms to quad-pixel images.

Our contributions In this paper we explain the image formation when using a quad-pixel. Indeed, the blur function of the SAI's varies in a complex maner depending on the depth, the defocus and the angular response. Completing the work of Punnapurath,[9] our image formation model explains how the blur functions are mirored between sub-apertures, and also we consider the impact of the diffraction.

Further author information: (Send correspondence to G.C.)
G.C.: E-mail: guillaume.chataignier@interdigital.com
B.V.: E-mail: benoit.vandame@interdigital.com
J.V.: E-mail: jerome.vaillant@cea.fr

Besides we recall how the dataset of quad-pixel images has been computed in section 3. In section 4 we address 3 algorithms with deep learning applied on the bayered SAI's ; 1/ SAI's demosaicing, which is a complex case of multi-frame demosaicing since blur function are not identical throughout SAI; 2/ diffraction correction, with custom networks trying to go from a raw to a corrected raw thanks to our database; and finally 3/ depth-estimation from the SAI's. In section 5, We then conclude our investigation on image processing dedicated to an integrated plenoptic sensor.

2. IMAGE FORMATION ON A PLENOPTIC SENSOR

2.1 Pixel level analysis

In this section we study a quad-pixel plenoptic sensor at pixel level using Finite Difference Time Domain simulations (FDTD). The plenoptic function is made by sharing a microlens between four square sub-pixels, and the pixel itself is made of severals layers as shown in Figure 1. We first optimize the design of an on-axis 3.5µm quad-pixel by changing the radius of curvature of the microlens and the height of the different layers. Then we optimize the design for the off-axis case by translating the microlens above the four subpixels, the other layers beeing unchanged. In the later case, the optimization is done given the angle θ_{cra} and φ_{cra} of the chief ray. We focus our analysis on the angular response of a 3.5 µm quad-pixel (1.75µm subpixel), which is the power absorbed by each subpixel given an angle of incidence θ and φ of a plane wave.

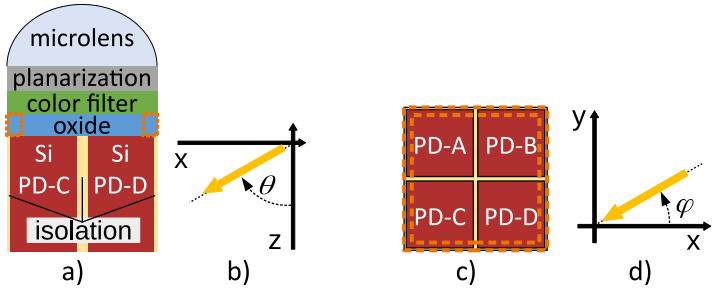

Figure 1. General design of a quad-pixel sensor with a planarisation layer, a color filter and a passivation layer. (a) and (c) show the view from the side and from the top. (b) and (d) show our angle convention for θ and φ.

We obtain 4D data (θ_{cra}, φ_{cra}, θ, φ) for each subpixel normalized between 0 and 1. These results (Figure 2) are represented in polar coordinates where θ is the radius and φ the polar angle for different Chief Ray Angles (CRA). They represent the crosstalk between the subpixels which is caused by the diffraction of the microlens. We showed[7] that the more subpixels crosstalk there is, the more the SAIs are similar, making disparity estimation difficult and reducing the plenoptic performance.

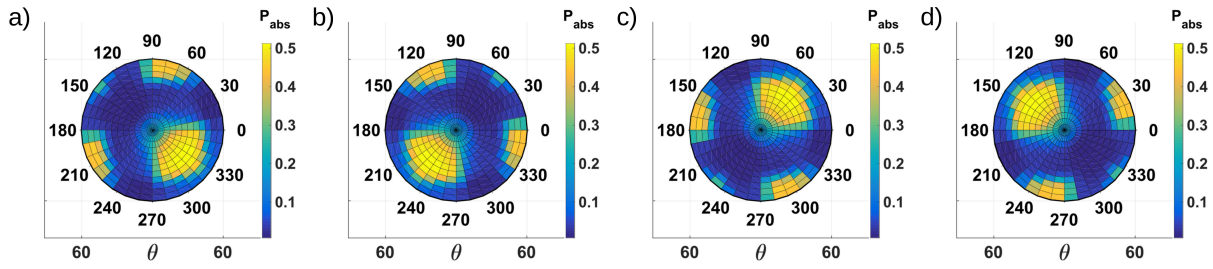

Figure 2. Angular response of the 4 sub-pixels with 3.5 µm centered microlenses.

2.2 Main lens spot diagram

A plenoptic sensor is used to capture images produced by a main lens. The SAIs define a 4D light-field were the sub-aperture 2D coordinates corresponds the 2D pupil coordinates. Depending on the amount of defocus, the

SAIs are globally shifted between themselves. Computational photography alogorithms preliminarly measures shifts between SAIs to estimate depth-maps. In our study we require the notion of spot digram to handle the geometrical aberrations of the main lens, and how these aberrations are projected in the 4D light-field.

Usually the spot diagram of a main lens is defined as the pattern formed by the impact of rays on the image plane, from a point source and through the optical system. It is defined for an object distance D and for a given position on the image plane (x_s, y_s) or a chief ray angle $(\theta_{cra}, \varphi_{cra})$. We propose here to extend the concept of spot diagram by adding the angle of incidence of each ray with respect to the image plane. What we call spot diagram in this section is now a reduced light field, corresponding to the parametrization (x, y, θ, φ) of every ray of a point source at distance D, going through the optical system and hitting the image plane at coordinates around (x_s, y_s). With a perfect lens, all rays hit the image plane exactly at (x_s, y_s) but it is not the case with a real lens. If we consider a finite amount of rays, for instance in an optical design software or in a ray-tracing based renderer, S is a set of rays:

$$S(D, x_s, y_s) = \{R_1, ..., R_i, R_N\} \tag{1}$$

where the rays have 4 coordinates, their position on the image plane and their angle of incidence:

$$R_i = R_i(x_i^r, y_i^r, \varphi_i, \theta_i) \tag{2}$$

(x_s, y_s) and $(\theta_{cra}, \varphi_{cra})$ are linked by the optical formula of the lens.

$$\theta_{cra} = f_{cra}\left(\sqrt{x_s^2 + y_s^2}\right) \tag{3}$$

$$\varphi_{cra} = \text{atan2}(x_s, y_s) \tag{4}$$

where f_{cra} can be computed by an optical design software. For instance Figure 3 shows CRA_{func} of a thin lens and a smartphone lens (US8320061-B2).

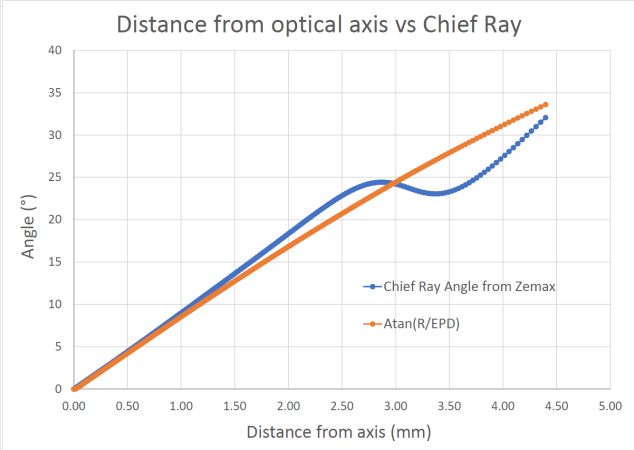

Figure 3. the blue curve is a CRA function of a smartphone lens computed with Zemax, the orange one is a CRA function of a thin lens.

This extended spot diagram will help us to explain how the blur kernel is formed in the next section.

2.3 Blur kernel model

A translating disk has been proposed by Punnappurath[9] as a model for the blur kernel (or Point Spread Function - PSF) of a dual-pixel sensor. Based on our knowledge about in-pixel diffraction, we propose a more generalistic equation which encompasses both the spot diagram of the main lens and the microlens diffraction's effect. Let's consider a portion of the scene F at a constant depth D, which forms an image centered at coordinates (x_s, y_s) on the image plane. We can write for each SAI:

$$G^{sai}(D, x_s, y_s) = F * H^{sai}(D, x_s, y_s) \tag{5}$$

and

$$G = \sum_{sai} G^{sai} = F * \sum_{sai} H^{sai} = F * H \tag{6}$$

where G^{sai} is the one of the SAI ($^{sai} = \{A, B, C, D\}$), H^{sai} is the blur kernel associated with one of the SAIs, G is the final image and H the global blur kernel of the system.

We propose to study H^{sai} more precisely. In a first approximation we neglect the diffraction of the main lens and we stick with the geometrical approach of a light-field. We can justify it by the fact that a wide-opened lens is used with a plenoptic sensor, in order to get a clear disparity signal, so the directions of rays is mostly affected by the geometrical aberrations rather than diffraction. Given the angular response of one SAI, normalized to 1

$$A = A^{sai}(\theta, \varphi, x_s, y_s) \tag{7}$$

we describe the blur kernel associated with this SAI as:

$$H^{sai}(D, x_s, y_s) = S(D, x_s, y_s) \otimes A^{sai}(\theta, \varphi, x_s, y_s) \tag{8}$$

$\otimes$ denotes what we call an angular multiplication. It formalizes the procedure of weighting each ray of the spot diagram by the angular response according to their angle of incidence, and add them to the kernel. We may also add the rays falling at the same coordinates (x, y). $S \otimes A$ is a weight map defined as:

$$S \otimes A = \{w_1, ..., w_i, w_N\} \tag{9}$$

where the weights have 2D coordinates x_i^w and , y_i^w, and are computed as:

$$w_i = w_i(x_i^w, y_i^w) = \sum_{\substack{r_j \in S \\ x_j^r = x_i^w, y_j^r = y_i^w}} r_j(x_j^r, y_j^r, \theta_j, \varphi_j) \times A(\theta_j, \varphi_j, x_s, y_s) \tag{10}$$

Equation (10) defines the angular multiplication for the image plane, the weights have decimal coordinates (x_i^w, y_i^w) which represent the intersection of the rays and the image plane. If we take into account the spacial sampling of the sensor, we have to add the rays falling on the microlens. In this case the weights have whole coordinates (x_i^w, y_i^w) which represent the indices of microlenses, and given the subpixel size p, we have:

$$w_i = w_i(x_i^w, y_i^w) = \sum_{\substack{r_j \in S \\ x_i^w - p \le x_j^r \le x_i^w + p \\ y_i^w - p \le y_j^r \le y_i^w + p}} r_j(x_j^r, y_j^r, \theta_j, \varphi_j) \times A(\theta_j, \varphi_j, x_s, y_s) \tag{11}$$

Finally, the blur kernel for one SAI at a constant depth is given by:

$$G^{sai} = F * H^{sai} = F * \left(S \otimes A^{sai}\right) \tag{12}$$

Figure 4 shows simulated Sub-Aperture Images of a point source by a thin lens, in front of the focus plane and behind the focus plane respectively. The first line of each figure is the blur kernel without diffraction and the second line is with diffraction. The global kernel is the sum of the 4 SAIs and it is shown on the right. Without diffraction, the light is perfectly split in 4 quarters which do not overlap. This is what we expect from an ideal quad-pixel plenoptic sensor. In the latter case, there is crosstalk between sub-pixels due to the microlens' diffraction and the blur kernels are not perfectly delimited.

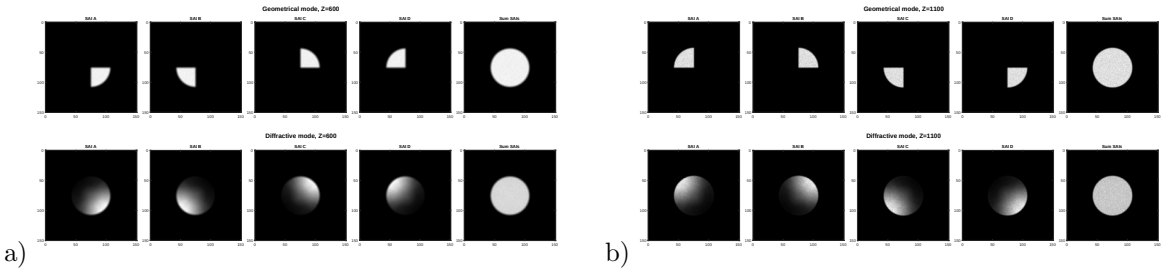

Figure 4. Simulated blur patterns using $D_{focus} = 750$mm. First and second lines in a) show the image of a point source at $z = 600$mm in geometrical and diffractive mode respectively. b) shows the image of a point source at $z = 1100$mm.

3. DATASET GENERATION

3.1 Mixing ray tracing and FDTD simulations

As quad-pixel plenoptic sensors are not available on the market, we had to work with synthetic images. In order to make them more realistic from a physical point of view, we wanted to take into account the effect of diffraction of the microlens.[7] In practice, the blur kernel described above is a bit hard to use because it depends on the distance of the scene, which is not constant, and we would have to generate multiple kernels for the different depths and sensor positions. Instead we modified the ray-tracing procedure and we implemented in PBRT v2. The principle is as following:

- One ray is launched from one sub-pixel, to the microlens.
- The angle of incidence (φ, θ) is computed just after the microlens. If the CRA is corrected, we also compute $(\varphi_{cra}, \theta_{cra})$.
- The ray gets (R,G,B) values from the scene.
- The ray is added to the surrounding sub-pixels under the same microlens, weighted by the angular responses of the given sub-pixels where it is added according the the angle of incidence.
- Repeat for multiple rays per sub-pixels and for every sub-pixels.

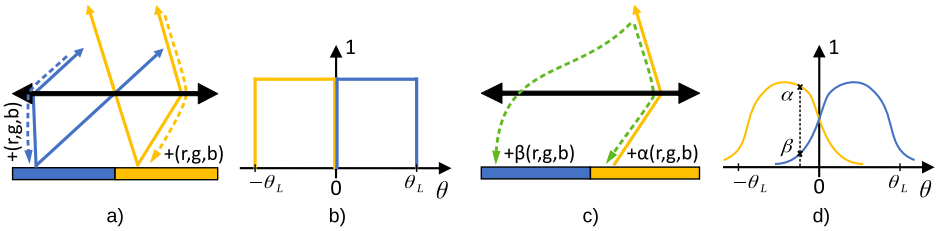

Figure 5. Modified ray-tracing procedure. a) shows the classic ray-tracing procedure applied to quad-pixel plenoptic sensors. b) shows the perfect angular responses formed by a). c) shows our custom ray-tracing procedure. Finally, d) shows the angular responses from pixel simulation from where we extract the weights applied to the rays.

Figure 5a) and b) illustrate the classic ray tracing procedure where rays are only added to the pixels from where they were launched, which forms perfect angular responses. Figure 5c) and d) illustrate our modified procedure, where we read the angular response of pixel simulations and use them to weight and splat the rays on the sub-pixels.

3.2 Different datasets

Using this new rendering method we have generated 35 synthetic images using the lens described in the patent US8320061-B2, 23 of them are based on the proposed 3D scenes of PBRT v2 (mainly SanMiguel), and other ones are photos placed as 2D textures so the depth is constant. Those 35 images are declined in diffractive and non-diffractive mode, for a total of 70 images. We also computed the depth maps for every scene. Each image is 3400x3400 pixels, so the SAIs are 1700x1700.

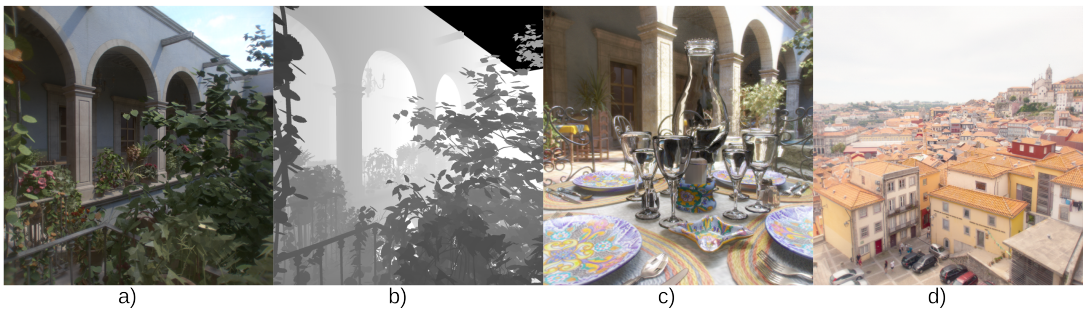

Figure 6. Exemples of generated images with our custom PBRT v2. a) and c) are 3D scenes, b) is the depth map of a), and d) is a photo used as plane texture.

4. IMAGE PROCESSING USING CONVOLUTIONAL NEURAL NETWORKS

4.1 Demosaicing

Usually for a plenoptic sensor of type 1, color filters are placed per microlenses in a RGB bayer pattern. It means that all SAIs are mosaiced and must be demosaiced. There are already neural networks for demosaicing RGB images, for instance the CNN presented by Syu[10] in its simple or very deep version (Figure 7 a and b)) or the bottlenecked resnet presented by Verma[11] (Figure 8) based on residual blocks. In our case, those networks must be applied to each SAI independently and the 4 demosaiced SAIs must be summed to produce the final image. However, the SAIs could be globally shifted between each other to achieve refocusing, or locally shifted to correct aberrations of the main lens. Depending on the shift applied it is possible to naturally obtain a partially demosaiced final image. For instance if the refocusing parameter ρ is 0, we have to demosaic each SAIs entirely, but if $\rho = 1$ then the final image is demosaiced completely.

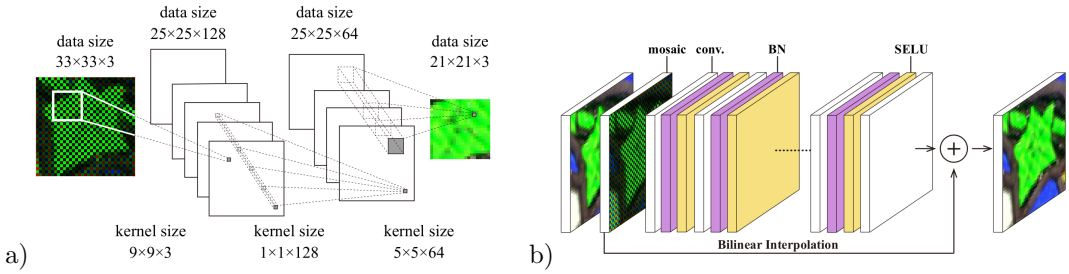

Figure 7. a) and b) show the demosaicing networks of Syu, respectively in simple and very deep version.

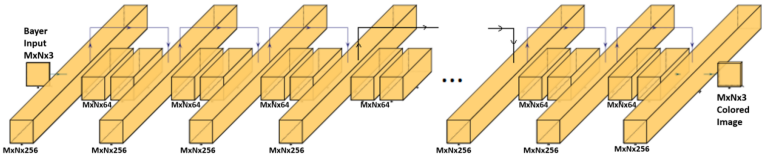

Figure 8. Demosaicing network of Verma based on residual blocks.

In this section we compare the networks presented above, used 4 times independently, to our network which takes 12 channels input (4 mosaiced SAIs concatenated) and outputs 12 channels (4 demosaiced SAIs). It is also based on residual blocks (Figure 9), as this architecture seemed better than regular CNNs, and has 485k free parameters (Syu VD: 670k, Verma: 714k). Each SAI goes through a first set of convolutions (Net 1) composed of a first convolution with kernel size of 7 and 48 channels then two other convolutions with a kernel size of 5 and 64 channels, each one follwed by a ReLu activation. This first net is shared between the SAIs. The 15

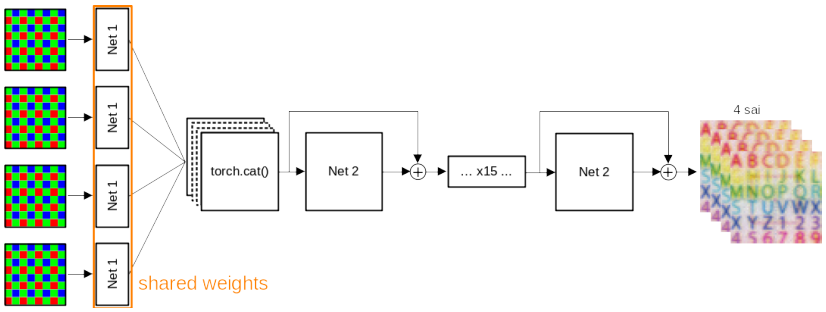

Figure 9. Our network, composed of 3 convolutions treating the SAIs, then a residual network which takes the features of the previous convolutions concatenated along channels dimension.

residucal blocks (Net 2) are similar to Verma's ones: one convolution with a kernel size of 1 and 128 channels, one convolution with a kernel size of 3 and 32 channels, and a final convolution with a kernel size of 1 and 128 channels. Each one is followed by a ReLu function. The results of each SAI after the Net 1 are concatenated along channels dimension, giving a 256 channels mid-result, and it is passed to the Net 2 using a single convolution with kernel size of 1 and 128 channels. The final convolution brings the 128 channels result of the last residual block to a 12 channels result, representing the 4 RGB SAIs.

4.1.1 Training

To train our network we use a custom loss. We compute the MSE of each SAI versus its ground truth, but also the MSE of the final image versus its ground truth. The goal is to have the network learn the final image, but also have good quality SAIs if desired. Without the MSE on SAIs, some experiments showed a good final image but SAIs with artefacts, preventing them from being useful for further processing. These two factors can be weighted, but we used $\alpha = 0.5$ and $\beta = 0.5$.

$$L = \frac{\alpha}{4} \sum_{s=SAI} \text{MSE}(s, \hat{s}) + \beta \text{ MSE}(\text{FI}, \widehat{\text{FI}}) \tag{13}$$

The images from PBRT are patched, then mosaiced using randomly choosen color pattern per microlens (namely RGGB, BGGR, GRBG, GBRG). During the mosaicing process we place zeros at unkown colors but the channel dimension is preserved. Finally they are transformed into SAIs and stacked along channels dimension, forming 12 channels data (4 SAIs of 3 channels). We also implemented some random transformations like flips or rotations. We also implemented random shift between the SAIs. We chose only integer values between -2 and +2 for both the horizontal and vertical axis. The shifts do not necessarily follow the refocusing equation because we also want to take into account local shifts due to aberration correction for instance. We were forced to use integer values because otherwise the interpolation needed for subpixel translation degraded the ground truth too much.

The Syu and Verma's networks are trained using the Flickr500 dataset, while our network is trained on the custom PBRT dataset presented above. From the PBRT dataset we have selected 5 images for the validation set and 5 other images for the test set. These 10 images are not seen by the network during the training. In the results presented in the next section we show the average PSNR of the 5 test images only. It should limit the impact of the bias due to using different dataset. On the other hand, the Flickr500 dataset is composed of very sharp images, contrary to our dataset. We are mainly limited by the 3D models and the textures used in the proposed scenes of PBRT v2. Overall, the networks from Syu and Verma have learned from more difficult cases (high frequencies images) and it should compensate the fact that our network is trained directly with the PBRT dataset.

4.1.2 Results

The inferences are done for 2 cases, for a refocusing parameter $\rho = 0$ and $\rho = 1$, and we compute the PSNR on the final image for each cases and networks (table 1). We only take into account the 5 images of the test set.

The networks for 2D images are used on each SAI independently. Figure 10 shows one exemple of inference of our network.

	Bilinear	Syu VD	Verma	Our
$\rho = 0$	35.00964	36.0144	37.651	37.775
$\rho = 1$	37.743	38.878	40.753	40.865

Table 1. PSNR of inferences for $\rho = 0$ and $\rho = 1$ and the different networks. Bilinear is given for reference.

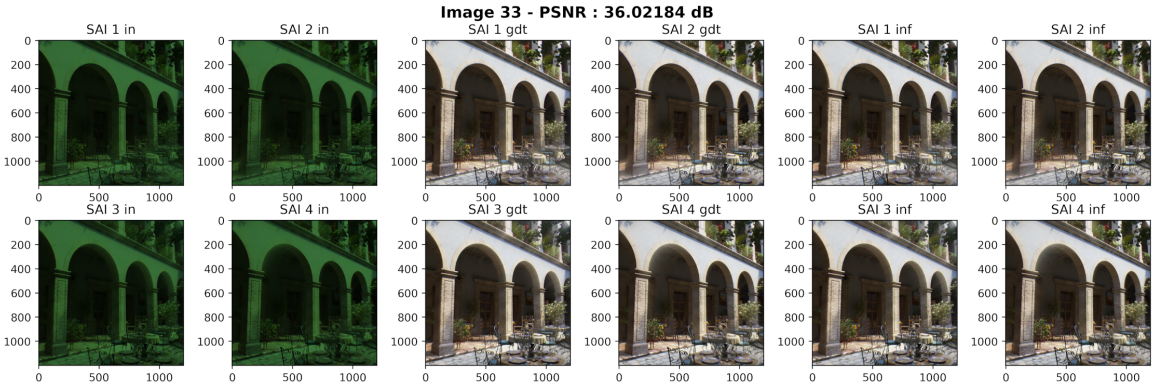

Figure 10. Exemple of inference (only the SAIs are represented). Left: mosaiced SAIs, center: ground truth, right: inference.

As expected, the case where the refocusing parameter $\rho = 1$ is way better because all pixels receive the contribution of the 4 different colours, and the gain is almost +3dB for every network. We can see that our network is slightly better, especially when the $\rho = 1$. It means the network can use the shifts between the SAIs to improve the quality of the demosaicing. It is also important to note that our network is roughly 25% lighter than Syu's VD network, which could be useful in energy contrained environments like smartphones. Those results need to be confirmed because the usage of 2 different datasets could have introduced a bias. For exemple one can train the Syu or Verma's networks on the PBRT dataset instead of the Flickr500.

4.2 Diffraction correction

In this section we present an attempt to correct the diffraction caused by the microlens, in other words to remove the crosstalk between SAIs. We tried a simplified bottleneck resnet and a simplified 3-levels unet.[12] The "diffraction images" from PBRT are the inputs, and the "diffraction-less images" are the ground truth. Because the diffraction appears at pixel level, we decided to work with "raw data": the images are first mosaiced using a per microlens Bayer pattern, then the channel dimension is flattened. These single channel images are patched are fed to the networks.

The unet follows the Ronneberger's architecture, but we only use 4 levels (instead of 5) and 8 channels for the first level (instead of 64). In this form, the network has 120k parameters. The residual network is very similar to the demosaicing one but shorter. ReLU functions follow each convolution. It has 382k parameters:

- A convolution of 256 channels, with a kernel size of 2 and a stride of 2. It is meant to transform the image from "raw space" to "SAI space", but we let the network learn different combinaisons of subpixels. In fact it is the same as feeding the network with SAIs and using a kernel size of 1 and a stride of 1. It depends only on how the data have been prepared.
- 3 residual blocks. Each block is made of 3 convolutions of 64, 64, and 256 channels respectively. The kernel size is 1, 3, and 1 respectively. It is the same block as the demosaicing network but we increase the number of features.
- A transposed convolution of 64 channels, a kernel size of 2 and stride of 2. It is meant to go back to "raw space".

- 2 convolutions of 32 channels and a kernel size of 5.
- One last convolution of 1 channel and a kernel size of 1 used to flatten the features.

We use the MSE on the raw images as the loss function. We show the real inference in Figure 11 and a bilinearly demosaiced version fot better visualization in Figure 12. On the test images we read an average PSNR of 21dB for the residual net, and 19dB for the unet. It shows it may be possible to remove the effect of diffraction by the microlenses and produce cleaner SAIs, without crosstalk.

Further investigation needs to be done, for instance better network architecture, different loss function, or a better dataset. This experiment is easy to do because we based our research on synthetic images and a rendering tool which easily allows to add or remove the diffraction effect. It seems more difficult to create diffraction-less images in real life. One way to do it would be to place 3 quarters of an opaque disk at the exit pupil position of a lens, allowing light to go through only one quarter of the pupil at a time. Four acquisitions are necessary, each one with the disk rotated by step of 90 degrees. On each one of the 4 images, one would keep only the subpixel that was really aimed by the quarter of the main lens, and zero the other subpixels. Finally, the 4 images would be merged together by replacing the zeros by the subpixel values from the 3 others acquisitions. This method is only theoritical and has not been tested.

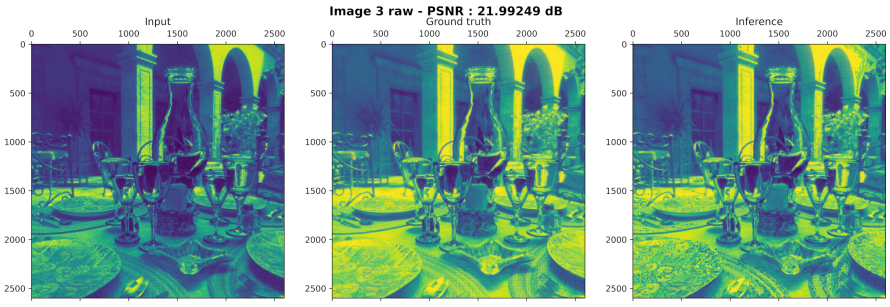

Figure 11. Exemple of inference, raw images. Left: raw with diffraction, center: ground truth without diffraction, right: inference.

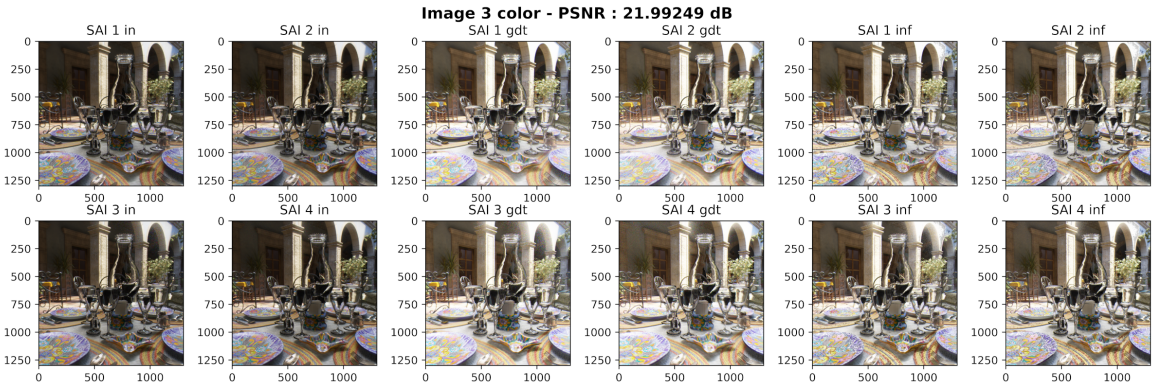

Figure 12. Exemple of inference, colored SAIs images. Left: SAIs with diffraction, center: ground truth without diffraction, right: inference.

4.3 Depth estimation

In order to test depth estimation with quad-pixel we re-implemented both the DU2Net from Zhang[6] and the method of Punnappurath,[9] and adapted them to the quad-pixel case. The Du2Net aims to fuse the data from dual-pixels and dual-cameras in order to produce a depth map, so we extracted only the dual-pixel part of the network. Unfortunately we were not able to produce good depth maps, the different networks did not

converge in our case. Concerning the Punnappurath's method, the dataset we generated with PBRT was not appropriate because Punnappurath method's relies on constant depth patches. Especially, the San Miguel scene was problematic in our case because they do not really contain constant depth areas. The only constant depth images we had were the photographs but they were simulated at the same depth every time. As for the Du2Net, we were not able to make it converge. We think that it is also due to our dataset which may not be large enough to train the network properly. In addition to that, the textures used in PBRT v2 may not be detailed enough to have the network "see" the disparities. However, we believe that passive depth mapping is possible and is likely to be better than dual-pixel because it adds the vertical disparities. We tried a simple block matching method on the San Miguel scene (the table, Figure 6c) and it appears that some disparities exist (Figure 13).

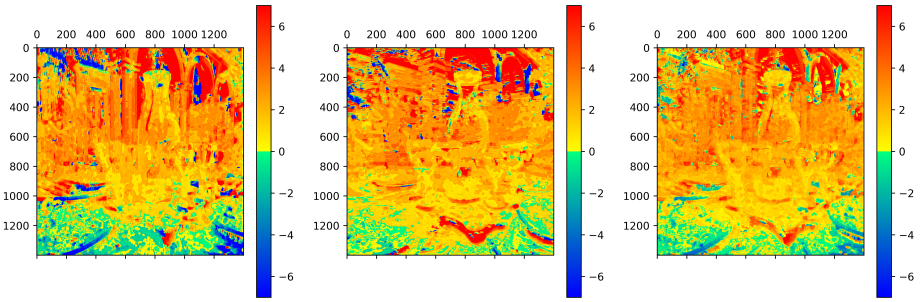

Figure 13. Exemple of depth map using block matching algorithm. Left: average of horizontal and vertical disparities. Center : horizontal disparities only. Right: vertical disparities only.

5. CONCLUSION

In this paper we recalled our previous work on quad-pixel simulations and described a model of the blur function which impacts the Sub Aperture Images. We also explained how we mixed it to a ray-tracing rendering software to generate a dataset. We used this dataset to investigate what was possible to achieve in terms of image processing and deep learning. We demonstrate that quad-pixel image demosaicing is a complex case of multiframe demosaicing and is improved over single image demosaicing. Also our demosaicing solution relies on network of reduced size compared to single image demosaicing. Deep learning applied to diffraction removal gives interesting results and needs to be pushed further. As for the depth estimation, we were not able to reproduce the results from the previous papers on dual-pixel. We think it is mainly due to our dataset, 35 images are too few to properly feed relatively big networks. Generally speaking, the generation of synthetic images with our method is too long to produce a big amount of different images, and the available scenes for PBRT v2 may not be detailed and various enough. That's why a prototype is being made and we hope to use it to produce a better and real dataset.

REFERENCES

[1] Kobayashi, M., Ohmura, M., Takahashi, H., Shirai, T., Sakurai, K., Ichikawa, T., Yuzurihara, H., and Inoue, S., "High-definition and high-sensitivity cmos image sensor with all-pixel image plane phase-difference detection autofocus," *Japanese Journal of Applied Physics* **57**(10), 1002B5 (2018).

[2] Choi, S., Lee, K., Yun, J., Choi, S., Lee, S., Park, J., Shim, E. S., Pyo, J., Kim, B., Jung, M., Lee, Y., Son, K., Jung, S., Wang, T., Choi, Y., Min, D., Im, J., Moon, C., Lee, D., and Chang, D., "An all pixel pdaf cmos image sensor with 0.64umx1.28um photodiode separated by self-aligned in-pixel deep trench isolation for high af performance," in [*2017 Symposium on VLSI Technology*], T104–T105 (June 2017).

[3] Yun, J.-H., Lee, K., Pyo, J., Lee, K., Lee, S., Fujita, M., Mok, K., Son, Yang, J., Song, Y., Kim, H., Park, Y., Choi, S., Shim, E., Jeongjin, Cho, Lee, S., Yoon, S., Jung, S., Nagano, T., Moon, C., and Park, Y.-I., "A small-size dual pixel cmos image sensor with vertically broad photodiode of 0.61μm pitch," in [*International Image Sensor Workshop*], (2019).

[4] Wadhwa, N., Garg, R., Jacobs, D. E., Feldman, B. E., Kanazawa, N., Carroll, R., Movshovitz-Attias, Y., Barron, J. T., Pritch, Y., and Levoy, M., "Synthetic depth-of-field with a single-camera mobile phone," *ACM Trans. Graph.* **37** (July 2018).

[5] Garg, R., Wadhwa, N., Ansari, S., and Barron, J., "Learning single camera depth estimation using dual-pixels," in [*2019 IEEE/CVF International Conference on Computer Vision (ICCV)*], 7627–7636 (2019).

[6] Zhang, Y., Wadhwa, N., Orts-Escolano, S., Häne, C., Fanello, S., and Garg, R., "Du2net: Learning depth estimation from dual-cameras and dual-pixels," in [*Computer Vision – ECCV 2020*], Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M., eds., 582–598, Springer International Publishing, Cham (2020).

[7] Chataignier, G., Vandame, B., and Vaillant, J., "Joint electromagnetic and ray-tracing simulations for quad-pixel sensor and computational imaging," *Opt. Express* **27**, 30486–30501 (Oct 2019).

[8] "Pbrt v2." https://www.pbrt.org/ (2013).

[9] Punnappurath, A., Abuolaim, A., Afifi, M., and Brown, M. S., "Modeling defocus-disparity in dual-pixel sensors," in [*2020 IEEE International Conference on Computational Photography (ICCP)*], 1–12 (2020).

[10] Syu, N., Chen, Y., and Chuang, Y., "Learning deep convolutional networks for demosaicing," *CoRR* **abs/1802.03769** (2018).

[11] Verma, D., Kumar, M., and Eregala, S., "Deep demosaicing using resnet-bottleneck architecture," in [*Computer Vision and Image Processing*], Nain, N., Vipparthi, S. K., and Raman, B., eds., 170–179, Springer Singapore, Singapore (2020).

[12] Ronneberger, O., Fischer, P., and Brox, T., "U-net: Convolutional networks for biomedical image segmentation," in [*Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*], Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., eds., 234–241, Springer International Publishing, Cham (2015).