Burst-shot based Image Selection

Ankur Khosla
2018meb1211
IIT Ropar
Rupnagar, Punjab

Abstract

The aim of this report is to present the final project on the topic - Burst Shot based Image Selection. The idea and architecture of the project was inspired by the previous research works. Core target of this project was to process a sequence of burst shot images, and select a final 'representative' image from the burst shot. The dataset used was HDR+ hosted by Google on Google console.

Siamese based model architecture was built to create a local ranking platform for every incoming pair of images. Each image will compete with its counterpart, and finally a single image is selected from the Burst Shot. By the end of the report, we would be able to understand the pros and cons of the architecture used in this project, and compare the results using MSE, SSIM, etc. Further, the discussion section provides some insights on how we can improve this model and lay out possible applications of this project.

1 Introduction

Almost all the image capturing devices provide the liberty of capturing burst-shot based image sequences. Multiple images are captured within a short duration of time, such that the user can select the best image manually. However, multiple attempts have been made to automate this process of selecting the best image from a continuous sequence. Given the application of this problem, major tech giants such as Microsoft, Google and Facebook have made attempts to solve it. Few of these research labs have successfully demonstrated their approaches and achieved an accuracy around 65%. Selecting the best image from a burst-shot based sequence depends on several factors, such as – low-level features (like Blur, exposure) and high-level features (facial expressions and head pose). In some experiments attempts were made to encode all these features at once and develop a ranking based mechanism. In other experiments, CNNs / Kernels were used for Burst-denoising.

Before proceeding forward, we need to define our problem statement. Previous research works have focused on denoising an image, using the burst shot sequence, enhancing overall quality of the image, generating super-resolution from burst shot and generating a combined image from all the images in burst shot sequence. All these are applications of Burst shot image sequence. But our major focus in this project is to select the best image from a captured burst shot sequence. The definition of 'best' may vary from person to person, but for simplicity, we could define it as an image with more features visible, and less blurriness. In case of Burst Shot sequence, the images are near to each other. There could be some translation and small rotation effect visible due to change in camera angle. This point was leveraged in this project, for creating augmented images while training. In case if the images

are very close to each other, we can track high-level features, such as head pose, expression in the image, etc. The architecture defined in this project was trained to learn these features in a generalized manner, but further improvements in this arena are definitely possible.

2 Previous Research works

Multiple attempts have been made to solve this problem. But all the attempts do not focus on a single application of Burst Shot sequence. Researchers have used Burst Shot sequence to either denoise an image, or to generate Super-resolution from an image, or to merge all the images in the sequence, and create a new thumbnail for burst-shot. Analyzing all the applications, we could develop a broad understanding about the methods through which we can track, generate and edit features in a burst shot sequence.

In the 'real-time burst photo selection using light head adversarial networks', authors argued that if we compare each pair of images, the complexity of the algorithm might go up to O(n2). This would impact the real time performance of the model. To solve this problem, authors proposed an architecture to generate a goodness score for all the images in burst sequence, in one shot. Image with the highest 'goodness' score was selected. The model was an extension to previous research work, and used generators in the process. Decreasing the performance time had a trade-off in terms of the complexity of architecture.

Huang et al described an architecture for learning multi-scale attentive features. Liu and Mildenhall focused on burst denoising using simple convolution neural networks and kernel prediction networks. Few of the papers, such as 'Super Resolution from Image Sequence' discuss how we can track and expand features in a burst shot sequence, to generate super resolution images.

Finally, the two most relevant works for this project were 'Burst photography on High dynamic range and low-lighting imaging in mobile cameras' by Google and 'Automatic Triage for photo series' by Princeton university and Adobe research. The former research used burst shot image sequence, captured from mobile device, and via transformation techniques, such as full resolution align and merge, white balance, local tone map, dehaze, sharpening and tinkering with hue and saturation, generated an upscaled final image. All the sequences used in this research work were hosted on Google console as HDR+ dataset. This HDR+ dataset was used in this project for training the model. The latter research paper provided a Siamese-based approach on selecting the best image from a sequence. But the labeling and performance tracking in this case were performed by humans and this induced some level of bias/manual error.

3 Dataset

Dataset used in this project is HDR+ dataset.

LINK - https://ai.googleblog.com/2018/02/introducing-hdr-burst-photography.html

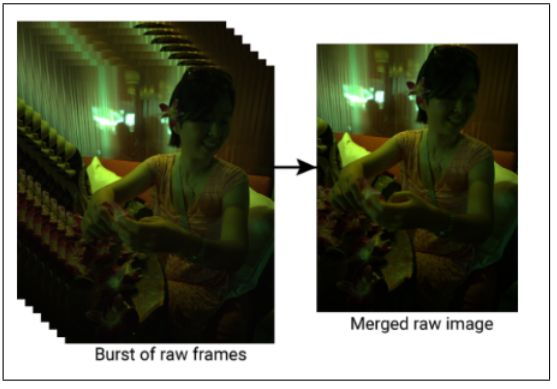

Figure 1: An exemplar image from burst shot sequence

The google console has 2 folders - a subset of the dataset, and complete dataset. Subset dataset consists of 153 burst shot sequences (37 GiB) and the complete dataset has 3640 sequences, and a total of 28461 images. Complete dataset size was 761 GiB. Since the dataset was humongous, we limited our experiments on a subset of the data (LINK).

The subset folder has 4 folders - Burst, gallery and two results folders. We will use the Burst folder and any results folder. The Burst folder further has folders with specific names. Each of this folder has a .tiff images, .dng images (namely payload_XXX.dng) and two additional files, describing the specifications of the mobile capturing device. Results folder has sub-folders with names corresponding to burst-shot sequence. Each sub-folder has merged.dng and final.dng images in it. Merged.dng is the upscaled image, and will be used as the final representative image of our burst shot sequence.

In a nutshell, we have a burst shot sequence, and a final representative image for each sequence. We will use these pairs to train our model. Since the data was hosted on google cloud, the data was downloaded on a local machine using gsutil command. Once all the dataset was downloaded, python scripts were used to generate .txt files for names of all the burst shot sequences. These python scripts were used to download only required images (.dng images) on the GPU machine. Handling such a huge dataset was a bit of a complicated task, but once set, the pipeline was built to process the dataset. (All these python scripts are attached along with code).

Burst shot sequence images have resolution of 4208 x 3120 and final image was 640 x 480. For training the mode, burst shot images were downscaled (since upscaling the final image would lead to unnecessary extrapolation and copying of features, which would ultimately create degraded quality of image).

4 Model Architecture

Some characteristics of a ‘good’ image are low blurriness, more sharp features visible, adequate lighting conditions, and some high level features, such as expressions on the face. We

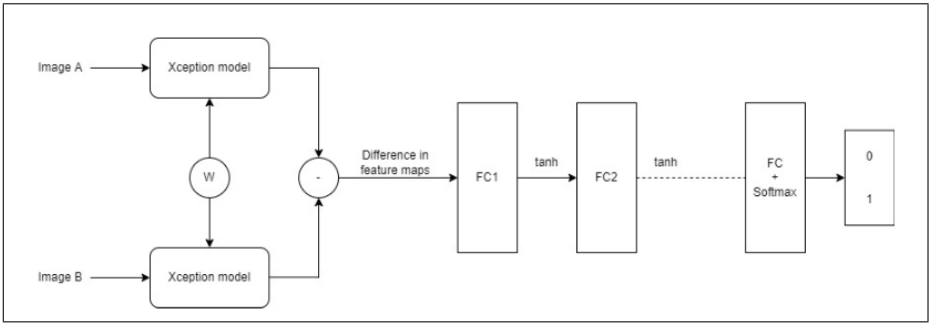

Figure 2: Basic model configuration

can track each of these features with conventional image processing techniques and feature descriptors such as HOGs can oriented gradients in a localized portion of image. But designing a comprehensive model for all the plausible features is a challenging task. We can develop a hard-coded algorithm for image burst shot sequences. But this would hammer the generalizability of the algorithm.

Hence, we focused on CNN based approach to select the best frame from a sequence. The idea is to use Siamese architecture in the model. Input to the model is the pair of images. Final output of the model is a 2x1 vector indicating probabilities, which image should be selected. Baseline model architecture is shown below, however different configurations of the same model were experimented with in the subsequent sections.

Imagenet weights were used in the feature extractor part. MobileNetV2 was also used to observe the results. [0, 1] denotes that the second input image was selected, and [1, 0] was used to denote the first image (image A) is preferred.

For data split, the data was distributed into test, train and validation manually (files attached with code). Out of 153 burst shot sequences, 4 burst shot sequences are used for testing the model. Performance metrics were tracked on each of the sequences. To understand the validation and training split, we need to look into the input form of the model.

If all the burst shot images are provided input as image A, and final image is always image B, the corresponding output will be [0, 1]. This will create a bias inside the model. Irrespective the order of images given as input to model, model will always output [0, 1]. To prevent this bias effect, we shuffle the burst shot sequence and final image on a random basis, with probability 0.5.

For example, if a sequence has 4 burst shot images, i.e. i1, i2, i3 and i4 and final image is denoted by f1, the one possible combination of four pairs of input to the model are shown in table 1.

After mixing data, we have 1319 such pairs of images. 15 such pairs were assigned as validation sets, and the remaining 1304 pairs were marked as training data. Names of images in training, test and validation data were read from a .txt file (txt files and code to generate the same are attached along the project).

Input	Output
i1 , f1	0 , 1
f1 , i2	1 , 0
f1 , i3	1 , 0
i4 , f1	0 , 1

Table 1: Possible combination of training dataset

4.1 Image Augmentation

Inside the Image Data Generator part of the code (_getitem_), we used random image augmentations. Three augmentations were used - blurring, translation and rotation (up to 5 degrees). All these augmentations were performed on burst shot sequence images. Further this image was paired with the final image and fed to the network. These augmentations were performed on a random basis with probability 0.5. Hence, it is possible that one image is blurred and rotated, but not translated. Or other images are translated and rotated but not blurred. This induced some level of additional variance into the training data during successive epochs.

As defined in Siamese architecture, the feature extractor part of the pipeline shared common weights. In one of the experiments, this part was even kept non-trainable after importing imagenet weights.

After the feature extraction part, the output of the feature extractors was subtracted (this part was also experimented with different types of subtraction techniques). The vector thus generated was fed to a sequence of Fully-connected (FC) layers. FC layers have tanh as their activation function, since it was known to perform well in the similar setting in photo-triage paper. Finally, the last layer had softmax activation function because we are considering probabilities of which input image is better. Hence, it makes sense that the sum of output probabilities should add up to 1. For example, output of [0.67, 0.33] would indicate that the first image (image A) is better, whereas, [0.33, 0.67] would denote the opposite. Cross-entropy loss function was used during training and the optimizer was Adam. Learning Rate was kept 0.001 initially, and ReduceLRonPlateau was used in the callback functions.

The basic idea behind designing this pipeline was to create a competing platform for incoming images in sequence, and the image with highest probability output will be declared as the winner or representative image of burst shot sequence.

Later part of the code represents the testing pipeline for test burst shot sequences. For example, if the burst shot sequence has image A, B and C as input images, the testing pipeline will perform the following operations.

The final winner image is compared against the final image in the test dataset. The performance metrics used in this process are listed below:

1. Mean Squared Error (MSE) - Ideal MSE for identical images should be 0. Greater the MSE, greater is the difference between the images.

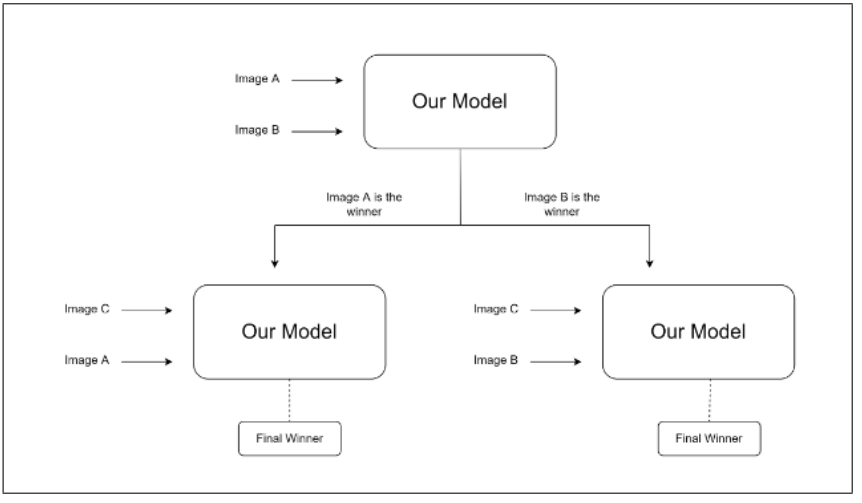

Figure 3: Burst-shot sequence testing algorithm

2. Root Mean Squared Error (RMSE) - Ideal RMSE for identical images should be 0. Greater the MSE, greater is the difference between the images.

3. Peak Signal-to-Noise Ratio (PSNR) - Higher the PSNR between two images, better the quality of comparing images. MSE is used in computation of PSNR.

4. Structural Similarity Index (SSIM) - MSE and PSNR target absolute errors. SSIM is based on the notion that pixels have strong interdependencies, especially when they are close to each other. Ideal SSIM is [1, 1].

5. Universal Quality Image Index (UQI) - Ideal UQI for matching images should be 1.

6. Multi-scale Structural Similarity Index (MS-SSIM) - Ideal MS-SSIM for matching images is 1.

7. Erreur Relative Globale Adimensionnelle de Synthèse (ERGAS) - Idéal ERGAS is 0.

8. Spatial Correlation Coefficient (SCC) - It tracks the spatial variation in the images. Ideal SCC for matching images is 1.

9. Relative Average Spectral Error (RASE) - Ideal error is 0.

10. Spectral Angle Mapper (SAM) - Ideal value for matching images is 0.

11. Visual Information Fidelity (VIF) - Ideal value for VIF is 1.

5 Experiments

The first experiment configuration pipeline is shown in Figure 4.

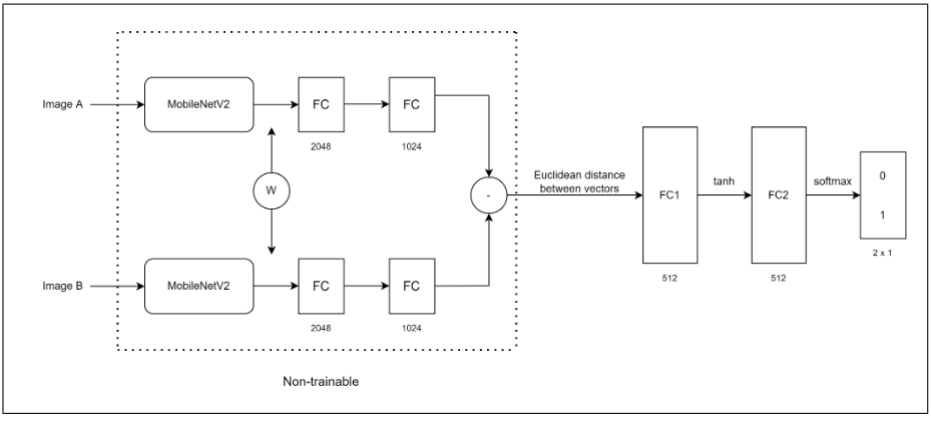

Figure 4: Pipeline for experiment 1

```
Epoch 57/60
652/652 [==============================] - 3123s 5s/step - loss: 0.5987 - accuracy: 0.7255 - val_loss: 3.7480 - val_accuracy: 0.3333 - lr: 1.0000e-06
Epoch 58/60
652/652 [==============================] - 3111s 5s/step - loss: 0.5866 - accuracy: 0.7239 - val_loss: 3.7106 - val_accuracy: 0.4000 - lr: 1.0000e-06
Epoch 59/60
652/652 [==============================] - 3100s 5s/step - loss: 0.5766 - accuracy: 0.7584 - val_loss: 3.7590 - val_accuracy: 0.4000 - lr: 1.0000e-06
Epoch 60/60
652/652 [==============================] - 3097s 5s/step - loss: 0.5688 - accuracy: 0.7615 - val_loss: 3.8190 - val_accuracy: 0.4000 - lr: 1.0000e-06
```

Figure 5: Epochs for experiment 1

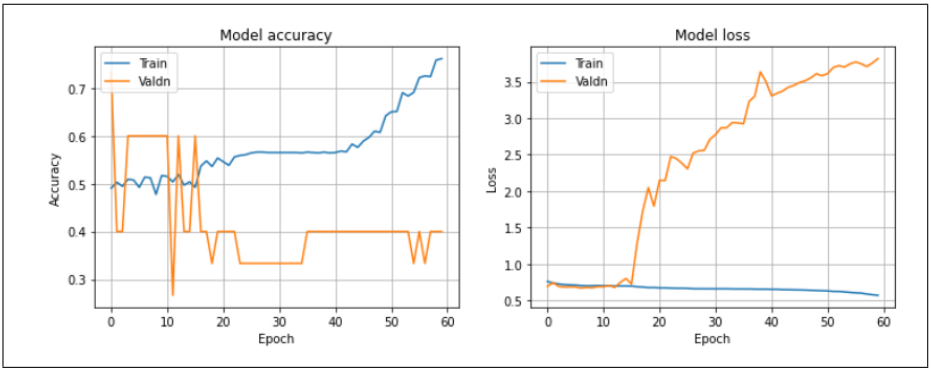

Figure 6: Accuracy and Loss curves, experiment 1

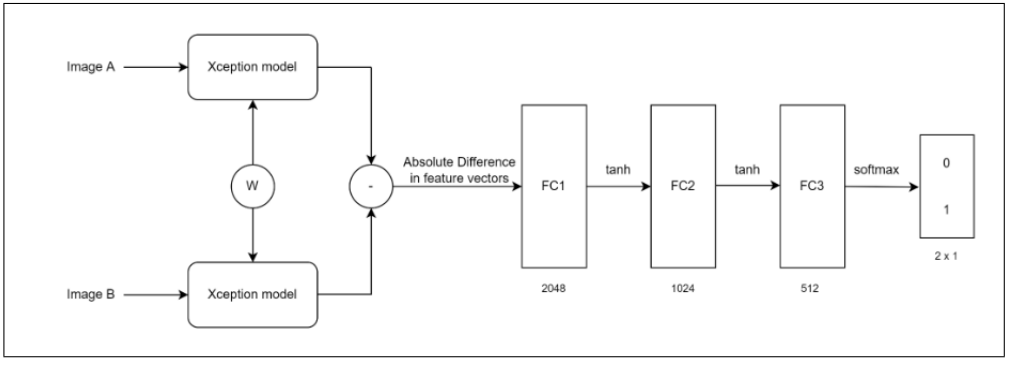

Figure 7: Pipeline for experiment 2

```
Epoch 4/6
1304/1304 [==============================] - 2384s 2s/step - loss: 0.7370 - accuracy: 0.6281 - val_loss: 0.8041 -
val_accuracy: 0.1333
Epoch 5/6
1304/1304 [==============================] - 2392s 2s/step - loss: 0.8163 - accuracy: 0.5439 - val_loss: 0.6023 -
val_accuracy: 0.9333
Epoch 6/6
1304/1304 [==============================] - 2377s 2s/step - loss: 0.7428 - accuracy: 0.5907 - val_loss: 0.8202 -
val_accuracy: 0.1333
```

Figure 8: Epochs for experiment 2

The Light architecture of MobileNetV2 was selected to speed up the training process. Input images were resized to 512 x 512. The training epochs are shown in Figure 5.

The pipeline used for the second experiment is shown in Figure 7.

Input image is kept 640 x 480 in this experiment. Instead of euclidean distance, absolute difference between feature vectors was used in this case. This was because of the fact that in euclidean distance, we might lose the order of input vectors. For example: If the input vectors are A = [3, -4] and B = [2, -3], (A - B) euclidean is sqrt(2) and (B - A) euclidean is also sqrt(2). A-B and B-A both produce the same result. Hence, taking net difference could help us generate a vector, which would help to differentiate between A-B and B-A. Further, a neural network is trained to predict [0, 1] or [1, 0] based on the vector.

6 Results

Results of the first experiment are mentioned in Table 2, and second experiment are mentioned in Table 3.

7 Conclusion and future scope

There is a huge scope of experiments and modifications possible on the current model. The training part of this model was limited by the resources (training 60-70 epochs took almost

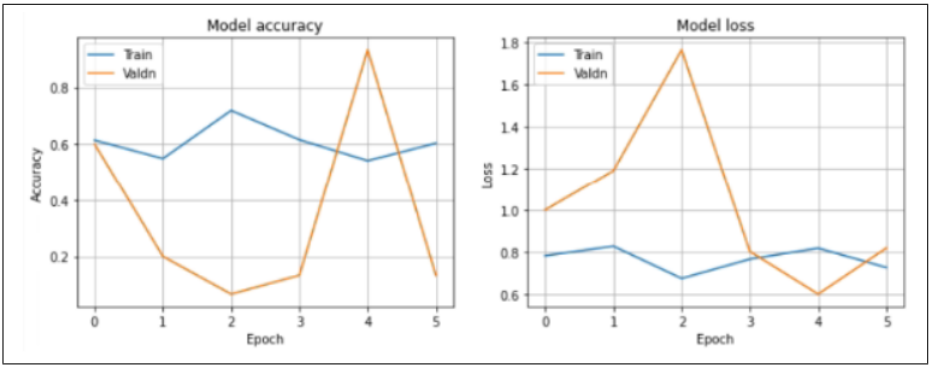

Figure 9: Accuracy and Loss curves, experiment 2

Metric	Case 0	Case 1	Case 2	Case 3
MSE	93.30	2822.59	10.35	1151.30
RMSE	9.65	53.12	3.21	33.93
PSNR	28.43	13.62	37.97	17.51
SSIM	(0.54, 0.54)	(0.30, 0.41)	(0.90, 0.906)	(0.53, 0.57)
UQI	0.87	0.54	0.99	0.87
MSSSIM	(0.8+0j)	(0.4+0j)	(0.9+0j)	(0.5+0j)
ERGAS	16121	30917	1830	18975
SCC	0.03	0.02	0.64	0.04
RASE	2228	4246	265	2736
SAM	0.19	0.5	0.025	0.25
VIF	0.07	0.02	0.43	0.01

Table 2: Results for experiment 1

Metric	Case 0	Case 1	Case 2	Case 3
MSE	266.69	816.15	82.96	697.10
RMSE	16.33	28.567	9.10	26.40
PSNR	23.87	19.01	28.94	19.69
SSIM	(0.40, 0.41)	(0.36, 0.38)	(0.72, 0.72)	(0.55, 0.56)
UQI	0.82	0.75	0.99	0.92
MSSSIM	(0.6+0j)	(0.5+0j)	(0.8+0j)	(0.5+0j)
ERGAS	22114	33635	3840	12456
SCC	0.01	0.01	0.09	0.04
RASE	3021	4603	550	1795
SAM	0.29	0.50	0.07	0.19
VIF	0.01	0.02	0.10	0.019

Table 3: Results for experiment 2

five days on the lab system). To resize input images, we could use an encoder, which could also be trained. This would help us focus on the most important features while training the model.

Overall the approach seems promising. Although the results are not very optimal, overall architecture can be extended to fit the more generalized dataset. One improvement could be to use ranking loss on top of contrastive loss as loss function. This will model the statement as a ranking problem. Another possible edit, which could be done in this regard is the image normalization during pre-processing. Several images in the dataset had dark contrast and hence, an image normalization might have helped. Further tuning of the hyper-parameters can help to improve model performance. Final training accuracy observed was 70%, validation was 55%-60%. (In some epochs, validation accuracy might be low, due to less number of examples assigned to the validation set. This problem could be resolved using the complete dataset at once).

Further this model can be fine-tuned to focus on specific distinguishing features in two parallel images. It can be an extended application of this architecture. Adding a generator to a baseline model like the one presented in this project, can also help improve the results (as shown by Wang et al.)

References

1. Chu, Wei-Ta, and Chia-Hung Lin. "Automatic selection of representative photo and smart thumbnailing using near-duplicate detection." In Proceedings of the 16th ACM international conference on Multimedia, pp. 829-832. 2008.

2. Irani, Michal, and Shmuel Peleg. "Super resolution from image sequences." In [1990] Proceedings. 10th International Conference on Pattern Recognition, vol. 2, pp. 115-120. IEEE, 1990.

3. Liu, Ziwei, Lu Yuan, Xiaoou Tang, Matt Uyttendaele, and Jian Sun. "Fast burst images denoising." ACM Transactions on Graphics (TOG) 33, no. 6 (2014): 1-9.

4. Huang, Jin, Chaoran Cui, Chunyun Zhang, Zhen Shen, Jun Yu, and Yilong Yin. "Learning Multi-Scale Attentive Features for Series Photo Selection." In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2742-2746. IEEE, 2020.

5. Mildenhall, Ben, Jonathan T. Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. "Burst denoising with kernel prediction networks." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2502-2510. 2018.

6. Godard, Clément, Kevin Matzen, and Matt Uyttendaele. "Deep burst denoising." In Proceedings of the European conference on computer vision (ECCV), pp. 538-554. 2018.

7. Wang, Baoyuan, Noranart Vesdapunt, Utkarsh Sinha, and Lei Zhang. "Real-time burst photo selection using a light-head adversarial network." IEEE Transactions on Image Processing 29 (2019): 3065-3077.

8. Hasinoff, Samuel W., Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T. Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. "Burst photography for high dynamic range and low-light imaging on mobile cameras." ACM Transactions on Graphics (ToG) 35, no. 6 (2016): 1-12.

9. Chang, Huiwen, Fisher Yu, Jue Wang, Douglas Ashley, and Adam Finkelstein. "Automatic triage for a photo series." ACM Transactions on Graphics (TOG) 35, no. 4 (2016): 1-10.