Multi-image restoration for computational photography and videography

Antoine Monod

Université Paris Cité

GoPro

UNIVERSITÉ PARIS CITÉ

École Doctorale de Sciences Mathématiques de Paris Centre (ED386)

Laboratoire MAP5 (CNRS UMR 8145)

Multi-image restoration for computational photography and videography

Restauration multi-images pour la photographie et vidéographie computationnelles

Par Antoine MONOD

Thèse de doctorat de mathématiques appliquées

Dirigée par Julie DELON

Présentée et soutenue publiquement le 16 Mai 2023

Devant un jury composé de :

Marcelo BERTALMIO	DR	Spanish National Research Council	Rapporteur
David TSCHUMPERLÉ	DR	Université de Caen Normandie	Rapporteur
Andrés ALMANSA	DR CNRS	Université Paris Cité	Examinateur
Gabriele FACCIOLO	PU	ENS Paris-Saclay	Examinateur
Arthur LECLAIRE	MCF	Université de Bordeaux	Examinateur
Matias TASSANO	PhD	Meta Inc.	Invité
Thomas Veit	PhD	Xiaomi Technology	Invité
Eva COUPETÉ	PhD	GoPro Technology France	Co-encadrante
Julie DELON	PU	Université Paris Cité	Directrice de thèse

Title: Multi-image restoration for computational photography and videography

Abstract: This manuscript explores multi-image approaches for image as well as video restoration. Image restoration using multiple images is studied through real raw burst denoising, where multiple images corrupted with real noise caused by the photographic acquisition process are combined to produce a single image with less noise. A popular classical algorithmic approach of the raw burst denoising literature is thoroughly explained, analyzed and reimplemented in an open source fashion. Given the increasing ubiquity of deep neural networks and their state-of-the-art performance for image and video restoration over the last half-decade, a second learning-based and data-driven approach is proposed for multi-frame raw burst denoising. Comparing the two clearly shows the appeal of properly designed convolutional neural networks for this kind of task. Video restoration can also leverage mutual information of multiple images to produce more appealing results. This topic is explored here via a Deep Plug-and-Play (PnP) method. Such methods consist in plugging a denoising deep neural network in an optimization scheme used to solve inverse problems, *e.g.* using the denoising network as a replacement for the proximal operator of the data prior under a Bayesian formalism. While Plug-and-Play methods have extensively been studied for image restoration, their use in video restoration is relatively uncharted territory, and is a key focus of this manuscript. This manuscript presents a novel method for restoring digital videos via a Deep Plug-and-Play approach. With it, a network trained once for denoising can be repurposed for multiple different video restoration tasks such as video deblurring, super-resolution, demosaicking and interpolation of random missing pixels. Our experiments all show a clear benefit to using a network specifically designed for video denoising, as it yields better restoration performance and better temporal stability than a single image network with similar denoising performance using the same PnP formulation. Said method compares favorably to applying other state-of-the-art PnP schemes separately on each frame of the sequence, opening new perspectives in the field of video restoration.

Keywords: Computational photography, image restoration, video restoration, raw burst denoising, photography pipeline, deep learning, Convolutional Neural Network (CNN), Bayesian imaging, proximal methods, Plug-and-Play (PnP)

Titre : Restauration multi-images pour la photographie et vidéographie computationnelles

Résumé : Ce manuscrit explore les approches multi-images pour la restauration d'images et de vidéos. La restauration d'image à l'aide d'images multiples est étudiée à travers le débruitage de rafales brutes réelles, où plusieurs images corrompues par du bruit réel causé par le processus d'acquisition photographique sont combinées pour produire une seule image avec moins de bruit. Une approche algorithmique classique populaire de la littérature sur le débruitage de rafales est expliquée en détail, analysée et réimplémentée en open source. Compte tenu de l'omniprésence croissante des réseaux de neurones profonds et de leurs performances de l'état de l'art pour la restauration d'images et de vidéos au cours de la dernière demi-décennie, une deuxième approche basée apprentissage et guidée par les données est proposée pour le débruitage de rafales brutes. La comparaison des deux approches montre clairement l'intérêt accru pour ce type de tâche des réseaux de neurone convolutifs correctement conçus. La restauration de vidéos peut également tirer parti de l'information mutuelle de plusieurs images pour produire des résultats plus attrayants. Ce sujet est exploré ici via une méthode Deep Plug-and-Play (PnP). Ces méthodes consistent à introduire un réseau de neurones profond de débruitage dans un schéma d'optimisation utilisé pour résoudre des problèmes inverses, par exemple en utilisant le réseau de débruitage pour remplacer l'opérateur proximal de l'a priori sur les données dans un formalisme Bayésien. Alors que les méthodes Plug-and-Play ont été largement étudiées pour la restauration d'images, leur utilisation en restauration de vidéos est un territoire relativement inexploré, et constitue un élément clé de ce manuscrit. Ce manuscrit présente une nouvelle méthode de restauration de vidéos numériques via une approche Deep Plug-and-Play. Avec cette approche, un réseau entraîné une fois pour du débruitage peut être réutilisé pour de multiples tâches différentes de restauration vidéo telles que le défloutage vidéo, la super-résolution, le dématriçage et l'interpolation de pixels manquants aléatoires. Toutes nos expériences montrent un avantage clair à utiliser un réseau spécifiquement conçu pour le débruitage vidéo, car il donne de meilleures performances de restauration et une meilleure stabilité temporelle qu'un réseau mono-image avec des performances de débruitage similaires utilisant la même formulation PnP. Cette méthode se compare favorablement à l'application d'un autre schéma PnP de l'état de l'art séparément sur chaque image de la séquence, ouvrant de nouvelles perspectives dans le domaine de la restauration vidéo.

Mots-clés : Photographie computationnelle, restauration d'images, restauration de vidéos, débruitage de rafales brutes, pipeline de photographie, apprentissage profond, réseaux de neurones convolutifs, imagerie Bayésienne, méthodes proximales, Plug-and-Play (PnP)

Remerciements

Vous l'aurez constaté, ces lignes figurent parmi les premières (et parfois les seules, selon le degré d'intérêt pour le sujet du manuscrit) rencontrées par le lecteur. Elles font cependant partie des dernières que j'écris au bout de l'exercice ardu et itératif qu'est la rédaction du manuscrit de thèse, et ne sont pas forcément les plus simples ou évidentes à écrire.

Je voudrais tout d'abord remercier Julie Delon, ma directrice de thèse, non pas parce que c'est apprécié ou d'usage, mais parce que c'est indéniablement la personne qui m'a le plus aidé au long de trois en et demi de doctorat, et celle qui a le plus participé à cet ouvrage. Julie, sur le plan scientifique, ta pédagogie, ta capacité à explorer et expliciter un sujet tout en le gardant intéressant, et ton large éventail de connaissances ont été un motivateur et un accélérateur essentiels pour l'étudiant de Master d'ingénierie se lançant dans un doctorat de mathématiques appliquées avec quelques lacunes que j'ai été. En plus de ces qualités dont on t'a probablement déjà fait l'éloge, je voulais aussi te remercier pour les échanges que l'on a pu avoir sur le plan humain. La bienveillance, la gentilesse, les encouragements et l'humour dont tu as fait preuve avec beaucoup de rigueur ont aussi été importants pour que je puisse aller au bout de cet exercice, notamment pendant les moments les plus difficiles, que cela soit du point de vue de la thèse que du contexte extérieur à celle-ci. Au risque de faire hausser quelques sourcils, le souvenir qui m'aura le plus marqué pourrait aussi paraître des plus anecdotiques : partir manifester en ta compagnie (et celle d'Antoine Chambaz) moins de deux mois après le début de notre collaboration. J'aurais bien aimé reconduire cette activité avant la fin de ma thèse pour "boucler la boucle", mais nos contraintes de temps respectives ne l'auront pas permis ; je ne doute pas que l'occasion se représenterera.

Je tiens aussi à remercier Eva Coupeté, Matias Tassano, et Thomas Veit, mes encadrants successifs à GoPro, par ordre chronologique inversé mais par ordre égal d'importance. Bien que l'on pourrait arguer qu'avoir trois responsables industriels différents au cours d'un doctorat n'est pas la panacée en terme de stabilité, telle est la réalité du marché du travail, et je ne me sens pas lésé, au contraire, puisque vous avez tous les trois été sympathiques, curieux et avenants avec moi et m'avez fait profiter de vos expertises légèrement différentes. Puisque l'on est dans le registre professionnel, je remercie aussi Thomas D. et Bruno F., et Gladys, Nicolas, Téo, Moctar, Ludovic, Alban et Yves de l'équipe Granon pour leur sympathie, leur curiosité et leur respect pour mon travail bien qu'il ne fut pas partie des "affaires courantes" de GoPro, et pour leur propre expertise. Je remercie également Violaine et Romain d'avoir bien voulu héberger un humble vagabond sans endroit où dormir dans leur palace berlinois. Enfin, je remercie tout le bureau de GoPro Paris de constituer un endroit où il fait bon travailler, manger, faire des Live It, et faire la fête le 4 Juillet, jour où les américains ne peuvent pas nous en empêcher.

Comment parler de ces trois ans et demi sans mentionner le MAP5, laboratoire de mathématiques appliquées de ~~l'université Paris Descartes~~ ~~l'Université de Paris~~ l'Université Paris Cité ? Je n'exagère rien en disant que ce laboratoire fut une de ces "deuxième maisons" qu'on quitte avec tristesse. Tous ses membres, permanents comme éphémères, contribuent chacun à faire de cet endroit un lieu réconfortant, où la recherche de pointe n'est pas associée à des luttes de pouvoir mais plutôt à de l'échange et du partage de connaissances, le tout dans une bonne ambiance. Merci à Fabienne Comte et à Anne Estrade d'avoir fait perdurer cette philosophie pendant mon séjour en dirigeant le laboratoire. Je tiens aussi à remercier Marie-Hélène Gbaguidi, pour qui je ne suis pas sûr de trouver le qualificatif qui souligne correctement son travail : botte secrète du laboratoire ? Couteau-suisse ? "Unsung hero" ?
Je me dois évidemment de saluer mes chers collègues du bureau 750, passés et présents : Florian, Alan, Safa, Yen, Apolline, Adrien, Chabane, Michel, Elisa, merci d'avoir fait de cet espace somme toute standard un endroit propice au travail, aux siestes et à la rigolade. J'espère que ces remerciements seront conformes au décrêt du tyran autocrate Charlie (que je salue tout de même), permettant ainsi à un exemplaire de ce manuscrit d'éviter un départ prématuré dans une corbeille du bureau.
Au MAP5, la proximité entre les éphémères n'a besoin d'être ni géographique, ni alphabétique, c'est pourquoi je remercie Adélie, Alessandro, Alexander, Alexandre, Andrea, Angie, Antoine S, Anton, Ariane, Arthur, Bernardin, Carlos, Claire, Diala, Eloi, Herb, Ivan, Juliana, Keanu, Laurent, Léonard, Loïc, Marie, Mariem, Martin, Mehdi, Ousmane, Pierre-Louis, Rémi B, Rémi L, Sergio, Sinda, Sonia, Thaïs, Thibaut, Vincent, Warith, Zoé, pas simplement de m'avoir permis d'atteindre 21 lettres sur 26 de l'alphabet avec au moins un prénom de personne par lettre (il ne manquait plus que le G, le N, le Q, le U et le X en comptant les membres du 750), mais aussi pour tous les GTE, les Friday beer (pas que le vendredi), les parties de tarot endiablées (mention obligatoire de https://github.com/eloitanguy/tarotbot), les père Noël surprise et autres repas participatifs, les soirées jeux et films (encore désolé à ceux qui n'ont pas apprécié The Green Knight), le parc Astérix, le ski, et toutes les autres activités que j'aurai oublié. Je n'oublierai certainement pas ces moments passé avec vous, j'espère continuer de vous voir, et que ceux qui y seront encore continueront de faire du MAP5 un lieu dont il est difficile de se défaire.

Merci à Andrés Almansa, Rémi Laumont, Samuel Hurault, José Luis Lisani, Marc Lebrun, Yacine Bouaouni, et Charles Laroche pour leurs échanges scientifico-techniques stimulants au cours de cette thèse, en espérant pouvoir collaborer de nouveau avec au moins quelques uns d'entre vous. Merci également aux administrateurs du supercalculateur Jean-Zay, fleuron national du calcul haute performance, qui fut à la fois un énorme atout technique et une source de satisfaction pour le bidouilleur informatique que je suis.

Rapprochons nous de la fin cette longue série de remerciements en passant par ma pierre angulaire, et ce n'est peut-être pas le plus original, ma famille et mes amis. Merci à Louis, à Simon (tu aurais pu me faire un meilleur rabais que 10% malgré ton statut d'intermittent, mais c'est de bonne guerre), à mon futur ex-coloc Thibault, aux compères Adrien, Alexis, Haris, Marie et Mattias, d'encore me supporter (je crois) après toute ces années. Je pense aussi à mes anciens camarades du CMI, on ne se voit plus tous régulièrement, mais vous avez contribué à la personne que je suis aujourd'hui et aussi fait que ce manuscrit existe. J'embrasse fort les familles Monod/Maître/Poppelaars, Le Sauce et assimilés, vous vous reconnaîtrez, et je vous aime tous. Je remercie aussi tous les autres amis, proches et rencontres qui ne sont pas explicitement nommés mais sont pour autant partie intégrante de ma vie, et dont la non-apparition n'est pas synonyme d'oubli. Je terminerai par dédier ce manuscrit à ma mère Françoise et mon père Didier, sans qui je n'aurais jamais écrit ces mots que vous êtes en train de lire, qui je l'espère les liront un jour, et dont je m'estime chanceux et reconnaissant que cela soit encore possible ; je dédie également ce manuscrit à la mémoire de ma tante Isabelle, qui je le sais aurait eu plaisir à lire ces lignes.

Bonne lecture à tous ceux qui liront au-delà de cette section !

Contents

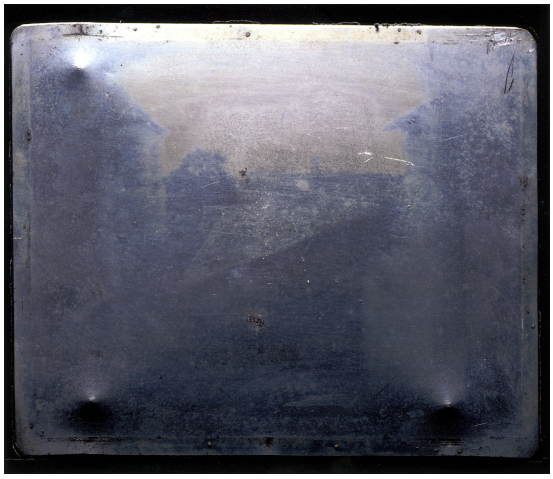
List of Figures

List of Tables

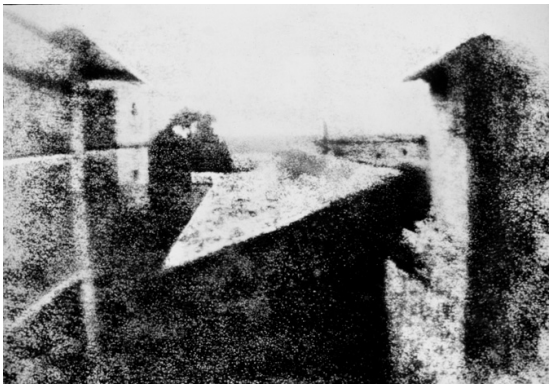
Introduction (FR)

Le compromis est un trait caractéristique de la photographie depuis ses débuts. Le concept de base de la photographie (d'où elle tire son nom) est la capture de la lumière émise par une scène sur une surface. Les compromis dans ce processus sont omniprésents. D'aucuns pourraient se demander : combien de temps doit durer le processus de capture pour que la scène soit correctement visible ? Une réponse possible est que cela dépend de la quantité de lumière émise par la scène pendant un temps donné ; mais cela dépend aussi de la quantité de lumière que le système photographique ou système d'imagerie est capable de capturer pendant ce temps donné ; cela dépend aussi de la luminosité que l'on veut donner au résultat final. On pourrait également se demander : que faut-il capturer exactement ? Une réponse pourrait être de réfléchir soigneusement à la composition de la scène : comme un système d'imagerie a un champ de vision et une résolution limités, tout ne peut pas être capturé en même temps ; si la durée de la capture (le temps d'exposition) est longue, on pourrait vouloir s'assurer que le dispositif d'imagerie (l'appareil photo) et la scène restent immobiles, sans quoi les objets qui bougent pendant le processus de capture apparaîtraient à plusieurs endroits de la surface, les rendant flous. Il est facile d'imaginer ces réflexions sur les compromis jusque dans la tête des tout premiers photographes, comme Nicéphore Nièpce quand il plaça sa camera obscura devant une fenêtre de sa maison de Saint-Loup-de-Varennes pour capturer *Point de vue du Gras* en 1827. La plaque en étain recouverte de bitume photosensible qui en résulte est la plus ancienne photographie conservée. Elle est présentée avec une copie retouchée en Figure 1.

Dans la photographie numérique, les interactions chimiques complexes de son homologue analogique ou argentique sont remplacées par un processus de capture électronique. La principale différence entre les deux est la surface sur laquelle la lumière est capturée : les appareils photo numériques comportent généralement un ensemble matriciel de diodes sensibles à la lumière ou photosites qui génèrent un courant qui est fonction du nombre de photons incidents. Cet ensemble est un composant du système appelé *capteur d'image* qui peut enregistrer une matrice de valeurs numériques à chaque photosite (ou pixel), chaque valeur correspondant au nombre de photons rencontrés par ledit photosite pendant un certain laps de temps. La matrice enregistrée est une image ou photographie numérique. Comme pour la photographie argentique, la photographie numérique comporte son lot de compromis. Certains peuvent découler du processus de formation d'image : lorsque le mouvement de l'appareil photo ou de la scène n'est pas négligeable par rapport



Plaque en étain originale.



La copie retouchée de l'historien Helmut Gernsheim.

FIG. 1 : *Point de vue du Gras*, la première photographie par Nicéphore Nièpce (1827).

au temps d'exposition, certaines parties de l'image peuvent apparaître floues, comme l'illustre la figure 2. Pour éviter le flou d'image, il est préférable d'utiliser des temps d'exposition plus courts ; mais pour compenser la perte de luminosité associée de l'image, il faut augmenter le gain analogique et numérique de l'appareil photo, représentés par le réglage ISO. Cependant, augmenter l'ISO amplifie le bruit de l'image, ce qui est particulièrement visible en situation de faible luminosité. Le bruit d'image numérique est en fait de l'aléa dans les valeurs attendues d'une image numérique. Ses sources sont multiples : le processus d'émission des photons est lui-même un processus aléatoire ; les processus qui sous-tendent la lecture du capteur ont également leur propre lot d'imprécisions ; la conception électrique et thermique de la caméra peut également induire une stochasticité de l'image. La Figure 3 présente un exemple de photographie nocturne avec une GoPro HERO 11 Black où la réduction du temps d'exposition et l'augmentation de la sensibilité ISO amplifient le bruit de l'image.

La conception et la sélection des capteurs d'image impliquent également des compromis. Par exemple, des pixels plus petits emmagasinent moins de lumière et sont plus sensibles au bruit ; on pourrait donc être tenté d'augmenter la taille individuelle des pixels, mais les capteurs avec moins de pixels ont une résolution limitée. Les capteurs plus grands produisent généralement de meilleures images, mais nécessitent des systèmes optiques plus grands et consomment plus d'énergie. Il existe même des compromis liés à la nature numérique des photographies : une meilleure qualité d'image implique généralement des coûts de stockage plus élevés.

La popularité de la photographie numérique a explosé depuis la généralisation de l'utilisation des smartphones dans les années 2010. Des milliards de personnes transportent désormais partout avec elles un appareil doté d'un ou de plusieurs appareils photo numériques. Les caméras d'action sont dotées d'une technologie

ISO 800, temps d'exposition 1/9s ISO 100, temps d'exposition 1s

FIG. 2 : Le flou d'image réel apparait lorsque le mouvement est non négligeable devant le temps d'exposition.

similaire : ces appareils permettent aux utilisateurs de capturer rapidement des événements de leur vie et de partager ces expériences avec un large public, même dans les cas où l'utilisation d'un smartphone pourrait être déconseillée (par exemple, lors de la pratique de sports extrêmes ou de la prise de vue sous l'eau). Comme beaucoup d'autres appareils électroniques grand public, les smartphones et les caméras d'action sont des systèmes d'imagerie qui sont intrinsèquement limités par certaines de leurs exigences de conception :

- Ils doivent utiliser des capteurs et optiques de petite taille pour maintenir un encombrement physique limité. À titre d'exemple, la taille relative du capteur d'une caméra d'action GoPro HERO 11 Black par rapport à celle d'un appareil photo plein format Canon EOS 6D Mark II est illustrée dans la figure 4.

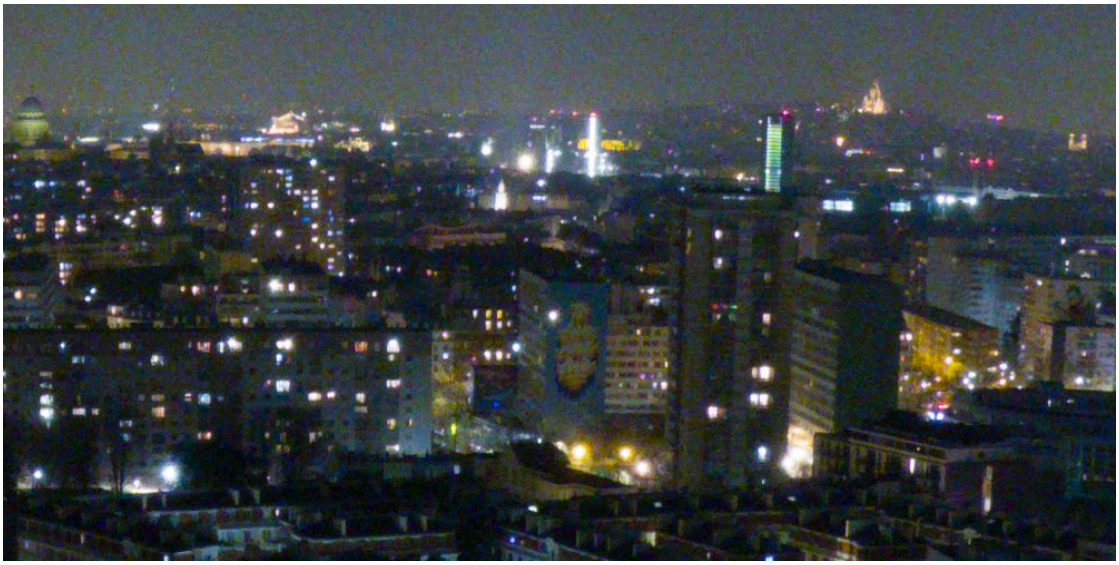
- Ils doivent avoir une consommation d'énergie raisonnable afin que l'appareil puisse être utilisé pendant de longues sessions de capture.

- Ils doivent être performants dans une grande variété de scénarios de capture : à l'intérieur, à l'extérieur, pour des portraits, des paysages, des photos et des vidéos, à différentes résolutions et fréquences d'image, etc.

- Ils doivent être fabriqués avec des pièces d'un coût raisonnable afin de pouvoir être vendus en grandes quantités à des prix acceptables.

Notez que ces contraintes ne sont pas communes à tous les dispositifs d'imagerie numérique : les microscopes, les IRM, les caméras hyperspectrales sur les satellites

ISO 100, temps d'exposition 5s



ISO 800, temps d'exposition 1s

FIG. 3 : Le bruit d'image réel est particulièrement visible en basse lumière à haut ISO. (La longue exposition n'est pas floue parce que l'appareil photo et la scène sont statiques ; les deux images sont des crops 1000×500 d'images brutes de taille 5568×4872 traitées de manière minimale avec Adobe Lightroom).

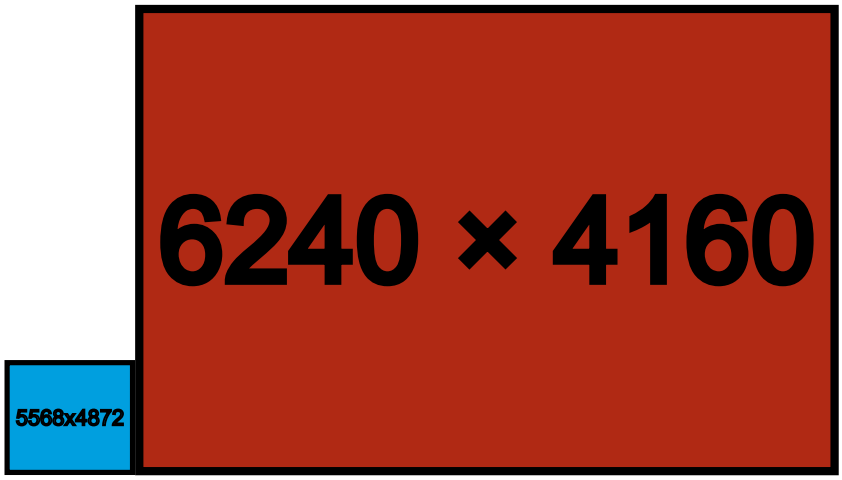


FIG. 4 : Taille relative du capteur 27MP d'une GoPro HERO 11 Black (cyan) par rapport au capteur 26MP d'un Canon EOS 6D Mark II (rouge). La résolution d'image maximale disponible pour chaque caméra est indiquée dans les rectangles.

ou le télescope spatial James Webb ont des exigences complètement différentes.

L'amélioration continue de la puissance de traitement des petits appareils a été le catalyseur de nombreuses avancées dans le domaine de la photographie et de la vidéographie computationnelles, où des algorithmes sont utilisés pour surmonter les propriétés physiques limitées des systèmes d'imagerie. Les algorithmes de restauration ou d'amélioration tels que le débruitage, le défloutage, la super-résolution, l'inpainting ou l'interpolation de trames sont un sujet très récurrent et actif de la littérature scientifique, puisque la dégradation des images est un processus complexe qui produit généralement des problèmes mal posés et sous-contraints. L'imagerie computationnelle a également son propre lot de compromis :

- La maximisation des ressources de calcul pour des algorithmes plus puissants se fait généralement au détriment de l'efficacité énergétique.
- Pour un algorithme donné, la performance maximale va souvent à l'encontre de la polyvalence (c'est-à-dire la bonne performance dans une variété de scénarios) ou la réutilisabilité (c'est-à-dire que la méthode peut être facilement mise à jour ou réutilisée pour des applications différentes).
- Les algorithmes qui doivent être exécutés en temps réel ou "en ligne" (par exemple, pendant la capture d'une vidéo) sont généralement beaucoup plus contraints que ceux qui peuvent être exécutés après la capture ou "hors ligne".
- Même dans un contexte hors ligne, les algorithmes qui peuvent être exécutés directement sur le système d'imagerie ou "on-device" disposent généralement de ressources informatiques limitées. Il est possible de bénéficier de ressources supplémentaires en exécutant ces algorithmes sur du matériel plus puissant

ou “off-device” (par exemple en transférant une photo ou une vidéo vers le cloud et en exécutant un algorithme sur un serveur), mais le processus de transfert et le matériel supplémentaire ont leurs propres coûts et compromis.

Alors que de nombreux algorithmes de restauration n'utilisent que les informations stockées dans l'image d'entrée (on parle généralement de méthodes *mono-image*), d'autres proposent d'exploiter l'information contenue dans des images supplémentaires (par exemple, des séquences vidéo ou des rafales) tout en apportant des contraintes supplémentaires au problème (il s'agit alors d'algorithmes *multi-images*). La question au cœur de ce manuscrit est la suivante : *comment peut-on concevoir des algorithmes qui bénéficient de l'information et des contraintes supplémentaires de plusieurs images pour améliorer la restauration d'images et de vidéos ?* Ce manuscrit apporte une réponse partielle à cette question très ouverte, et suit principalement l'ordre chronologique des travaux menés au cours de trois années de recherche.

Chapitre 1 est une introduction pratique au problème de restauration multi-images de débruitage de rafales brutes réelles par le biais d'une méthode classique “fait main”. Il consiste en une analyse, explication et réimplémentation approfondie de HDR+, un pipeline de traitement d'image conçu par les chercheurs de Google présenté pour la première fois en 2016 dans [Has+16] et intégré dans certains de ses smartphones Pixel. Bien que le pipeline soit décrit en détail dans [Has+16], il s'agit d'un système complexe composé de nombreuses parties, certaines pouvant bénéficier d'observations et d'expériences supplémentaires. La publication n'étant accompagnée d'aucun code source, notre réimplémentation en Python a été mise à la disposition de la communauté scientifique en même temps que notre article d'analyse et s'est révélée assez populaire.
HDR+ étant conçu comme un pipeline de photographie complet, il contient plusieurs étapes de traitement d'image supplémentaires après la partie débruitage de rafales brutes. Bien que ce ne soit pas l'objet principal de ce chapitre, ces étapes constituent une bonne introduction au concept de pipeline d'imagerie lui-même ; chaque fabricant d'appareil (y compris GoPro) a son propre ensemble ordonné et interconnecté d'algorithmes de traitement d'image, avec des principes de conception, des réglages et des compromis spécifiques.

Depuis la fin des années 2010, les algorithmes de restauration d'image traditionnels basés sur des modèles mathématiques et physiques sont de plus en plus surpassés par les méthodes basées sur l'apprentissage. Ces méthodes exploitent de grandes quantités de données pour résoudre des problèmes d'optimisation à grande échelle et apprendre des représentations implicites complexes de ces données. Le Chapitre 2 explore donc le problème du débruitage des rafales brutes réelles de Partie I via une approche apprentissage profond. Il s'appuie sur FastDVDnet [TDV20], une contribution d'un précédent travail de doctorat avec GoPro sur le débruitage vidéo gaussien, et le modifie pour un nouveau cas d'utilisation pratique. Les pipelines d'imagerie font leur retour ici, puisque deux paradigmes sont explorés pour entraîner un modèle de débruitage de rafales brutes réelles : générer des données brutes synthétiques variées à partir d'images traitées

existantes en appliquant les opérations inverses d'un pipeline d'imagerie, ou utiliser des données brutes réelles provenant de scénarios de capture contrôlés mais limités.

Pour en revenir aux compromis dans le domaine de la photographie computationnelle, il ressortira des deux chapitres précédents que les méthodes basées sur des modèles physiques et mathématiques et les méthodes basées sur l'apprentissage ont chacune leurs propres forces et faiblesses. Il n'est donc pas surprenant qu'une part croissante de la recherche dans ce domaine tente d'adopter une approche "meilleur des deux mondes", en tirant parti à la fois de la modélisation explicite du processus de dégradation de l'image et des représentations implicites de plus en plus puissantes apprises à partir de grandes quantités de données. Le Chapitre 3 de la Partie II de ce manuscrit étudie l'une de ces approches pour la restauration vidéo. La composante multi-images de cette thèse prend une tournure légèrement différente avec l'amélioration de vidéos, puisque l'entrée et la sortie de la méthode ont toutes deux plusieurs images, alors que les méthodes de restauration de rafales des chapitres précédents combinent plusieurs images en une seule. L'approche "meilleur des deux mondes" explorée dans ce chapitre est le cadre Plug-and-Play (PnP) : en résumé, il combine un terme d'attache aux données orienté modèle et un terme de régularisation défini par un algorithme de débruitage dans un schéma d'optimisation alterné afin de résoudre des problèmes inverses de restauration. En particulier, le Chapitre 3 se concentre sur le Deep Plug-and-Play, où l'algorithme de débruitage est un réseau de neurones profond. De nombreuses expériences sont menées dans différentes tâches de restauration vidéo telles que la super-résolution, la déconvolution ou l'interpolation de pixels. À notre connaissance, il s'agit du premier travail étudiant l'utilisation du cadre Deep Plug-and-Play pour la restauration de vidéos.

Les principales contributions de cette thèse sont les suivantes :

- Une analyse approfondie du pipeline de photographie en rafale HDR+ est fournie dans [MDV21] et est accompagnée d'une démo interactive[1] et d'une implémentation Python open-source[2].

- Avec RBDnet, nous nous appuyons avec succès sur les fondations de FastDVDnet pour l'application du débruitage de rafales brutes réelles. Des expériences sur les réseaux multi-images versus mono-image, les avantages et inconvénients des données d'entraînement synthétiques et réelles, ainsi que des études d'ablation multiples sont réalisées pour fournir des informations sur la conception et l'entraînement d'un réseau de débruitage de rafales brutes. Notre réseau surpasse HDR+ de manière significative et obtient des résultats acceptables par rapport aux méthodes d'apprentissage profond de l'état de l'art. Nos expériences seront soumises à la revue scientifique en ligne Image Processing On Line, avec une démo interactive et le code open-source.

[1] https://ipolcore.ipol.im/demo/clientApp/demo.html?id=336
[2] https://github.com/amonod/hdrplus-python

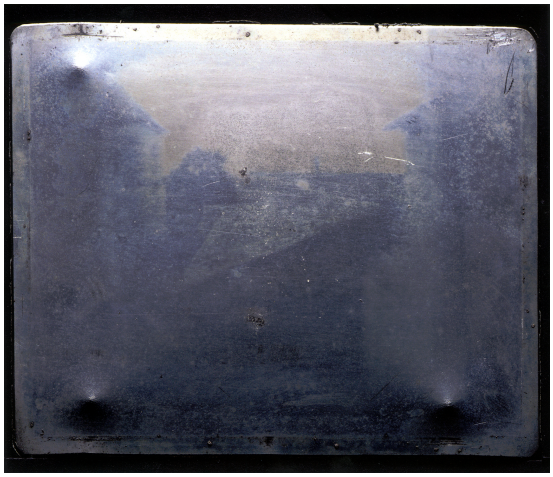
- Nous démontrons la viabilité et la polyvalence du framework Deep Plug-and-Play pour la restauration vidéo, où un seul réseau de débruitage peut être réutilisé pour de multiples tâches de restauration. A travers deux publications [MDT22 ; Mon+22b] et des expériences supplémentaires dans ce manuscrit, nous montrons l'intérêt d'utiliser un réseau profond spécifiquement conçu pour le débruitage vidéo plutôt qu'un débruiteur mono-image, aussi bien en terme de qualité de restauration par image que de stabilité temporelle. Bien qu'une telle comparaison ne soit pas très répandue dans la littérature Plug-and-Play, nous effectuons l'inévitable comparaison de notre méthode avec celles basées apprentissage. Il en ressort que les méthodes Deep PnP sont plus robustes aux changements dans le processus de dégradation sans qu'il soit nécessaire de réentraîner le réseau, mais que les méthodes basées sur l'apprentissage sont nettement plus performantes que les méthodes PnP pour le modèle de dégradation spécifique pour lequel elles ont été entraînées. Toutes nos expériences sont en cours de mise à disposition et seront reproductibles en ligne[3].

[3] https://github.com/amonod/pnp-video

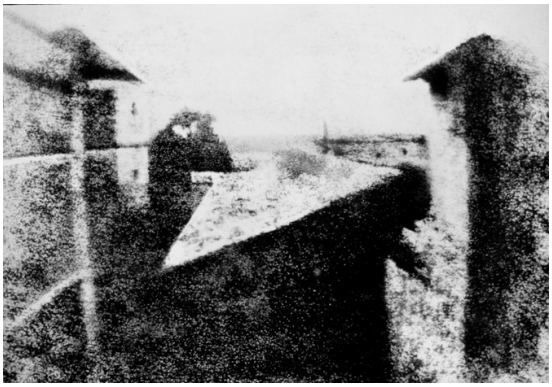
Introduction (EN)

Compromises have been a defining trait of photography since its very beginning. The core concept behind photography (which is where it get its name from) is the capture of the light emitted by a scene on a surface. Compromises in this process are everywhere. One might ask: how long must the capture process last for the scene to be correctly visible? An answer could be that it depends on how much light the scenes emits for a given quantity of time; but it also depends on how much light the photographic or imaging system is able to capture during that given quantity of time; it also depends on how bright one wants the final result to appear. One might also ask: what exactly should be captured? An answer would be to carefully think about scene composition: because an imaging system has a finite field of view and resolution, not everything can be captured at once; if the duration of capture (the exposure time) is long, one might want to make sure both the imaging device (the camera) and the scene remain still, or else objects that move during the capture process will appear at multiple locations of the surface, making them blurry. It is easy to imagine all these thoughts about compromises even in the heads of the very first photograpers, such as Nicéphore Nièpce when he put his camera obscura in front of a window of his house in Saint-Loup-de-Varennes to capture *Point de vue du Gras* in 1827. The resulting tin plaque coated with light-sensitive bitumen is the oldest surviving photograph. It is showcased along with an enhanced copy print in Figure 5.

In digital photography, the complex chemical interactions of its "analog" counterpart are replaced by an electronic capture process. The main difference between the two is the surface on which light is captured: digital cameras typically feature an array of light-sensitive diodes or photosites that generate a current as a function of the number of incident photons. This array is a component of the system called *image sensor* that can record an array of numerical values at each photosite (or pixel), each value corresponding to the number of photons encountered by said photosite during a certain amount of time. The recorded array is a digital image, picture or photograph. As in film photography, digital photography has its set of compromises. Some might stem from the image formation process: when camera or scene motion is not negligible with respect to exposure time, parts of the image can appear blurry as illustrated in Figure 6. To avoid image blur, one might want to use shorter exposure times; but to compensate for the associated loss of image brightness, one must increase the camera analog and digital gain, represented by its ISO setting. However, increasing ISO will amplify image noise, which is partic-



Original tin plaque.



Historian Helmut Gernsheim's enhanced copy print.

Figure 5: *Point de vue du Gras*, the first photograph by Nicéphore Nièpce (1827).

ISO 800, 1/9s exposure time

ISO 100, 1s exposure time

Figure 6: Real image blur occurs when motion is non negligible with respect to exposure time.

ularly visible in low light scenarios. Digital image noise is effectively randomness in the expected values of a digital image. It has multiple sources: the photon emission process itself is a random process; the processes behind sensor readout also have their own set of inaccuracies; the electrical and thermal design of the camera can also induce image stochasticity. Figure 7 showcases an example of night time photography using a GoPro HERO 11 Black where lowering exposure time and increasing ISO amplifies image noise.

Designing and selecting image sensors also implies making compromises. For example, pixels that are smaller receive less light and are more sentitive to noise, so one might be tempted to increase the size of individual pixels; but sensors with fewer pixels have limited resolution. Larger sensors usually produce better images, but require bigger optical systems and consume more electrical power. There are even compromises that arise from the digital nature of the photographs: better image quality usually implies larger storage costs.

The popularity of digital photography has exploded since the generalization of the use of smartphones in the 2010s. Billions of people now carry a device that has one or multiple digital cameras everywhere with them. Similar technology is featured in action cameras: these devices allow users to quickly capture events of their life and share these experiences to a large audience, even in cases where using a smartphone might be ill-avdised (*e.g.* when practicing extreme sports or filming underwater).

As many other consumer electronics devices, smartphones and action cameras are imaging systems that are inherently limited by some of their design requirements:

- They must use small sensors and optics to maintain a limited physical footprint. As an example, the size of the sensor of a GoPro HERO 11 Black action camera relative to the sensor size of a Canon EOS 6D Mark II full-frame camera is illustrated in Figure 8.

- They must have reasonable power consumption so that the device can be used for long capture sessions.

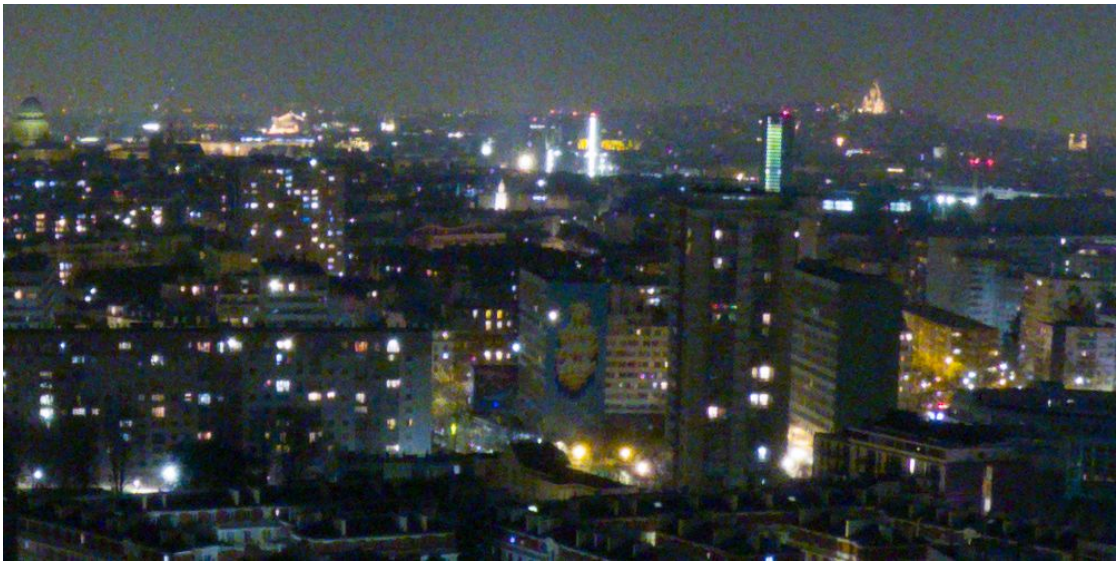
- They must perform well in a wide variety of capture scenarios: indoors, outdoors, for portraits, for landscapes, for photos and videos, at various resolutions and framerates, and so on.

- They must be made with parts that are of reasonable cost so that they can be sold in large quantities at acceptable prices.

Notice that these constraints are not common to all digital imaging devices: microscopes, IRMs, hyperspectral cameras on satellites, or the James Webb Space Telescope have completely different requirements.

Continuous improvement in the processing power of small devices has been the catalyst of many advances in computational photography and videography, where algorithms are used to overcome the limited physical properties of imaging

ISO 100, 5s exposure time



ISO 800, 1s exposure time

Figure 7: Real image noise is particularly visible in low light at high ISO settings. (The long exposure is not blurry because the camera and the scene are static; both images are 1000×500 crops of 5568×4872 raw images minimally processed using Adobe Lightroom.)

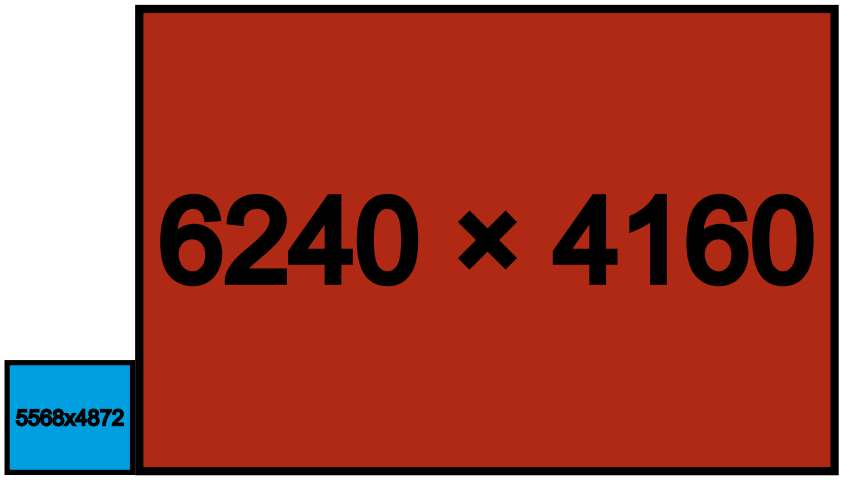


Figure 8: Relative size of the 27MP sensor in a GoPro HERO 11 Black (cyan) compared to the 26MP sensor in a Canon EOS 6D Mark II (red). The maximum image resolution available for each camera is indicated in the rectangles.

systems. Restoration or enhancement algorithms such as denoising, deblurring, super-resolution, inpainting or frame interpolation are a very recurrent and active topic of the scientific literature, as image degradation is a complex process that yields typically ill-posed and under-constrained problems. Computational imaging also has its own set of tradeoffs:

- The maximization of computing resources for more powerful algorithms is typically at the expense of energy efficiency.
- For a given algorithm, maximum performance is often at odds with versatility (*i.e.* performing well on a variety of scenarios) or reusability (*i.e.* the method can easily be updated or repurposed for different applications).
- Algorithms that have to run in real-time or "online" (*e.g.* during the capture of a video) are typically far more constrained that ones that can run after capture or "offline".
- Even in an offline context, algorithms that can run directly on the imaging system or "on-device" usually have limited computing resources. It is possible to benefit from additional resources by running the algorithms on more powerful hardware or "off-device" (*e.g.* by transferring a photo or video to the cloud and running an algorithm on a server), but the transfer process and the extra hardware have their own costs and compromises.

While many restoration algorithms only use the information stored in the input image (they are usually referred to as *single-frame* methods), others suggest leveraging information stored in additional images (*e.g.* video sequences or bursts) while providing additional constraints to the problem (these being *multi-frame* algorithms). The question at the core of this manuscript is the following: *how can*

we design algorithms that benefit from the additional information and constraints of multiple images to improve image and video restoration? This manuscript provides a partial answer to this very open-ended question, and mostly follows the chronological order of the work conducted during three years of research.

Chapter 1 is a hands-on introduction to the multi-frame restoration problem of real raw burst denoising through a classical handcrafted method. It consists of a thorough analysis, explanation and reimplementation of HDR+, an image processing pipeline designed by Google researchers first presented in 2016 in [Has+16] and embedded in some of its Pixel smartphones. While the pipeline is extensively described in [Has+16], it is a complex system made of many parts, some of which could use additional insights and experiments. Since the publication did not come with any source code, our Python reimplementation was made available to the scientific community alongside our analysis article and has shown popularity.
HDR+ being designed as an end-to-end photography pipeline, it contains several additional image processing steps after the raw burst denoising part. Although it is not the main focus of the Chapter, these steps are a good introduction to the concept ot the imaging pipeline itself; every device manufacturer (GoPro included) has their own ordered and interconnected set of image processing algorithms, with specific design principles, tuning, and tradeoffs.

Since the late 2010s, handcrafted and model-based image restoration algorithms are increasingly outperformed by learning-based methods. These methods leverage high amounts of data to solve large-scale optimization problems and learn complex implicit representations of said data. Chapter 2 then explores the real raw burst denoising problem of Part I via a deep learning approach. It builds upon FastDVDnet [TDV20], a contribution of a previous PhD work with GoPro on Gaussian video denoising, and modifies it for a new, practical use case. Imaging pipelines make a return here, as two paradigms are explored to train a real raw burst denoising model: generate varied synthetic raw data from existing processed images by applying the reverse operations of an imaging pipeline, or use real raw data from controlled but limited capture scenarios.

Going back to tradeoffs and compromises in computational photography, it will become apparent from reading the two previous chapters that methods built on physical and mathematical models and learning-based methods each have their ownstrengths and weaknesses. It is no surprise then than an increasing part of the research in the field tries to take a "best of both worlds" approach, leveraging both the explicit modeling of the image degradation process, and the ever-more powerful implicit representations learned from large amounts of data. Chapter 3 of Part II of this manuscript studies one of these approaches for video restoration. The multi-frame component of this thesis takes a slightly different turn with video enhancement, as both the input and the output of the method have multiple frames, while the burst restoration methods of the previous chapters combine multiple images into one. The "best of both worlds" approach explored in this chapter is the Plug-and-Play (PnP) framework: in a nutshell, it combines a model-based data fidelity term and a regularization term defined by a denoising algorithm in an

alternate optimization scheme to solve inverse restoration problems. In particular, Chapter 3 focuses on Deep Plug-and-Play, where the denoising algorithm is a deep neural network. Numerous experiments are conducted in many different video restoration tasks such as super-resolution, deconvolution, or pixel interpolation. To the best of our knowledge, it is the first work studying the use of the Deep Plug-and-Play framework for video restoration.

The main contributions of this thesis are the following:

- An in-depth analysis of the popular HDR+ burst photography pipeline is provided in [MDV21] along with an interactive demo[4] and an open-source Python implementation[5].
- With RBDnet, we successfully build upon the foundations of FastDVDnet for the real-world application of raw burst denoising. Experiments on multi-frame versus single-frame networks, pros and cons of synthetic versus real training data, and multiple ablation studies are performed to provide insights when designing and training a raw burst denoising network. Our network outperforms HDR+ significantly and performs acceptably when compared to state-of-the-art deep learning methods. Our experiments will be submitted to the peer-reviewed Image Processing On Line journal, along with an interactive demo and the open-source code.
- We demonstrate the viability and versatility of the Deep Plug-and-Play framework for video restoration, where a single denoising network can be repurposed for multiple restoration tasks. Through two publications [MDT22; Mon+22b] and additional experiments in this manuscript, we showcase the benefit of using a deep network specifically designed for video denoising instead of a single-frame denoiser both in terms of per-frame restoration quality and temporal stability. Though such a comparison is not very prevalent in the Plug-and-Play literature, we perform the inevitable comparison of our method to state-of-the-art learning-based ones. The takeaway is that Deep PnP methods are more robust to change in the degradation process without the need to retrain the network, but state-of-the-art learning-based methods clearly outperform PnP methods for the specific degradation model they were trained on. All our experiments are in the process of being made available and reproductible online[6].

[4] https://ipolcore.ipol.im/demo/clientApp/demo.html?id=336
[5] https://github.com/amonod/hdrplus-python
[6] https://github.com/amonod/pnp-video

Part I

Raw Burst Denoising

Chapter 1

An Analysis and Implementation of the HDR+ Burst Denoising Method

1.1 Background

Image noise is a common issue in digital photography, and that issue is even more prominent in smaller sensors used in devices like smartphones or action cameras. Algorithms that aim at removing digital noise are a very recurrent topic of the computational photography literature. Most of them only use the information stored in the input image (they are usually referred to as "single-frame" algorithms), while others suggest the use of information stored in additional images (usually videos or bursts). These multi-frame "denoisers" usually make use of similarity measurements in order to combine the information of multiple pixels or patches [Bua+09].

A multi-frame denoising algorithm is at the heart of the HDR+ system. It was first officially introduced as a feature within the Google Camera App in the newly released Nexus 5 and Nexus 6 smartphones in fall 2014[1]. In 2016, a full raw image processing pipeline with the same HDR+ moniker was presented in [Has+16]. This method is the main subject of this chapter. The pipeline was embedded in smartphones such as the Google Nexus 6, 5X and 6P and first generation Google Pixel. Further algorithm refinements and optimizations were included in the Google Pixel 2 which was released in fall 2017. This device was the first to embed HDR+ in a dedicated Google-designed Image Processing Unit[2]. The same raw burst alignment and merging technique was included in a new "Night Sight" mode that was launched alongside the Pixel 3A. The changes and optimizations of this mode were presented in a new publication in 2019 [Lib+19]. As such, the complex HDR+ system is not only part of a detailed, frequently compared and quoted scientific publication, but also embedded into multiple recent mass-produced consumer elec-

[1] https://tinyurl.com/hdrplus-2014
[2] https://tinyurl.com/pixel-visual-core

tronics devices (which have received praise for their camera performance[3]), making it a relevant subject of study.

In this chapter, we will describe the core part of the system, a raw burst denoising algorithm (although it will not be our main focus, we will also mention the rest of the pipeline). This chapter is complemented by a publicly available HDR+ Python implementation (with a simplified finishing pipeline)[4], along with an interactive demo on Image Processing On Line[5]. The Chapter is structured as follows: Section 1.2 presents the design principles of the HDR+ pipeline; Section 1.3 details the burst alignment part of the algorithm; Section 1.4 describes the Wiener-based temporal and spatial denoising in the Fourier space. The rest of the pipeline is described in 1.5, while comparison against denoising results provided by Google is performed in Section 1.6. Section 1.7 showcases running our implementation off-device on real bursts captured by a GoPro camera.

1.2 Characteristics of the HDR+ Pipeline

HDR+ was designed as a consumer-centric smartphone photography pipeline. Its goal is to produce individual pictures with good contrast and dynamic range, little noise and motion blur, and pleasing colors in most shooting scenarios, all while looking natural and requiring little to no user input. In order to achieve that goal, the authors of [Has+16] conceived a system with the following constraints:

- The images that are captured and processed are raw images, as they typically have higher bit depth and a better known noise model than their post-processed and compressed 8-bit JPEG counterparts.
- Images are underexposed at capture time to avoid clipped highlights and saturated pixels, which effectively allows the capture of more dynamic range. Thanks to a custom auto-exposure algorithm, a gain is also memorized to compensate later in the pipeline for that under-exposure.
- Images also have high shutter speed (or short exposure time) to avoid motion blur (from the scene or the camera).
- For a fixed gain, images that have a shorter exposure time tend to have a lower signal-to-noise ratio; to compensate for that, multiple images are actually captured (usually between 2 and 8 images depending on scene brightness measurements, also selected through the auto-exposure algorithm), and the temporal information of the burst is combined through an alignment and merging procedure. All images have identical shutter speed (meaning that unlike some HDR fusion methods, exposure is not bracketed) and the interval between each frame is constant. This allows both a similar noise profile

[3] https://tinyurl.com/dxomark-pixel2
[4] https://github.com/amonod/hdrplus-python
[5] https://ipolcore.ipol.im/demo/clientApp/demo.html?id=336

across the burst and easier, more accurate motion estimation and compensation.

- The resulting higher bit depth, high dynamic range, and less noisy image is then tone mapped to compensate for underexposition and to produce a more natural look, even in high contrast scenarios (in that case, shadows are boosted while as few highlights as possible are clipped).
- Additional processing is performed to produce a more visually pleasing final image with a distinctive "HDR+ look" (c.f. Section 1.5.1).
- Total burst processing time on a mobile device is only a few seconds, and the mode is transparent to the user (they do not know that several pictures are being taken or what processing needs to be performed).
- The whole process is automatic and parameter-free for the photographer.

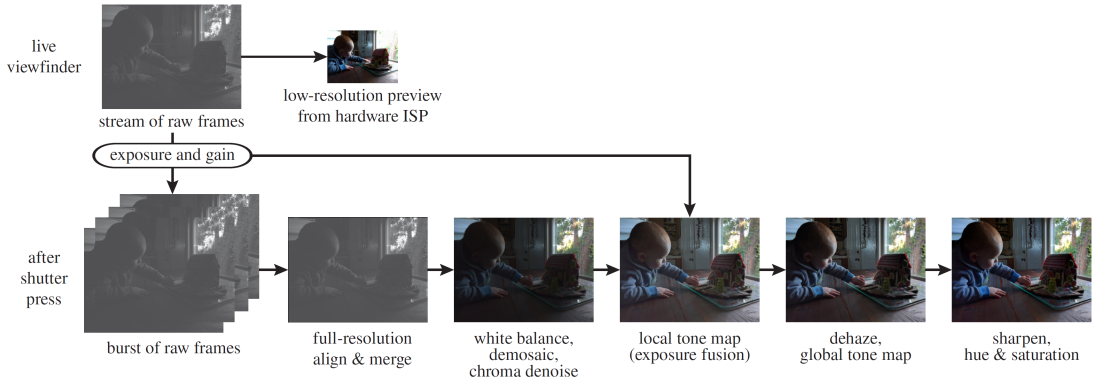


Figure 1.1: The HDR+ pipeline (extracted from [Has+16], lens shading and chromatic aberration correction have been omitted for brevity)

A representation of the whole image processing pipeline can be seen in Figure 1.1. For more information on the capture process, including the functioning of the example-based auto-exposure algorithm, please refer to the original article [Has+16], and its supplemental material[6]. In the next sections, we will explain the whole processing pipeline after capture.

1.3 Bayer Tile-based Alignment

Before combining the information of multiple noisy images to produce a single, clean image, one must make sure that all images feature similar content. Since they share the same exposure time, scene brightness can be considered constant for the duration of the burst. Even though that exposure time is short, there

[6] https://static.googleusercontent.com/media/hdrplusdata.org/en//hdrplus_supp.pdf

can be apparent motion in the burst, mainly due to camera shake (the pipeline being designed for smartphones, most pictures will be taken handheld) and objects moving in the scene. That motion has to be compensated.

The strategy to compensate motion across the burst is to cleverly select one image as a reference, and then estimate motion between that image and each one of the other burst images (sometimes referred to as alternate frames). To that end, the reference image is divided into a regular array of square tiles and, for each tile, the corresponding tile in the alternate image is the one of minimal distance within a defined search radius around the original location of the reference tile. Estimating the motion of image tiles instead of individual pixels is sufficient for the merging phase, and as we will explain in Section 1.4, the rest of the algorithm is robust to alignment errors.

1.3.1 Reference Image Selection

The reference image is selected among the first three images of the burst in order to minimize perceived shutter lag (the content of subsequent images of the burst can be quite different from what it was when the user pressed the shutter). Among these, the sharpest image is picked, the reasoning being that images that are less sharp are more likely to feature motion blur, and will not merge well with images that are sharp in the same regions. Sharpness is measured by the magnitude of gradients within a single (green) channel. We did not re-implement this part of the algorithm in our own implementation: for bursts provided by Google (which we'll discuss in Section 1.6.1), we simply reuse their selected reference image. In the case of new bursts, we let the user select the reference or default to the first image otherwise.

For the remainder of the alignment step, the full resolution Bayer images are converted to single channel, lower resolution grayscale images using a simple 2×2 box filter: each R, G, G, B square Bayer pattern is averaged in order to produce a single intensity value. Since the motion estimation of the grayscale images produces final results at pixel level, motion can only be estimated in multiples of 2 pixels in the original full-size raw images. Although this could be considered as a lack of precision, it is actually convenient because the arrangement of color planes in the Bayer image will be preserved after alignment (the value of a blue pixel will never be replaced by the value of a red pixel).

1.3.2 Multi-scale Pyramid Alignment

In order to quickly cover a large search area to find the tiles with minimal distance, the following multi-scale coarse-to-fine alignment strategy is adopted:

- Gaussian pyramids of the single channel burst images are constructed (e.g. in the HDR+ article supplement, it is typically 4-level, with successive downsampling factors of 2, 4 and 4 from the grayscale image to the coarsest level of the pyramid).

- At each pyramid level starting from the coarsest (i.e. lowest resolution) level, the reference image is divided in equally spaced square tiles (e.g. in the HDR+ supplement tiles of size 8×8 at the coarsest level and 16×16 at other levels). For each reference tile, we compute the L2 or L1 distance between it and each possible tile of the alternate frame within a specified search radius (e.g. + or - 4 pixels) around some initial guess (that guess being the location of the reference tile at the coarsest level)

$$D_p(u,v) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} \left| T(i,j) - I\left(i+u+u_0, j+v+v_0\right)\right|^p, \tag{1.1}$$

  where T is the reference tile of size $n \times n$, (u,v) is the possible location of the alternate image tile within the larger search area I currently evaluated, p is the power of the norm used (1 or 2) and (u_0, v_0) is the initial guess which indicates the location of the search area ($u_0 = v_0 = 0$ at the coarsest scale). As suggested in the HDR+ supplement, the L2 distance (written in the case where $u_0 = v_0 = 0$ for brevity) can be rewritten as

$$D_2(u,v) = \sum_{x=0}^{n-1}\sum_{y=0}^{n-1} T(i,j)^2 + \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} I(i+u, j+v)^2 - 2\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} T(i,j) I(i+u, j+v), \tag{1.2}$$

  where the second term can be computed by filtering the squared elements of I with a $n \times n$ box filter, and the third is proportional to the cross-correlation of I and T, which can be computed quickly using fast Fourier transforms.
  The displacement from the location of the reference tile to the location of the alternate tile with minimal distance gives us the motion vector attributed to the reference tile.

- Alignments are upsampled, meaning that motion vectors are scaled and propagated to the correct number of tiles in the finer level. For example, if tiles are 16×16 at both levels and the upsampling factor is 4, a tile with a (2, 1) motion vector at the coarser level implies 16 tiles with a (8, 4) motion vector at the finer level.

- An additional step is added in order to mitigate some potential upsampling problems, particularly when coarse-scale tiles straddle over the boundaries of moving objects. For each tile at a new (finer) level, instead of directly using the upsampled motion vector of the corresponding coarse-scale tile as the initial alignment guess (u_0, v_0), 3 candidates are evaluated: the alignment of the coarse-scale tile, plus those of the nearest coarse-scale tiles in each spatial dimension. The candidate alignment that minimizes the L1 distance between the reference tile and the corresponding alternate tile at the new pyramid level becomes the new initial alignment guess. This step is illustrated in Figure 1.2.

- The selected upsampled motion vectors are then used as the location of the search area for the motion estimation at the subsequent pyramid level.

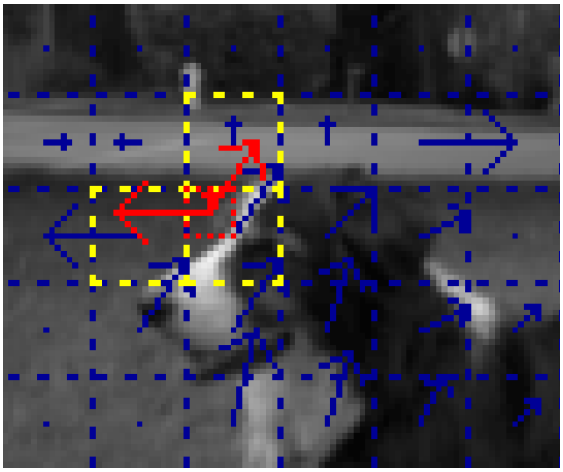
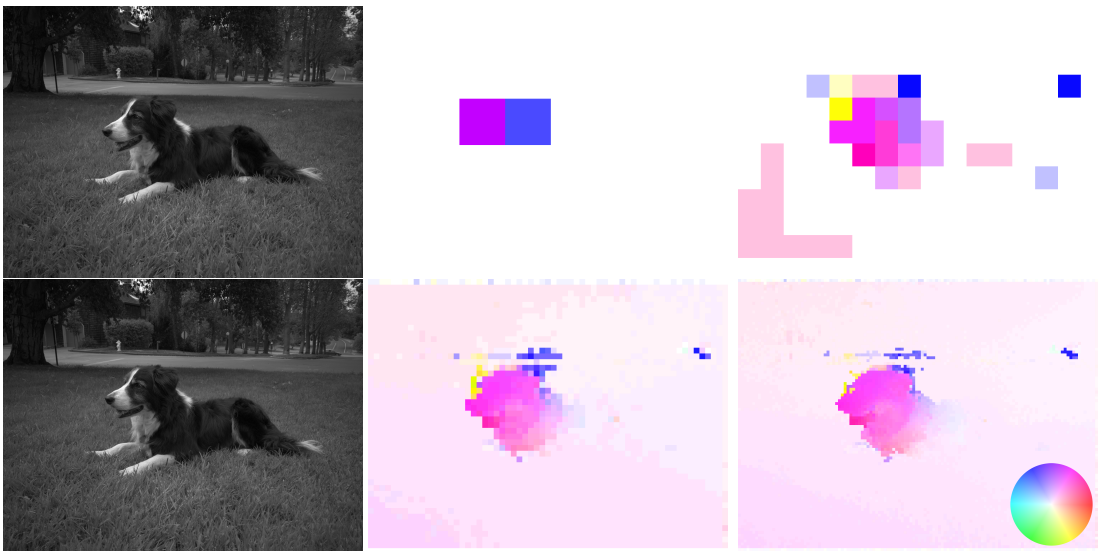


Figure 1.2: **Blue**: coarse-scale motion field (arrows: motion vectors, dotted lines: tiles). **Red**: finer-scale tile. **Yellow**: its 3 nearest coarse-scale tiles. The 3 motion vectors extracted from those neighboring tiles will be the candidates evaluated at the finer scale (shown in **red**).

For example, if (u_m, v_m) is the upsampled motion vector resulting from the previous steps, the initial guess (u_0, v_0) will be updated to (u_0+u_m, v_0+v_m).



(a) Image pair (ref=top) (b) Coarse-to-fine motion fields (hue: direction, saturation: magnitude)

Figure 1.3: Multi-scale tile-based alignment using Gaussian pyramids

An example of coarse-to-fine motion fields can be seen in Figure 1.3. Given that the alignments are computed for downsampled grayscale versions of the original images, motion vectors must be multiplied by a factor of 2 to go back to the raw Bayer array size.

1.3.3 Subpixel Alignment

We implemented subpixel alignment as defined in the article: between two pyramid scales, for each motion estimation result, we fit a bivariate quadratic polynomial to the 3×3 window surrounding the L2 distance minimum, and find the minimum of the polynomial.

As described in the article supplement[7], D_2 can be approximated as

$$D_2(u, v) \approx \frac{1}{2} [u; v]^\top \boldsymbol{A} [u; v] + \boldsymbol{b}^\top [u; v] + c, \tag{1.3}$$

where $\boldsymbol{A}$ is a 2×2 positive semi-definite matrix (because the shape of D_2 is assumed to be an upward-facing quadratic surface near the minimum), $\boldsymbol{b}$ is a 2×1 vector and c is a scalar.

If $(\hat{u}, \hat{v})$ is the estimated distance minimum, the technique consists in constructing a weighted least-squares problem fitting a quadratic polynomial to the 3×3 window of D_2 centered around $(\hat{u}, \hat{v})$, which we call D_2^{sub}. Without loss of generality, we can solve the problem assuming $(\hat{u}, \hat{v}) = (0, 0)$ and then shift the estimated subpixel position μ by $(\hat{u}, \hat{v})$.

The free parameters of the quadratic can be estimated by computing the inner product of D_2^{sub} with a set of six 3×3 filters, each corresponding to an unknown parameter in $(\boldsymbol{A}, \boldsymbol{b}, c)$

$$\begin{aligned} \boldsymbol{A} &= \begin{bmatrix} F_{A_{1,1}} \cdot D_2^{sub} & F_{A_{1,2}} \cdot D_2^{sub} \\ F_{A_{1,2}} \cdot D_2^{sub} & F_{A_{2,2}} \cdot D_2^{sub} \end{bmatrix}, \\ \boldsymbol{b} &= \begin{bmatrix} F_{b_1} \cdot D_2^{sub} \\ F_{b_2} \cdot D_2^{sub} \end{bmatrix}, \\ c &= F_c \cdot D_2^{sub}. \end{aligned} \tag{1.4}$$

In some cases, additional operations might be required so that $\boldsymbol{A}$ is guaranteed positive semi-definite. Please refer to the supplement for demonstrations, including the derivation of the filters.

Once the quadratic approximation parameters are estimated, we can compute the minimum of the fitted surface

$$\boldsymbol{\mu} = -\boldsymbol{A}^{-1}\boldsymbol{b}, \tag{1.5}$$

which, in the case of a bivariate polynomial is equivalent to

$$\mu = -\frac{\left[\boldsymbol{A}_{2,2}\boldsymbol{b}_1 - \boldsymbol{A}_{1,2}\boldsymbol{b}_2, \quad \boldsymbol{A}_{1,1}\boldsymbol{b}_2 - \boldsymbol{A}_{1,2}\boldsymbol{b}_1\right]^{\mathrm{T}}}{\boldsymbol{A}_{1,1}\boldsymbol{A}_{2,2} - \boldsymbol{A}_{1,2}^2}. \tag{1.6}$$

The location of the minimum of the quadratic μ yields a subpixel displacement vector (because we assumed $(\hat{u}, \hat{v}) = (0, 0)$ to solve the problem, when in reality

[7] https://static.googleusercontent.com/media/hdrplusdata.org/en//hdrplus_supp.pdf

$(\hat{u}, \hat{v})$ can be at any position). We simply need to add this displacement to the original location of the pixel level minimum. This procedure is quite vulnerable to image noise and flat regions, so we only take that displacement into account if it is less than a pixel away from the original distance minimum location: if $||\boldsymbol{\mu}|| \leq 1$, we update the estimated motion vector $(\hat{u}, \hat{v})$ to $(\hat{u}, \hat{v}) + \boldsymbol{\mu}$.

Even though this procedure is supposed to help with the upsampling of motion vectors from one level of the pyramid to the next, we have found very few visual improvements to the overall result in our experiments. Subpixel alignment is not applied at the end of the finest pyramid level: because we are dealing with downsampled Bayer to grayscale images, having subpixel motion vectors at the last scale could imply having pixels aligned to pixels from different Bayer CFA channels, which would induce significant color shifts (which we verified in our experiments).

The pseudo-code of the overall alignment strategy of the HDR+ pipeline is described in Algorithm 1.

Algorithm 1: HDR+ tile-based alignment

input : $\{I_0, \dots, I_{N-1}\}$ burst of N downsampled grayscale images
I_0=*reference image*

input : $\{p_0, \dots, p_3\} \in \{1, 2\}^4$ norm power at each pyramid level
Typically $\{2, 2, 2, 1\}$

output: Sets of aligned tiles at reference image location (i, j)
$\{T_0(i, j), \dots, T_{N-1}(i, j)\}$

foreach $k \in \{0, \dots, N-1\}$ **do**
  Compute G_k^l, $l \in \{0, \dots, 3\}$ *4-level coarse-to-fine Gaussian pyramid of* I_k, $G_k^3 = I_k$.
foreach $k \in \{1, \dots, N-1\}$ **do**
  $(u_{-1}, v_{-1})(i, j) \leftarrow (0, 0)\ \forall i, j$ *Initial guess at coarse scale: no offset.*
  foreach *pyramid level* $l \in \{0, \dots, 3\}$ **do**
    Divide G_0^l in equally spaced tiles T_0^l of size $n_l \times n_l$
    if $l > 0$ **then**
      Upsample the previous level alignments $(u_{l-1}, v_{l-1})(i, j)$, $\forall i, j$
    foreach *reference tile* T_0^l *at location* (i, j) **do**
      if $l > 0$ **then**
        Update the upsampled $(u_{l-1}, v_{l-1})(i, j)$ by keeping the alignment that minimizes $D_1(u_{l-1}, v_{l-1})$ among those of the 3 nearest coarse-scale tiles
      Get all possible (u, v) locations in G_k^l of tiles of size $n_l \times n_l$ in a search area of size $(n_l + 2r_l) \times (n_l + 2r_l)$ centered around $(i + u_{l-1}, j + v_{l-1})$ *r_l: search radius.*
      Compute all possible distances $D_{p_l}(u, v)$, $p_l \in \{1, 2\}$ *Equation (1.1).*
      $(u_l, v_l)(i, j) \leftarrow (u_{l-1}, v_{l-1})(i, j) + \text{argmin}_{u,v} D_{p_l}(u, v)$ *Add computed displacement to initial guess.*
      if $l < 3$ **then**
        Compute subpixel displacement vector $\boldsymbol{\mu}(u_l, v_l)$
        if $||\boldsymbol{\mu}|| < 1$ **then**
          $(u_l, v_l)(i, j) \leftarrow (u_l, v_l)(i, j) + \boldsymbol{\mu}$
  Associate to the reference tile at finest scale T_0^3 at location (i, j) the tile T_k^3 of I_k at location $(i + u_3, j + v_3)$

1.4 Fourier Tile-based Merging

For a given reference tile, we now have a set of corresponding tiles (one per alternate frame according to the results of the alignment step). If these tiles feature the same image content, i.e. if motion (or lack thereof) is correctly estimated, intensity

differences between them can only be attributed to noise, and we can merge these tiles to obtain a single, temporally denoised tile.
This is of course an ideal case, and in practice, tiles can also differ because of errors in the motion estimation. The method presented in Section 1.3 has several limitations that can lead to alignment errors:

- The alignment is tile-based, so pixels that belong to different objects can be part of the same tile and therefore be attributed the same motion.
- Motion is estimated locally as the most likely translation of a tile of fixed size in the image plane, according to a minimal L1 or L2 norm. This means that any object motion that is not parallel to the image plane is not guaranteed to be correctly approximated.
- Depending on the selected parameters (successive downsampling factors, tile sizes, search radii and norm types) versus the actual motion, motion estimation could be stuck in a local mininum. That said, since most real world scenes will feature motion that is compatible with the short exposure time of the burst, a good choice of parameters will mitigate that problem.
- The motion estimation is sensitive to noise: the higher the noise level with respect to the image content, the higher the likelihood of having an incorrect tile minimizing the distance.
- As with many motion estimation techniques, it does not bode well with occlusions: areas that are not occluded in the reference frame but are in the alternate frame will probably yield incorrect motion vectors.

Such errors are expected given the relatively simple nature of the alignment. Keep in mind that it is designed to run quickly on systems like smartphone systems on a chip (SoCs). Moreover, this fast, imperfect motion estimation can be sufficient for subsequent processing steps, provided the merging step takes those potential alignment errors into account. In this section, we will discuss the strategy employed by the authors of [Has+16] to perform temporal denoising while being robust to alignment errors.

1.4.1 Noise Level Estimation

For a set of tiles, the first prerequisite of the merging step is an estimation of the noise level. Since the noise profile is considered identical for all the images of the burst, we perform that estimation on the reference image.

In the HDR+ pipeline, raw image noise is estimated using the Poisson-Gaussian model described in [Foi+08], where it is composed of two mutually independent parts: a Poissonian signal-dependent component and a Gaussian signal-independent component. It is further simplified to a heteroscedastic Gaussian

model where the noise variance σ^2 can simply be expressed as an affine function of the signal level x:

$$\sigma^2(x) = \lambda_s x + \lambda_r \tag{1.7}$$

where the parameter λ_s can be tied to photonic / shot noise, and λ_r can be tied to read noise [Mil+18].

There are multiple ways to obtain these two noise curve parameters:

- One can take one or several pictures of a scene that features multiple uniform (typically gray) areas and perform mean and variance measurements of intensities [Foi+08; HK94]. (λ_s, λ_r) can then be computed from simple linear regression.

- In [Has+16], the values of (λ_s, λ_r) are claimed to be function of the analog and digital gain settings selected at capture time. Since the authors implemented their pipeline on a specific set of smartphones and sensors, they could likely use a look-up-table that gives them the appropriate per-camera (λ_s, λ_r) tuple as a function of the applied gain settings. In some raw DNG files, λ_s and λ_r can be directly found in the `NoiseProfile` DNG tag[8].

- The `NoiseProfile` tag is not present in all DNG files, and separate analog and digital gain values are not directly accessible in image metadata. A single value is usually specified: ISO, which combines analog and digital gain differently from one manufacturer to another. This makes it difficult to extract noise curve parameters *a posteriori*, especially given that different sensors can have different shot versus read noise curves [Mil+18]. That said, even though they might not correspond to the actual values of λ_s and λ_r of the image, we found that computing a (λ_s, λ_r) tuple from the image ISO and baseline values of λ_s and λ_r for an image at ISO=100 produces equally pleasing results for the rest of the merging step. We use the following formula:

$$\begin{aligned}
\sigma^2\left(\alpha_{\text{ISO}} x\right) &= \left(\alpha_{\text{ISO}}\right)^2 \sigma^2(x) \\
&= \left(\alpha_{\text{ISO}}\right)^2 \left(\lambda_{s_{\text{ISO}_{100}}} x + \lambda_{r_{\text{ISO}_{100}}}\right) \\
&= \alpha_{\text{ISO}} \lambda_{s_{\text{ISO}_{100}}} \alpha_{\text{ISO}} x + \left(\alpha_{\text{ISO}}\right)^2 \lambda_{r_{\text{ISO}_{100}}} = \lambda_s^{'} x^{'} + \lambda_r^{'},
\end{aligned} \tag{1.8}$$

where $\alpha_{\text{ISO}} = \frac{\text{ISO}}{\text{ISO}_{100}}$ is the ratio between the ISO setting used at capture time and ISO_{100} (considering that $\text{ISO}_{100} = 100$ is the baseline ISO where no gain is applied), $x^{'} = \alpha_{\text{ISO}} x$ is the actually observed image, and $(\lambda_{s_{\text{ISO}_{100}}}, \lambda_{r_{\text{ISO}_{100}}})$ are the baseline noise curve parameters at ISO_{100} (these could be known for a given camera, averaged from a certain amount of ISO-normalized images, or arbitrary).

[8] https://helpx.adobe.com/content/dam/help/en/photoshop/pdf/dng_spec_1_6_0_0.pdf

- Another solution is to use an off-the-shelf noise curve estimation algorithm directly on the input image. These algorithms usually revolve around finding homogeneous regions within the image and computing the variance and means of said regions [CB13; RA16]. Using a noise curve estimation algorithm would effectively allow the HDR+ burst denoising algorithm to run blind, as all necessary information is stored in the raw Bayer data. That said, extra care should be taken to verify the robustness of these algorithms and the consistency of their outputs. In our experiments, we found that sticking to image metadata or computing (λ_s, λ_r) from ISO and baseline values produced more consistent results, all while requiring less computational time and fewer external dependencies.

In order to avoid potential problems due to the undersampled nature of the red and blue channels of a Bayer pattern, the noise estimation and the remainder of the merging method are performed separately for each color plane (the two green channels are also treated independently). Since we computed pixel-level alignment for tiles of downsampled grayscale images, we can actually assign the same tile size and motion vectors to each individual color plane.

For computational efficiency, even though the noise is claimed to follow a signal-dependent model across the reference image, the noise variance is actually considered signal-independent within each reference image tile. This means that for a given tile T, a single intensity value ρ is used to evaluate the noise model. Instead of taking the average of all intensities within the tile, the authors of [Has+16] decided to consider the root-mean-square (RMS) of the tile

$$\rho(T) = \mathrm{RMS}(T) = \left(\frac{1}{n^2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} T(i,j)^2 \right)^{\frac{1}{2}}. \tag{1.9}$$

The reasoning being that given two tiles of the same average, one having a higher contrast than the other, the higher contrast tile will have a higher root-mean-square (and thus a higher estimated noise variance), which will allow a more aggressive temporal denoising.

1.4.2 Pairwise Wiener Temporal Denoising

Given a set of tiles (the reference tile and the alternate tiles selected by motion estimation), each alternate tile T_z is compared to the reference T_0. The difference between the two tiles $D = T_0 - T_z$ is computed, and that difference is compared to the estimated noise variance $\sigma^2(\rho(T_0))$. That comparison is then used to compute a weighted average of T_z and T_0 (if the difference between the two tiles is far greater than the estimated noise level, the alternate tile will have a very small weight and we fall back to the reference).

The final temporally denoised result is the average of all pairwise merges. This method is actually performed in the frequency space (i.e. via the 2D DFT of tiles), which allows each spatial frequency bin to be treated individually. This can be

useful in cases where alignment failure is partial (i.e. some frequencies match well while others do not) as not all the content of the tiles will be discarded.

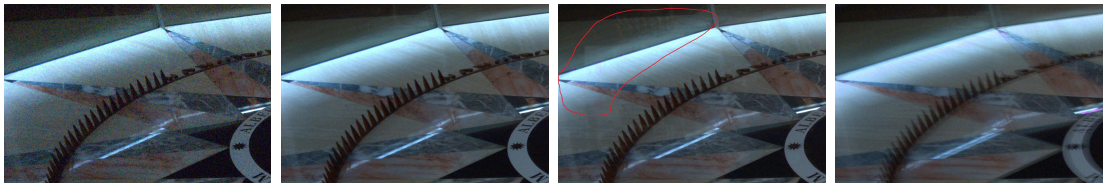
Mathematically,

$$\tilde{T}_0(\boldsymbol{\omega}) = \frac{1}{N}\sum_{z=0}^{N-1}(1 - A_z)T_z(\boldsymbol{\omega}) + A_z(\boldsymbol{\omega})T_0(\boldsymbol{\omega}), \tag{1.10}$$

where the shrinkage operator $A_z(\boldsymbol{\omega})$ is similar to a Wiener filter

$$A_z(\boldsymbol{\omega}) = \frac{|D_z(\boldsymbol{\omega})|^2}{|D_z(\boldsymbol{\omega})|^2 + c\sigma^2(\rho(T_0))}, D_z(\boldsymbol{\omega}) = T_0(\boldsymbol{\omega}) - T_z(\boldsymbol{\omega}), \tag{1.11}$$

where c is defined in the original article as a scaling and tuning factor[9]. For the sake of simplicity in this chapter, we will set $c = k\tau$ where k is the scaling factor and τ is the tuning factor that effectively controls the strength of temporal denoising. In [Has+16], k is fixed to $n^2 \times 1/4^2 \times 2$, where n is the tile length. However, the justification for this in the article is questionable, because they are trying to scale a single variance value computed in the spatial domain to a per-frequency-bin Wiener filter of the squared difference of two images. Still, we leave the value of k as is, since its influence can be superseded by simply changing the values of τ. The pseudo-code of temporal denoising is described in Algorithm 2. If $\tau \to 0$, $\tilde{T}_0(\boldsymbol{\omega}) \to T_0(\boldsymbol{\omega})$: we stick to the reference frame and the result is not temporally denoised. If $\tau \to +\infty$, $\tilde{T}_0(\boldsymbol{\omega}) \to \frac{1}{N}\sum_{z=0}^{N-1} T_z(\boldsymbol{\omega})$: the result is equivalent to an average of all aligned frames in the spatial domain, which can showcase alignment errors. These edges cases can be observed in Figure 1.4.



(a) Reference image crop ($\tau = 0$)

(b) Temporal denoise ($\tau = 75$)

(c) Temporal denoise ($\tau = +\infty$, ghosting)

(d) Burst average (no alignment)

Figure 1.4: Impact of the tuning factor τ on the result of temporal denoising.

[9]This 2D frequency-based denoising technique bears similarities with the burst deblurring method of Fourier Burst Accumulation [DS15; AM17]. However, the comparison of 2D frequency bins in HDR+ serves a different purpose: if T_0 is sharp, and it is still matched to a blurry tile T_z after alignment, the frequency content corresponding to the blurry elements ot T_z will be discarded, but only because it does not match that of T_0. On the contrary, if we have a blurry T_0, but a sharp T_z, there is also a strong frequency mismatch and we will not merge that content either. In effect, the Wiener filter does not introduce additional blur from other images, but it does not remove blur either.

Algorithm 2: HDR+ per-channel, tile-based, pairwise temporal denoising

input : Set of aligned tiles $\{T_0, \dots, T_{N-1}\}$ of size $n \times n$ at reference image location (i, j) (all tiles correspond to a single channel of the Bayer image)
input : Noise curve parameters λ_s, λ_r, temporal denoising tuning factor c
output: Single denoised tile $\tilde{T}_0$ at reference image location (i, j)

$\rho \leftarrow \mathrm{RMS}(T_0) = \left(\frac{1}{n^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} T_0(i,j)^2\right)^{\frac{1}{2}}$ *Compute the root-mean-square of* T_0.
$\sigma^2 \leftarrow \lambda_s \rho + \lambda_r$ *Compute the noise variance assigned to* T_0.
foreach $z \in \{0, \dots, N-1\}$ **do**
  $T_z(\boldsymbol{\omega}) \leftarrow \mathrm{FFT}(T_z)(\boldsymbol{\omega})$ *DFT of* $T_z(x, y)$.
$\tilde{T}_0(\boldsymbol{\omega}) \leftarrow T_0(\boldsymbol{\omega})\ \forall \boldsymbol{\omega}$
foreach $z \in \{1, \dots, N-1\}$ **do**
  foreach *frequency bin* $\boldsymbol{\omega} \in \{0, \dots, n-1\}^2$ **do**
    $D_z(\boldsymbol{\omega}) \leftarrow T_0(\omega) - T_z(\boldsymbol{\omega})$
    $A_z(\boldsymbol{\omega}) \leftarrow \frac{|D_z(\boldsymbol{\omega})|^2}{|D_z(\boldsymbol{\omega})|^2 + c\sigma^2}$
    $\tilde{T}_0(\boldsymbol{\omega}) \leftarrow \tilde{T}_0(\boldsymbol{\omega}) + (1 - A_z) T_z(\boldsymbol{\omega}) + A_z(\boldsymbol{\omega}) T_0(\boldsymbol{\omega})$ *Sum pairwise temporal denoisings.*
$\tilde{T}_0(\boldsymbol{\omega}) \leftarrow \frac{1}{N}\tilde{T}_0(\boldsymbol{\omega})$ *Average pairwise temporal denoising.*
$\tilde{T}_0(i, j) \leftarrow \mathrm{FFT}^{-1}(\tilde{T}_0)(i, j)$ *Inverse DFT of* $\tilde{T}_0(\boldsymbol{\omega})$.

1.4.3 Wiener Spatial Denoising

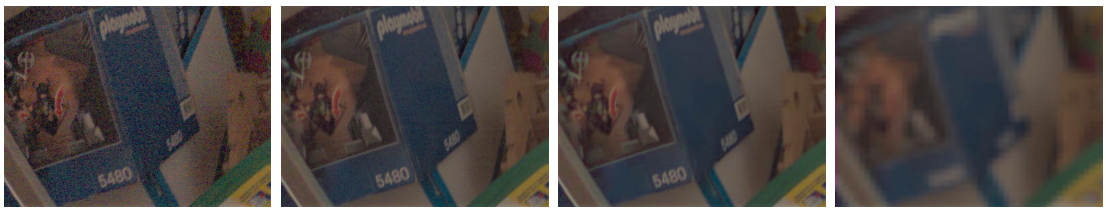
In addition to temporal filtering, spatial filtering is performed on the 2D DFT of tiles in order to remove some of the residual noise. Since we expect the noise level to be significantly reduced after temporal denoising, the estiated noise variance is updated to $\sigma^2(\rho(T_0))/N$ (this implies a perfect averaging of all N frames, which is not the case in practice but allows for a more conservative spatial denoising). We compute a shrinkage operator similar to the one used for temporal denoising and apply it to the spatial frequency coefficients

$$\hat{T}_0(\boldsymbol{\omega}) = \frac{\left|\tilde{T}_0(\boldsymbol{\omega})\right|^2}{\left|\tilde{T}_0(\boldsymbol{\omega})\right|^2 + f(\boldsymbol{\omega})\frac{\sigma^2(\rho(T_0))}{N}} \tilde{T}_0(\boldsymbol{\omega}), \tag{1.12}$$

where $f(\boldsymbol{\omega})$ is a noise shaping function; in [Has+16], it is a piecewise linear function that increases the "effective noise level" for higher spatial frequencies, tuned to subjectively maximize image quality. As we do not know the specifics of said shaping function, we simply replaced it with $f(\boldsymbol{\omega}) = \gamma\,|\boldsymbol{\omega}|$, which also effectively increases the noise level for higher frequencies, and features a scaling and tuning

factor γ. We set $\gamma = \frac{k}{2}s$, where k is the scaling factor defined in Section 1.4.2 and s controls spatial denoising strength.

Mathematically, if $s = 0$ we do not perform any spatial denoising. The higher the s, the more aggressively we filter higher spatial frequencies and obtain a lower-frequency final image (with little to no edges and textures within a tile in extreme cases). We typically set it to 0.1 to remove a bit of high frequency residual noise without loosing too much detail. The influence of s on the spatial denoising can be observed in Figure 1.5.



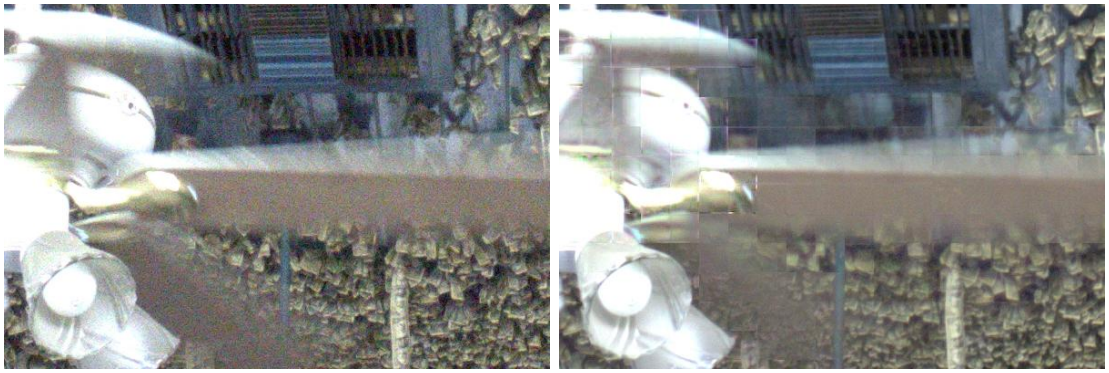
(a) Reference image (b) Temporal denoise ($s = 0$) (c) Temporal + spatial denoise ($s = 10$) (d) Temporal + spatial denoise ($s = 10^{12}$)

Figure 1.5: Impact of the tuning factor s on the result of spatial denoising

This frequency-based denoising technique (and particularly our simplified implementation) is not expected to be as efficient as state-of-the-art spatial denoising algorithms [Dab+09; LBM13; ZZZ18]. However, this step is used as a very computationally light complement to temporal denoising (since we are already in frequency space), which can be tuned to be fairly gentle and not remove image detail.

1.4.4 Overlapped Tiles and Raised Cosine Window

The steps described so far are the following: we start by dividing the reference image in equally spaced tiles; we then associate each reference tile to a specific stack of alternate tiles (one for each other image of the burst according to the result of the alignment step); after that, each stack is merged using the strategy described in the previous steps of Section 1.4. Once all stacks are merged, we obtain a set of denoised tiles of the same number and disposition we had after dividing the reference frame, which we can call tiles of the *merged* image. Unfortunately, these steps are not sufficient to create both a less noisy image and a realistic looking one. Because we merge each stack independently of others, and because different stacks have different image content, a different noise variance estimation, and can feature different alignment errors, continuity between adjacent tiles is not guaranteed. Additional artifacts can be seen on tile edges, because the merging operation is performed in the DFT domain (we compute the FFT of a tile stack, merge it, and then compute the inverse FFT to obtain a merged tile). These issues can be seen in Figure 1.6.



(a) Noisy reference image crop

(b) Merged image crop. Discontinuities between tiles and artifacts on tile edges due to DFT can be observed

Figure 1.6: Results obtained with the steps described so far

In order to create smoother images with less visually noticeable artifacts while retaining image detail, the solution of the authors of [Has+16] is twofold:

- **Tiles** (for both the alignment and merging steps) are **overlapped by half in each spatial dimension**. This means that the total number of tiles (and thus the number of operations for alignment and merging) is actually multiplied by a factor of 4.
- The window used for blending is a **modified raised cosine window**, defined in 1D as

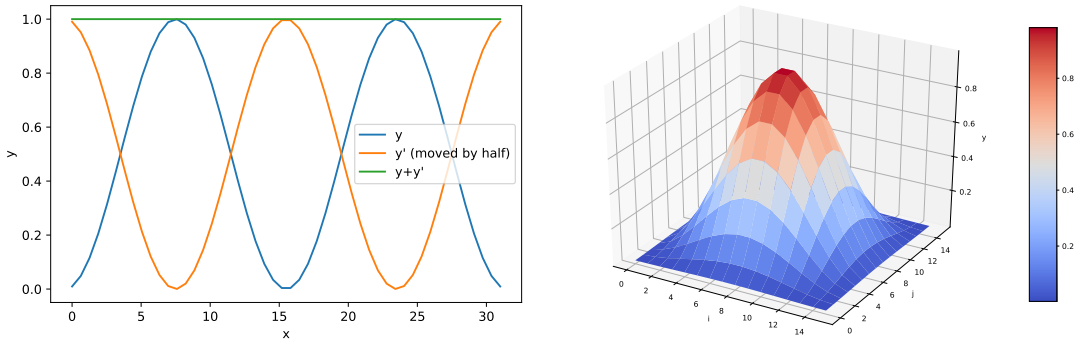
$$w_1(x) = \frac{1}{2} - \frac{1}{2}\cos\left(2\pi\left(x + \frac{1}{2}\right)/n\right), \quad 0 \leq x \leq n-1. \tag{1.13}$$

  If x is a vector of positions from 0 to $n-1$, this window function is centered, and has nonzero values at 0 and $n-1$ which means that content on edges is not fully discarded. As can be seen in Figure 1.7a, another interesting property is that when they overlap by half, the sum of two such windows is always equal to 1. When dealing with tiles that overlap by half in two spatial dimensions, the actual window used for blending (visible in Figure 1.7b) is the product of the 1D version in each dimension

$$w_2(i,j) = w_1(i)w_1(j), \quad 0 \leq i,j \leq n-1. \tag{1.14}$$

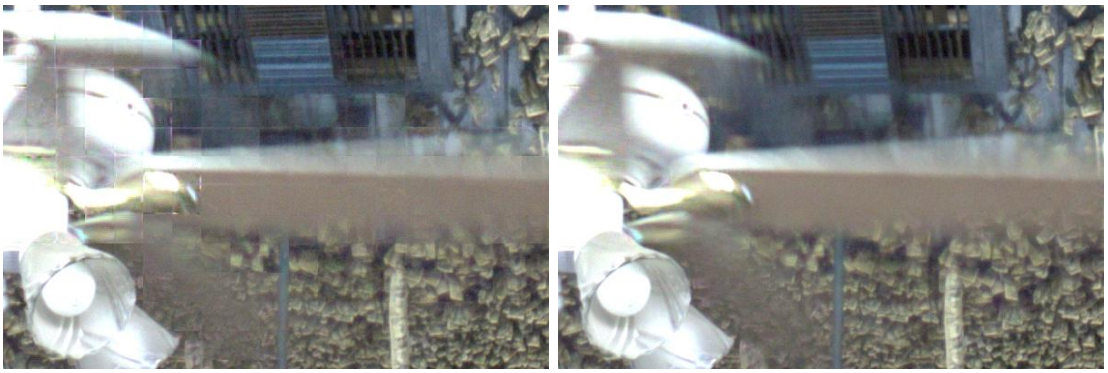
  This window allows the smooth blending of the overlapped tiles and has the added benefit of removing DFT artifacts on tile edges.

Going back to our real burst denoising example, Figure 1.8 showcases that using the aforementioned blending technique does create a more pleasing image without loosing image detail.



(a) 1D window ($n = 16$) (b) 2D window

Figure 1.7: Modified raised cosine window.



(a) Merged image crop without overlapped tiles (b) Merged image crop with overlapped tiles + cosine window

Figure 1.8: Using overlapped tiles and the modified raised cosine window prevents observable discontinuities and edge artifacts while retaining image detail.

1.5 Finishing

After the alignment and merging steps, we obtain a single raw image based on the reference but with significantly less noise. The system described in [Has+16] being a full camera pipeline, the subsequent steps that lead to a final `.jpg` image are also presented. Even though this chapter mainly focuses on the temporal denoising part of the HDR+ solution, this section discusses the remainder of the image processing pipeline.

1.5.1 Google's Pipeline

The system described in [Has+16] being a full camera pipeline, the subsequent steps that lead to the final image are also presented:

- Black level subtraction: saved values of additional sensor pixels that are

shielded from the light are subtracted so that image pixels that do not receive any light have a 0 value.

- Lens shading correction: areas on the image corners that are darker are brightened. Low-resolution lens shading maps that are provided by the ISP are used for that compensation.

- White balance: colors are shifted so that objects human observers would perceive as gray actually appear as gray in the final image. Gains are applied to each one of the 4 RGGB channels individually, and those gains are provided by the ISP.

- Demosaicking: the raw Bayer image where each pixel corresponds to a single color channel is converted to an RGB image of the same size where each pixel has 3 color channels. The raw image being undersampled, an interpolation algorithm is used (the algorithm in the HDR+ pipeline is said to use a combination of techniques featured in [Gun+05] and is most likely proprietary).

- Chroma denoising: even after the temporal and spatial denoising steps, the image can still contain artifacts such as red and green blocks in dark areas, especially in the case of low-light images. To that end, the HDR+ authors apply a sparse 3×3 tap non-linear kernel in two passes to the image in the YUV space, acting as an approximate bilateral filter.

- Color correction: a 3×3 matrix supplied by the ISP is used to convert the image from sensor RGB to linear sRGB values.

- Dynamic range compression / local tone mapping: aside from burst denoising, the HDR+ pipeline aims at improving the dynamic range of the final image in specific scenarios (hence its name): in scenes that feature a high dynamic range, darker areas of the image must be brightened, while brighter content must remain unsaturated and local contrast must be preserved. To that end, a local tone mapping method derived from the exposure fusion algorithm [MKV09; Hes18] is employed. However, since this method requires several images of different brightness (usually several captures at different exposure times), two gamma-corrected "synthetic exposures" are created from the intermediate result: one short (a gamma-corrected grayscale version of the current image, which features the exposure time used during the burst capture) and one long exposure (a gamma-corrected grayscale version of the current image with a gain applied according to the auto-exposure algorithm specific to that pipeline). From these two images, the exposure fusion algorithm uses image pyramids to create a smoothly blended image where each pixel is best-exposed. The gamma correction is then inverted and the image re-colorized by keeping the chroma ratios of the intermediate image.

- Dehazing: an extra global tone curve that pushes low pixel values even lower while preserving highlights and midtones is applied in order to mitigate the effect of veiling glare.
- Global tone adjustment / gamma curve: an S-shaped contrast enhancing tone curve is concatenated to the sRGB color component transfer function to transform the linear sRGB image to a nonlinear sRGB image where contrast is increased.
- Chromatic aberration correction: longitudinal and lateral chromatic aberration are corrected by replacing chroma values on high contrast edges by those of nearby pixels that are less likely to be affected by chromatic aberration.
- Sharpening: the authors implement unsharp masking[10] through a sum of Gaussian kernels constructed from a 3-level convolution pyramid.
- Hue-specific color adjustments: some custom transformations are applied to the image colors (shifting bluish cyans and purples towards light blue, and increasing the saturation of blues and greens generally), which is said to make vegetation and blue skies more visually appealing.
- Dithering: to mitigate quantization artifacts when reducing from 10/12 to 8 bits per pixel (sRGB images are typically destined for viewing on 8-bit displays), dithering is implemented by adding blue noise from a precomputed table.
- JPEG quantization and compression: the image is quantized and encoded to 8 bits where the lossy compression is set to a quality level of 95, resulting in the final `.jpg` file.

All these steps are performed via software (i.e. in a smartphone, these operations can run on the CPU/GPU and not necessarily on the ISP).

1.5.2 Our Simplified Pipeline

Since we decided to mainly focus on the denoising part of the HDR+ algorithm for this chapter, we implemented a simpler finishing pipeline. It still features parts of Google's own solution, and it is presented as a proof of concept, to showcase some of the decisions they made when designing their pipeline. It does not aim for parity with Google in terms of visual quality, as it would require a considerable amount of additional development time and a lot of trial and error, especially given the relatively sparse description of these steps in the original article.

Using rawpy, a Python wrapper of the LibRaw library[11] we transform a temporally and spatially denoised 16 bit Bayer array (even though smartphone raw

[10] https://micro.magnet.fsu.edu/primer/java/digitalimaging/processing/unsharpmask/index.html

[11] https://pypi.org/project/rawpy/

images typically have a 10 or 12 bit precision, they are often stored on 16 bit arrays) into a 16 bit linear RGB image of the same resolution with the following operations:

- **Black level subtraction**
- **White balance**: rawpy uses the gains stored in the `As Shot Neutral` Exif metadata tag in the `.dng` file of the reference image to apply scales to the red and blue Bayer channels.
- **Demosaicking**: the raw Bayer image where each pixel corresponds to a single color channel is converted to a RGB image of the same size where each pixel has 3 color channels. We use the AHD algorithm [HP05] since it was both used for burst fusion comparisons in the HDR+ supplement (where it was deemed "representative of the algorithms used by mobile ISPs") and available in rawpy. That algorithm is not the proprietary one used in Google's implementation.
- **Color correction**: rawpy uses image metadata such as the `Color Matrix` Exif tag in the `.dng` file of the reference image to convert the image from sensor linear RGB to standard linear sRGB color space.

Once we are done with rawpy postprocessing, we apply additional custom operations:

- **Hdr tone mapping**: we use a technique similar to the one employed in [Has+16], where two synthetic exposures are created from a single image and combined with exposure fusion [MKV09] (a similar strategy is extended and demonstrated in [Hes19]). From the result of all previous steps (which stems from an underexposed burst), we get a grayscale image by averaging the 3 color channels and then synthesize two exposures: we apply the standard sRGB gamma correction to get a short exposure, and we apply a gain followed by the same gamma correction to synthesize the long exposure. We then perform exposure fusion using the OpenCV `mergeMertens` implementation[12]. Given that we are blending grayscale images, and that in their implementation, the authors of [Has+16] use "a fixed weighting function of luma that favors moderately bright pixels", we found that only taking the "well-exposedness" weights of exposure fusion into account for blending (we set the `exposure_weight` parameter to 1 and `contrast_weight` and `saturation_weight` to 0 when calling the `createMergeMertens` function) produced convincing results (Figure 1.9 shows an example of the applied exposure fusion). We undo the gamma correction of the fused grayscale image, and compute a per-pixel scaling with the element-wise division of the result by the original grayscale image. We then apply that scaling to each channel of the underexposed RGB image to obtain the tone-mapped result.

[12] https://docs.opencv.org/master/d7/dd6/classcv_1_1MergeMertens.html

(a) Synthetic short exposure (b) Long exposure (gain=15) (c) Exposure fusion result

Figure 1.9: Dynamic range compression / local tone mapping.

- **Contrast enhancement / global tone mapping**: our next step is to increase image contrast by applying the following simple S-shaped function to each color channel

$$y = \max(0, \min(x - \alpha \sin(2\pi x), 1)). \tag{1.15}$$

This makes dark areas of the image darker and bright areas brighter, thus

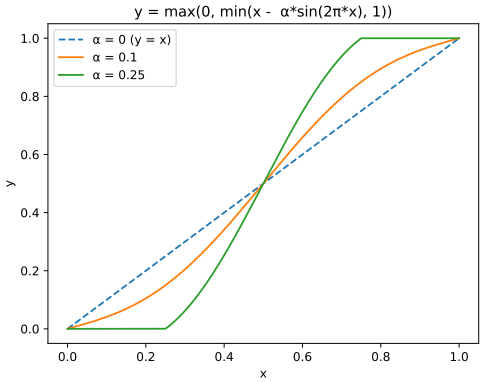


Figure 1.10: S-shaped contrast enhancement curve.

enhancing contrast. As illustrated in Figure 1.10, the α parameter must be carefully selected to avoid completely black or white regions while retaining a visually pleasing result. Although it is probably different from the one used in Google's own HDR+ implementation, this S-shaped contrast enhancing curve is actually very similar to the one referenced and used in another Google publication, namely "Unprocessing images for learned raw denoising" [Bro+19] which we'll discuss in greater detail in Chapter 2 of this manuscript.

- **Gamma compression**: pixel values are shifted from linear to non-linear sRGB space in order to both reduce the required bandwidth and produce a more pleasing and identifiable look, taking advantage of the human visual

sensitivity which is also non-linear (for example, humans are more sensitive to brightness differences in darker tones versus brightness differences in brighter tones). The sRGB transfer function is defined by the IEC 61966-2-1 international standard[13] as

$$V_{out} = \begin{cases} 12.94V_{in}, & 0 \leq V_{in} \leq 0.0031308, \\ 1.055V_{in}^{\frac{1}{2.4}} - 0.55, & 0.0031308 < V_{in} \leq 1. \end{cases} \tag{1.16}$$

- **Sharpening**: we get a final sharper image by computing the average of 3 images obtained via unsharp masking

$$I_{sharp} = \frac{1}{3}\sum_{m=1}^{3} I_{sharp_m}, \tag{1.17}$$

where

$$I_{sharp_m}(x,y) = \begin{cases} I(x,y) + \alpha_m(I(x,y) - I_{blur}^{\sigma_m}(x,y)) & \text{If } \left|I(x,y) - I_{blur}^{\sigma_m}(x,y)\right| > \tau_m, \\ I(x,y) & \text{otherwise,} \end{cases} \tag{1.18}$$

$I_{blur}^{\sigma_m}$ being the result of the convolution of I with a Gaussian kernel of standard deviation σ_m, α_m controls the sharpening strength and τ_m is a threshold that controls which pixels will actually be sharpened. We empirically found that setting $\alpha_m \in \{1, 0.5, 0.5\}$, $\sigma_m \in \{1, 2, 4\}$ and $\tau_m \in \{2\%, 4\%, 6\%\}$, $m \in \{1, 2, 3\}$ produced visually pleasing results (image that looks a bit sharper while remaining natural-looking, little to no noise reintroduced after sharpening homogeneous regions). This mask is certainly different from the one employed in [Has+16] given that its parameters are unknown.

- **JPEG quantization and compression**: the image is quantized and encoded to 8 bits, resulting in the final `.jpg` file. We set the quality level of the lossy compression to 100 to minimize compression artifacts.

Although these steps produce less visually pleasing images than the full HDR+ finishing pipeline, they still showcase one of its most defining aspects: HDR tone mapping from a single underexposed merged image. A full or partial combination of these steps can be used for any raw .dng image, including the noisy reference image, or intermediate results of Google's own implementation. This will come in handy for visual comparisons, which we'll discuss in Sections 1.6.2 and 1.6.3.

1.6 Comparison to the Original HDR+ Implementation

[13] http://www.color.org/chardata/rgb/srgb.xalter

1.6.1 The HDR+ Dataset

In 2018, [Has+16] was completed by a very large dataset captured with several smartphones, simply named the HDR+ dataset[14]. It is comprised of 3640 raw bursts (28461 images total), for a total of 765 GB. Each burst contains the following files:

- a set of raw images stored in the open and commonly used Adobe Digital Negative (`.dng`) format, which can be read by rawpy and contains useful metadata such as exposure time, ISO, black level as well as white balance gains estimated by the camera.
- additional metadata stored in separate files, such as maps for lens shading correction and color correction matrices.

A curated subset of 153 bursts is also available. Since the HDR+ algorithm is not learning-based, we mostly focused on that subset for the purposes of this chapter.

Two sets of Google's own results are available on the dataset website: the 2016 results, said to correspond to the algorithm and parameters described in [Has+16], while the 2017 ones correspond to a more recent version of the pipeline with "algorithm refinements and updated tuning". For each raw burst, a set of results comes with the following data:

- A raw `merged.dng` file which is said to correspond to the result of the alignment and temporal denoising steps (the dataset authors claim no additional processing of the raw image).
- A `final.jpg` file, which is obtained after the full finishing pipeline described in Section 1.5.1 is applied to the merged result.
- A `reference.txt` file that contains the index of the picked reference frame.

1.6.2 Comparison to Google's Merged Results

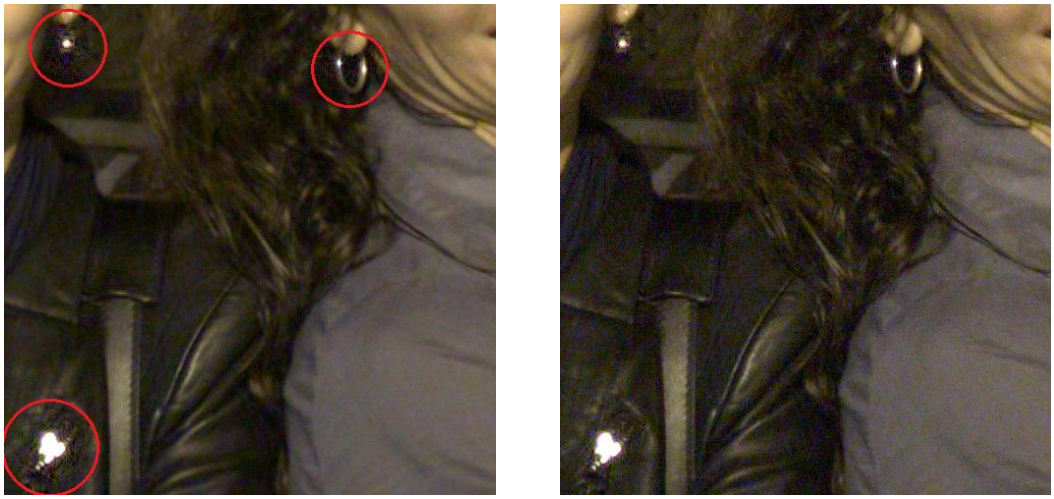
The inclusion of Google's raw alignment and temporal denoising results along with the chosen reference frame allows for an apples-to-apples comparison with our open-source Python implementation of the burst alignment and merging algorithm: we can measure similarity between the raw images and/or put the alignment and merging results through the same finishing pipeline for visual comparison.

Statistical Comparison of Raw Merged Images

We first wanted to compare our raw merged results to the 2016 Google ones, because they are said to be the closest to the implementation presented in the article. However, by putting these files through a simplified finishing pipeline (black level

[14] HDR+ Burst Photography Dataset, http://www.hdrplusdata.org/

subtraction, white balance, demosaicking, color correction, gamma curve), we actually found that it is very likely that the 2016 version of the `merged.dng` files also features spatial denoising, while the 2017 might not. Significant differences can be observed in Figure 1.11.



(a) 2016 "Merged" image (post-processed) (b) 2017 "Merged" image (post-processed)

Figure 1.11: The Google 2016 results seem to feature less residual noise except around strong high contrast features / specular highlights, which is typical of spatial denoising as stated in [Has+16].

On the subject of spatial denoising, the article provides no indication of the slopes of the "piecewise linear function" of spatial frequency (or of any potential tuning of the spatial denoising strength). Therefore, it would be ill-advised to compare the 2016 results to our own. We instead decided to perform comparisons against the 2017 results. Even though they are said to feature different tunings and algorithm refinements (we do not know all the 2016 tuning factors either), the alignment and merging results seem more in line with what we were able to obtain with our own implementation.

Using the subset of 153 bursts provided in the HDR+ dataset, we thus took the bursts where the raw reference image and the 2017 Google "`merged.dng`" result were of the same size (for an unknown reason, some Google images are larger than the reference), and set out to compute two PSNRs:

- the PSNR between Google's result and the noisier reference image it is based on.
- the PSNR between Google's result and our own result, tuned to match Google's result as much as possible:
  - Coarse-to-fine upsampling factors of the 4-level Gaussian pyramid: 4, 4, 2 (as suggested in the HDR+ supplement).

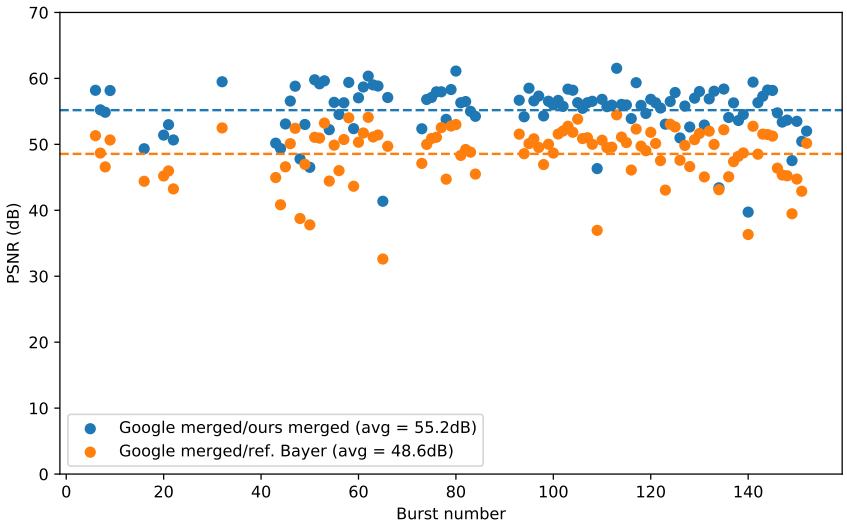


Figure 1.12: PSNR between Google's and our merged result on the applicable bursts of the "`20171106_subset`".

- Coarse-to-fine tile sizes: 8, 16, 16, 16 (as suggested in the HDR+ supplement).
- Type of norm used for tile distance computation: L2 at all levels except the finest, L1 for the finest (as suggested in the HDR+ supplement).
- Temporal denoising factor τ: 75 (it is claimed to be set to 8 in [Has+16], but we were not able to reproduce similar temporal denoising performance with our own way of computing the noise parameters (λ_s, λ_r) and thus σ^2).

On that topic, Google decided to increase the bit depth (and in turn change the black and white levels) of their raw merged result to put emphasis on the accuracy gained after alignment and temporal denoising, so we had to normalize the images before computing those PSNRs (our merged result has the same black and white levels as the reference image). With these settings, we achieve an approximate +7dB PSNR improvement when compared to the reference on average, our results having higher PSNR for every burst. This suggests that our implementation does perform alignment and temporal denoising somewhat similarly to Google's implementation. Bear in mind that we are trying to be close to images whose tuning and algorithm refinements are unknown. These similarity measurements can be seen in Figure 1.12.

Visual Comparison of Minimally Processed Merged Images

If we put the three raw images (the reference image, our merged result, and Google's) through the same minimal finishing pipeline (identical demosaicking +

white balance + color matrix + gamma curve), we can visually assess the similarity between Google's implementation of the alignment and temporal denoising steps and our own.

Looking at the subset of 153 bursts, some images seem to have less residual noise on our end, while others look more noisy than Google's result. The most likely candidates for noticeable differences between the two are:

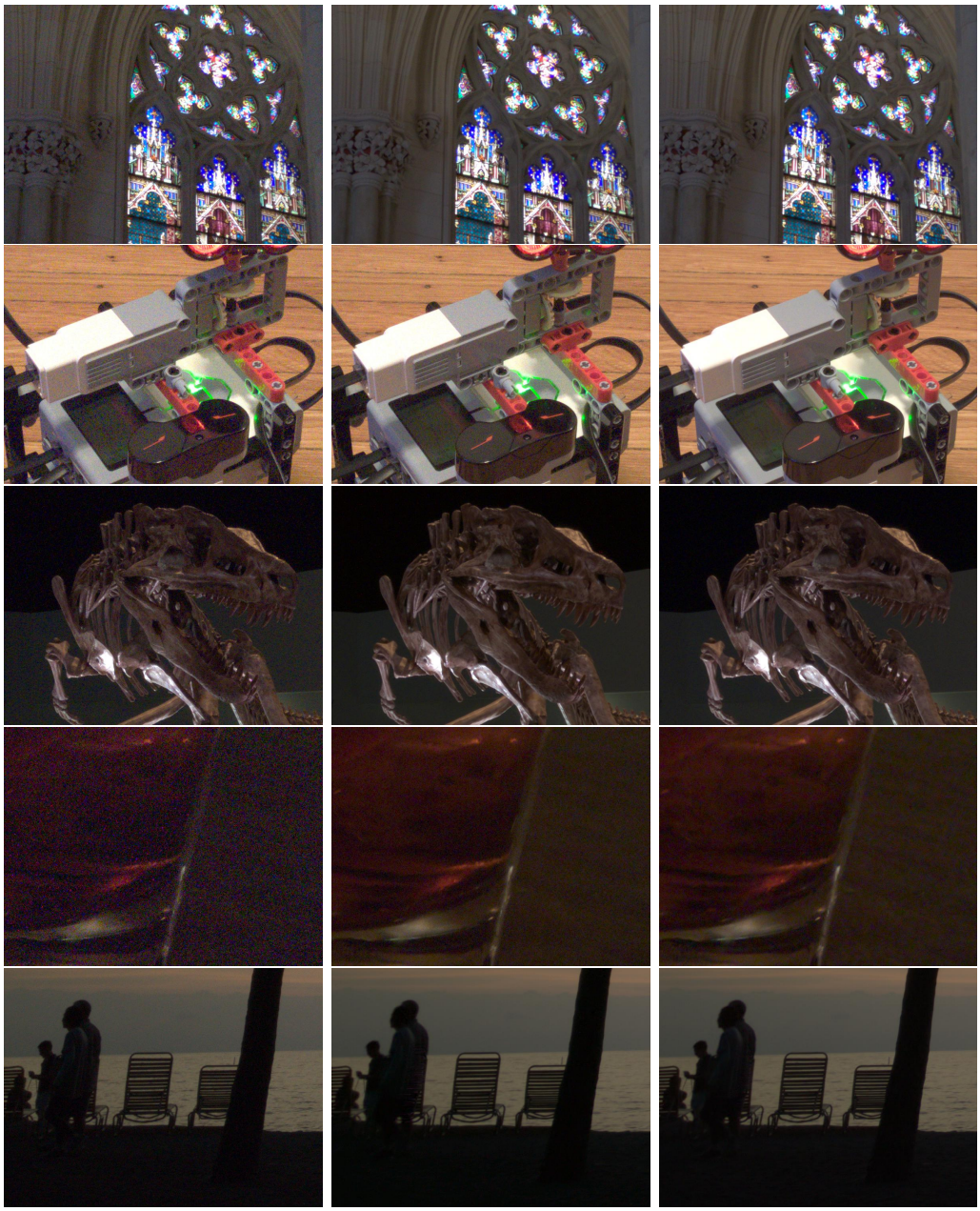
- The way we compute the (λ_s, λ_r) noise curve parameters: we either extract them from `.dng` metadata when possible, or compute them as a function of baseline ISO 100 values and actual ISO (as explained in Equation (1.8) of Section 1.4.1), while Google can derive these parameters from the analog and digital gain set at capture time.
- In [Has+16], the authors claim to typically use 16×16 tiles for alignment (at the finest pyramid level) and merging, except for very dark scenes where they use 32×32 tiles. We do not make that distinction and always use 16×16 tiles.
- Google could use different tuning factors from ours (this interacts with the two previous potential differences anyways).
- The so called "algorithm refinements" of the 2017 results we are unaware of.

Selected crops can be observed in Figure 1.13. Full-size results for the whole subset of 153 bursts can be consulted and downloaded from a Google Photos album[15].

1.6.3 Visual Comparison to Google's Final Results

The presence of their final results also enables visual comparison of their consumer-grade raw to JPEG finishing pipeline against our own. That said, given the simplified nature of our own finishing step, if we compare our final images to Google's cleverly chroma denoised, tone-mapped, sharpened, dehazed, color corrected (and so on and so forth) final HDR+ results, such comparison is definitely not in our favor in the vast majority of cases. It would certainly make sense to fully implement these additional finishing steps, but that would require a lot of additional development, testing and fine-tuning time to reach a similar level of quality. The added complexity would also likely imply longer processing times in our Python implementation, and would stir this chapter away from our main subject of focus, the raw burst alignment and merging algorithm. Still, if we compare our images to a simple post-processing of the reference image (demosaicking + white balance + color matrix + gamma curve), we can see that our pipeline does provide significant visual improvement. Some examples are featured in Figure 1.14. Final full-size results for the whole subset of 153 bursts can be consulted and downloaded from another Google Photos album[16].

[15] https://photos.app.goo.gl/mEhyNrqKc2x7rbqj9
[16] https://photos.app.goo.gl/QurTFcvUMc8i1DM7A



Reference **Ours merged** **Google merged**

Figure 1.13: Visual comparison of crops of minimally processed versions of the reference image, our "merged" result and Google's "merged" result (viewers are invited to zoom in). Both alignment and merging procedures significantly reduce noise. *Top row*: both merged results are virtually indistinguishable. *Second row*: Google seems to further reduce noise. *Third row*: our result seems more denoised. *Fourth row*: Google's residual noise seems to be of lower spatial frequency compared to ours (they might be using a larger tile size for this low-light image). *Bottom row*: for this image, ghosting is slightly stronger in our result.

Reference **Ours final** **Google final**

Figure 1.14: Visual comparison of crops of the minimally processed reference image, our final result and Google's final result (viewers are invited to zoom in). Google's different spatial denoising, tone mapping, sharpening and additional operations (chroma denoising, lens shading correction, dehazing, color adjustment, dithering, etc) ensure better results in almost any scenario. *First Row*: our results can be similar to Google's in some static, well lit environments. *Second row*: Google's more aggressive tone mapping produces an image with better dynamic range at the expense of more residual noise. *Third row*: since we do not apply any chroma denoising, residual noise can be amplified by our tone mapping and sharpening operations. *Bottom row*: Google also seems to perform hot pixel removal, which we do not (look at the top right corner of the crops).

1.7 Results on GoPro Bursts

In order to further verify the appeal of the HDR+ raw burst denoising algorithm and the improvements provided by parts of its finishing pipeline, we set out to test our own implementation on bursts that were not part of the HDR+ dataset. We used GoPro HERO 8 Black and Hero 9 Black action cameras as an alternative to the Android devices used by Google. They feature similar sensors (Sony CMOS sensor with relatively small pixels) but much wider lenses when compared to the main camera typically found in modern smartphones.

We captured sequences of 10 raw images using the dedicated burst mode of the HERO 8 and 9, all while trying to follow the base principles described in [Has+16] (underexposed bursts, with identical focal length and short exposure time for all images, and scene and camera motion that is manageable with respect to exposure time). Here are the HERO9 parameters we had access to in order to respect these principles as much as possible:

- We did not have direct control of the exposure time, as it is handled by the auto-exposure algorithm of the camera. Instead, we could specify the burst rate (how many images we want in a specified amount of time). We set it to its fastest value (10 images in 1s when capturing raw images), ensuring a maximum exposure time of 0.1 s.

- We could also change the EV compensation parameter (which controls how bright the burst images are). Since we wanted the bursts to be underexposed, we typically set it between 0 and -2 depending on scene brightness, selected ISO and the expected behavior of the auto-exposure algorithm.

- Finally, we could set minimum and maximum ISO values. We actually had to set both parameters to the same value if we wanted all images of the burst to feature the same ISO (had we not done that, the AE algorithm might have selected different values for different images of the same burst). This means that we had to manually pick ISO before capture. We usually set ISO to 100 in most scenarios (in order to avoid clipped highlights) except in low light where we might set it between 200 and 800.

We deliberately shot most bursts in very low light / high dynamic range scenarios to ensure these would have a lot of noise. Since GoPro cameras output raw images in their proprietary `.gpr` format, we converted them to Adobe Digital Negative (`.dng`) files using Adobe DNG Converter[17], which is the same format used for the HDR+ dataset. We then simply executed the exact same code we ran on the images provided by Google, without changing any tuning parameter.

We applied our own finishing pipeline (described in Section 1.5.2) to both the reference image and the result of the alignment and merging steps in order to assess their impact on noise reduction. We also added a minimally processed version of

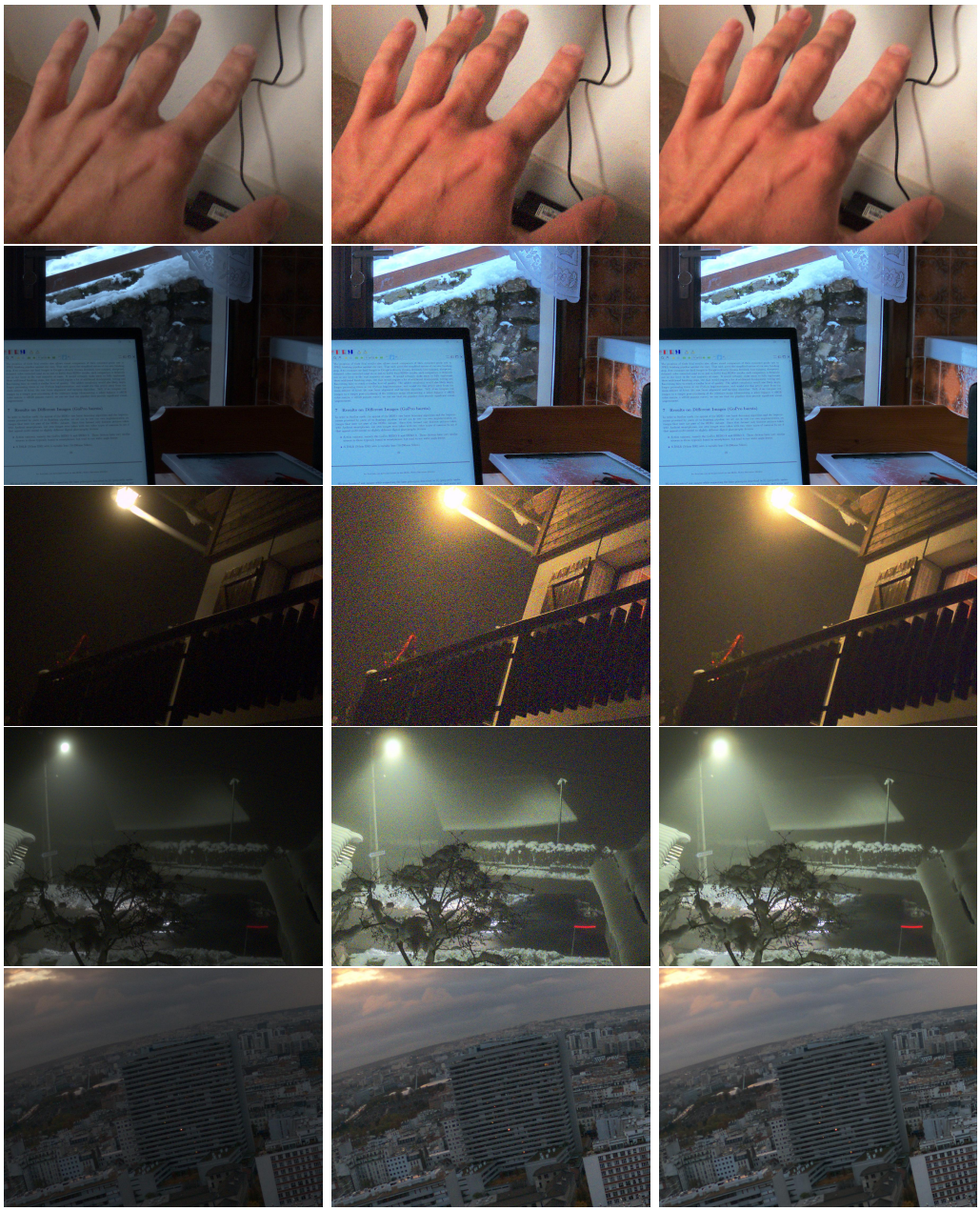
[17] https://helpx.adobe.com/photoshop/using/adobe-dng-converter.html

the reference image (with no tone-mapping) to show how bright the original raw image actually is. We can see that noise is significantly reduced by the alignment and merging procedure (even on image edges, where these steps might perform worse because of image distortion induced by the wide lens). Residual noise can however still be quite present in some images (particularly in very low light), and some of it can be amplified by the tone mapping and sharpening steps of our pipeline. Figure 1.15 showcases some of these results, while the full-size results on 17 GoPro bursts can be consulted and downloaded from Google Photos[18]. The 17 raw DNG bursts can also be downloaded from Google Drive[19].

1.8 Summary

Throughout this chapter, we presented the core principle at the heart of the HDR+ digital photography pipeline: a raw burst alignment and merging algorithm. It uses a Wiener filter variant to combine the information of aligned tiles stacks in the 2D DFT space, to produce individual tiles with significantly less noise. Little priors and image metadata are required to produce convincing results, the most important one being a crude estimation of the noise level. When tuned properly, a good compromise between residual noise and ghosting artifacts can be found, resulting in natural, visually pleasing images. Many additional steps are required after alignment and merging to obtain a final RGB image acceptable for consumer-grade products, especially when trying to stand out from competitors. When compared to more recent state-of-the art approaches, the constrained computational complexity of the whole pipeline makes it well suited for mobile photography, but the strategies employed are versatile enough to improve image quality on many different types of cameras.

[18] https://photos.app.goo.gl/XKiiSSNmqUEMYpYS8
[19] https://tinyurl.com/gopro-dng-google-drive



Reference (minimal processing) | **Reference + finishing** pipeline | **Merged + finishing** pipeline

Figure 1.15: Visual comparison of crops of the minimally processed reference image, the reference processed by our finishing pipeline, and our alignment and merging result processed by our finishing pipeline (viewers are invited to zoom in).

Chapter 2

RBDnet: turning FastDVDnet into a Raw Burst Denoising network

In Chapter 1, we presented a handcrafted raw burst denoising algorithm inside an end-to-end photography pipeline. Classsical methods like this one typically blend physics-based mathematical models and engineering tips and tricks. They also tend to feature many tuning hyperparameters that can dramatically change the output each time the algorithm runs. In this Chapter, we will explore another paradigm to solve the same real raw burst denoising problem, namely deep learning methods. It is structured as follows: Section 2.1 is a comprehensive review of the deep learning image restoration literature, including multi-image methods and the raw denoising problem. Section 2.2 describes the design choices behind RBDnet, our raw burst denoising network. Quantitative and qualitative performance of our method is discussed in Section 2.3, with additional experiments and ablation studies in Section 2.4, including comparisons to single frame variants of our model.

2.1 State of the Art of learning-based methods for single and multi image restoration

While the mechanism of backpropagating an error through linear units to learn representations was already presented in the second half of the 1980s [RHW86] and rapidly suggested for computer vision tasks [LeC+89], the resurgence of neural networks can be traced back to the 2010s, where fast-paced CPU, GPU and storage performance increases in computers, combined with easier access to large amounts of data, have allowed neural networks (especially deeper ones) to be viable (if not better performing) alternatives to standard "handcrafted" algorithms. In the 2020s, deep learning is effectively a center-stage computer science and applied mathematics research topic. Learning-based methods are also prevalent in computational image and video restoration, sometimes called *low-level vision*, where

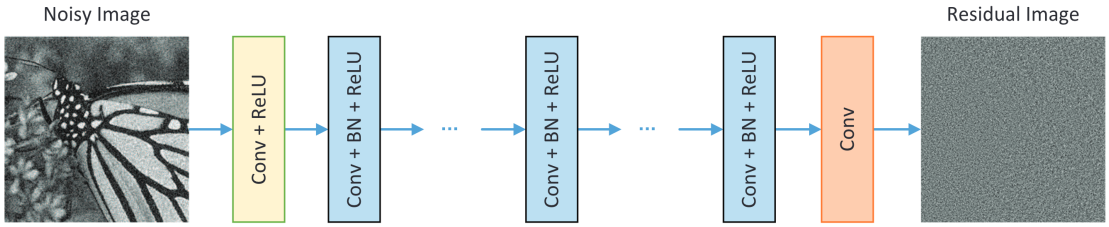


Figure 2.1: DnCNN architecture (extracted from [ZZZ18]). The first block is composed of 64 convolution filters of size $3 \times 3 \times c$ (where c is the number of channels of the input image) and a ReLU activation function, yielding 64 channels. It is then followed by $D-2$ blocks of 64 $3 \times 3 \times 64$ filters with batch normalization and ReLU activation. The final block is made of c filters of size $3 \times 3 \times 64$ and outputs the residual of the noisy image.

deep neural networks achieve state-of-the-art performance on tasks such as denoising [Bro+19; Zam+20], super-resolution [Wan+18; SC20] or deblurring [Tao+18; Car+21].

2.1.1 The rise of convolutional architectures

The ever-growing popularity of convolutional architectures for computer vision tasks such as image recognition [KSH12; SZ14; He+16] or optical flow estimation [Wei+13] in the early 2010s has been a great source of inspiration for the restoration community. In 2017, Zhang *et al.* introduced DnCNN [Zha+17a], a simple yet powerful deep CNN architecture for image noise removal. It is mainly comprised of a succession of learned convolution filters and Rectified Linear Unit (ReLU) activation functions. Key intuitions of this publication are the use of batch normalization, and the dense prediction of the noise residual instead of the noise-free image: formally, if we write the noise generation problem as $y = x + n$ with y the noisy observation, x the underlying clean image and n the noise, DnCNN learns to directly map y to the residual $R(y) = y - x = n$, and the denoised image can be recovered with $x = y - R(y)$. The architecture of DnCNN is showcased in Figure 2.1. For Gaussian denoising, DnCNN outperformed classical methods such as BM3D [Dab+07], EPLL [ZW11] or WNNM [Gu+14]. Zhang *et al.* also propose training a single version of DnCNN for three different tasks: Gaussian denoising, super-resolution, and JPEG image deblocking, which showcased competitive performance with back-then state-of-the-art methods as well.

Many works have since then replaced straightforward convolutional architectures by CNNs with a multiscale convolutional encoder-decoder layout with skip connections. This is in great part thanks to the introduction of the seminal U-Net [RFB15], initially proposed for biomedical image segmentation, but very quickly adopted in many low-level vision tasks such as denoising [Mil+18; Bro+19; Zha+21a] or deblurring [Car+21; Che+21]. The "hourglass" layout of the U-Net allows learning representations at multiple image scales (*e.g.* noise or texture at

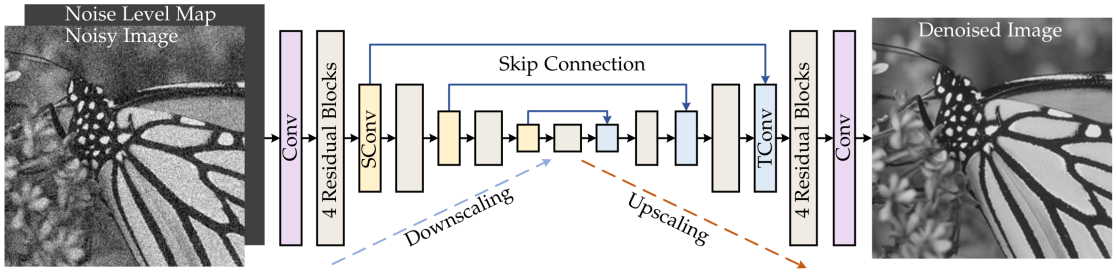


Figure 2.2: Architecture of DRUNet, a deep denoising network based on U-Net that also includes residual blocks presented in [He+16] (extracted from [Zha+21a]).

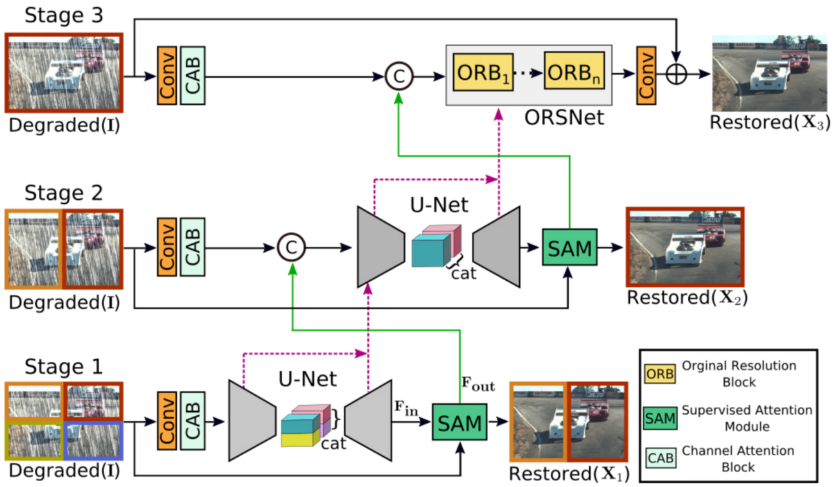


Figure 2.3: MPRNet architecture (extracted from [Zam+21]). Please refer to the original publication for additional details on the different blocks of the network.

different spatial frequencies), while the skip connections allows the preservation of higher frequency detail in spite of successive downscaling operations while limiting the vanishing gradient problem. An example of a U-Net based architecture is presented in Figure 2.2.

Some works propose more complex architectures that can include both multi-scale and single-scale subnetworks, as well as connections between these scales, such as MPRNet [Zam+21]. It is a 3-stage progressive architecture, where each stage handles different non-overlapping patches of the image and contributes to the global training loss. Stages 1 and 2 include a multi-scale architecture based on the U-Net, while stage 3 remains at full image resolution. The network also performs fusion of feature maps accross different scales, and includes channel attention blocks [Zha+18]. The multi-scale and multi-stage design of MPRNet make it a robust and powerful neural network architecture that is competitive with other state-of-the-art single image restoration methods for multiple applications such as denoising, deblurring or deraining.

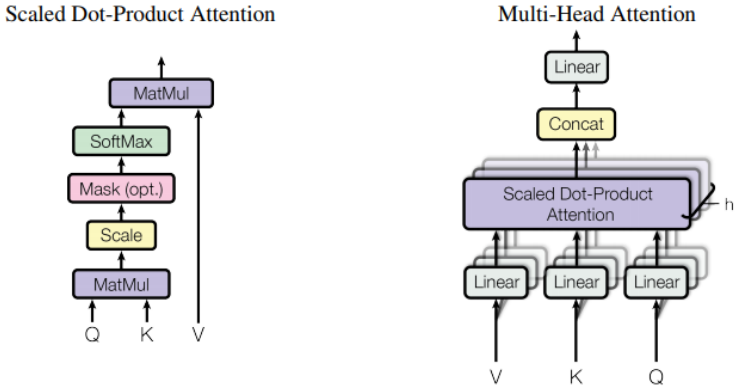


Figure 2.4: Left: Scaled Dot-Product Attention. Right: Multi-Head Attention consists of several attention layers running in parallel (extracted from[Vas+17]).

2.1.2 Trends in the restoration literature of the early 2020s

In the early 2020s, previously state-of-the art convolutional architectures for image restoration are progressively being supplanted by transformer-based architectures. At their core lies the attention mechanism first introduced in 2017 in [Vas+17]. In a nutshell, an attention function computes a compatibility between a query vector q and a set of key vectors k to produce a weighted sum of value vectors v, yielding a final output vector. In particular, scaled dot-product attention computes the dot product between the vectors q and k_i each of dimension d_k, for each possible key k_i. Each product is then normalized by $\sqrt{d_k}$ and a softmax function is applied to obtain the weigths that will be applied to v_i. Formally, if we simultaneousy evaluate multiple queries by packing them in a matrix Q, and if we pack the set of keys in a matrix K and the associated values in the matrix V, we have:

$$\text{Attention}\ (Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{2.1}$$

In [Vas+17], several scaled dot-product attention modules are put in parallel, and different learned linear combinations are applied to their queries, keys and values. The outputs of these parallel modules are then concatenated and then linearly combined using other learned weights. This block is called a multi-head attention layer. Scaled dot-product attention and multi-head attention layers are illustrated in Figure 2.4. The seminal transformer network includes multi-head attention layers along with feed-forward layers in an encoder-decoder architecture.

A key property of transformers is that they can perform similarity measurements between elements that are arbitrarily far away from each other. In contrast, convolution layers architectures mostly perform learned local filtering; CNNs usually circumvent the problem of limited receptive field of the network by adding many layers and/or by adding operations that induce changes in scale. While the transformer architecture was originally used for natural language processing in [Vas+17], following works have proposed transformer architectures for computer

vision applications by suggesting to compute attention between patches [Dos+20]. This includes image restoration, where attention-based architectures yield remarkable performance in *e.g.* denoising [Lia+21] or super-resolution [Yan+20; Lia+21]. Another interesting property of transformer networks is that they can easily incorporate multimodal information, as the attention mechanism can be computed between queries, keys and values of different shapes and/or properties. Transformers typically do not need as many training parameters as CNN architecture with similar performance, but they are usually slower and more computationally expensive as multi-head attention layers are computationally quadratic. While some works try to reduce the computational burden of visual transformers (particularly for non-dense computer vision applications [WT22; Men+22; Tan+22; Zha+22; Yin+22]), these types of architecture are still not realistic targets for on-device inference in systems with constrained computational power such as mobile phones or action cameras, especially in the case of dense prediction applications such as image or video restoration where per-pixel restored estimates must be provided.

Another interesting trend of the image restoration literature in the early 2020s is the introduction of normalizing flows and invertible networks. These neural networks map a vector of a known (typically Gaussian) distribution to a target distribution using a sequence of differentiable and invertible blocks. While this additional constraint of differentiabilty and invertibility reduces the number of possible architectures, a key property of these networks is that they can be used for both density estimation from a sample (forward mapping) as well as sampling (inverse mapping). This has shown prowess in tasks such as noise modeling [ABB19; Mal+22], super-resolution [Lug+20], or raw to sRGB / sRGB to raw mapping [XQC21].

Some publications of the restoration field propose a single architecture for multiple tasks. In general, one application and the other only differ by the nature of the degradation from clean to corrupted samples. Because neural networks are usually trained through a large scale optimization problem using a large set of examples, one might expect samples corrupted by a certain kind of degradation to lead to a local minimum of the problem, and samples corrupted by a different degradation to lead to a different minimum. It is not surprising then than most works presenting the same architecture for different restoration tasks required separate trainings [Xue+19; Lia+21; Zam+21; Zam+22; Lia+22b]. However, some publications of the early 2020s have started proposing a network architecture trained for multiple restoration applications at once. A common trait of these works is to try to explicitly distinguish samples corrupted by a certain type of degradations from samples corrupted by a different degradation. For example, [Che+22] and [Li+22a] leverage contrastive learning to push one degradation task away from the others in the training loss; a transformer architecture trained for deraining, desnowing, and dehazing using distinct weather queries is presented in [VYP22]. Potential benefits of these multi-task approaches is increased robustness of the network to unkown degradations [Li+22a], as well as reduced memory footprint, as only the weights of a single network must be stored, which can be appealing to embedded

systems.

Setting new network architectures and training techniques aside, another welcome trend in the learning-based image and video restoration scientific community is research and challenges that go beyond the pursuit of solely maximizing restoration performance in terms of metrics such as PSNR or SSIM. For example, state-of-the-art networks that maximize said metrics typically cannot run on systems with constrained memory and compute requirements, including some high-end smartphones of the early 2020s that have started featuring dedicated "neural engines" for hardware-accelerated inference of neural networks. Designing architectures that meet these requirements while providing adequate performance is a research topic of its own. In the case of image restoration for example, the NTIRE 2022 workshop has proposed an Efficient super-resolution challenge [Li+22b], where the objective is to design and train a network that yields a PSNR $>$ 29 dB with minimal runtime, size, and FLOPs. Good performers of this challenge typically used training techniques such as knowledge distillation (teach a smaller "student" network to reproduce the behaviour of a larger "teacher" network that learns the target mapping), pruning and reparametrization to meet the requirements.

2.1.3 Beyond traditional supervised network training

A majority of end-to-end deep neural networks for image restoration are trained with supervised learning:

$$\theta^\star = \underset{\theta}{\operatorname{argmin}} \sum_i L(f_\theta(x_i), y_i) \tag{2.2}$$

where θ are the learnable parameters of the neural network represented by the function f_θ, the observation x_i is a degraded version of the clean target y_i, and L is a cost function. Successfully training such networks usually requires large datasets of pairs of degraded and clean images (x_i, y_i). There are multiple instances where they can be difficult to create:

- if one chooses to synthetically create x_i from an image y_i deemed "clean", it is not easy to reproduce the degradations process of real photographs, as it can originate from multiple sources (*e.g.* motion blur, defocus blur, noise, undersampling), and each of these sources of degradation can be difficult to accurately model.

- if one instead chooses to capture pairs of real images (*e.g.* one noisy short exposure image and one long exposure image with less noise), it can be difficult to ensure than both images capture the exact same underlying scene. This is even more difficult when dealing with dynamic scenes.

However, recent advances in unsupervised and self-supervised approaches in computer vision have also been ported to image restoration tasks to circumvent these issues. With the Noise2Noise paradigm [Leh+18], two or more independent noisy

realisations of the same latent clean target can be used to train a denoising network. For example, if we use pairs of noisy images (x_i, x'_i) drawn from the same underlying distribution conditioned on the clean target y_i, the optimization problem becomes

$$\theta^\star = \underset{\theta}{\operatorname{argmin}} \sum_i L(f_\theta(x_i), x'_i), \quad \text{s.t.} \quad x_i \sim p(x \mid y_i), \quad x'_i \sim p(x \mid y_i). \tag{2.3}$$

In [KBJ19], a denoising network is successfully trained using individual noisy image patches as both input and target in conjunction with a blind-spot architecture (enforcing the use of neighboring pixels to predict the value of a pixel hidden in the input). In [UVL18], the sole structure of an untrained neural network is used to extract useful image statistics from a single degraded sample, thus allowing to solve image restoration problems without access to additional data (*i.e.* the handcrafted network structure is an image prior itself). An example application is noise removal, where the network can output the denoised image after being optimized to reconstruct the noisy image, said optimization being interrupted after a fixed number of iterations before convergence (and thus reconstruction of noise artifacts; this strategy is called early stopping). These unsupervised and self-supervised methods are typically slightly less performant than their supervised counterpart, but said counterpart might not exist in some applications as discussed before[1].

2.1.4 Deep learning for raw image denoising

On the subject of image denoising, many algorithms of the deep learning literature focus on the removal of additive white Gaussian noise (AWGN), for which variance does not depend on image content, on top of pre-existing images. This has been shown to be limited in efficiency for the denoising of real photographs [PR17], where noise inherent to the physical properties of the sensor is typically signal-dependent[2]. Moreover, it is important to be mindful of the step of the image processing pipeline at which one wishes to characterize noise (and in turn perform denoising). The stochasticity of the acquisition process can be quite accurately modeled thanks to the known physical properties of imaging devices, meaning that the noise model of raw images can be considered tractable. On the other hand, accurately modeling the noise after many image processing steps is a very difficult task: it is then typically correlated, and blended with other artifacts (compression, quantization, saturation, etc.) [Ber18].

It is no surprise then that many publications of the denoising literature have suggested handling noise modeling and denoising as early as possible in the image generation pipeline, including works presenting deep learning methods. Looking at

[1] Classical / handcrafted restoration methods usually do not require pairs of corrupted and clean images either; one might argue that the design of said methods also incorporates *a priori* knowledge of images.

[2] Applying a variance stabilizing transform such as the Anscombe transform does increase the performance of Gaussian denoising methods when applied to real photographs [MF12].

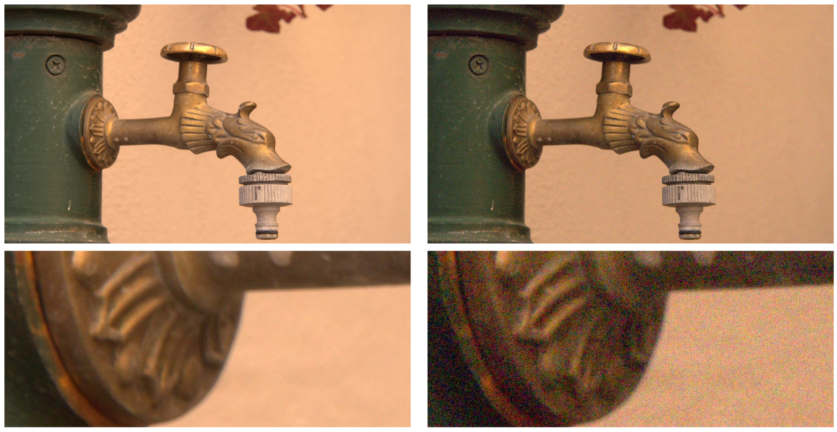


Figure 2.5: Left: clean DND sample processed to sRGB. Right: associated noisy sample (extracted from [PR17]).

methods leveraging supervised learning to train their models, the data generation approaches used therein can be divided in two main categories:

- *capture pairs of real clean and noisy images.* This concept was first introduced in 2017 with the creation of the Darmstadt Noise Dataset (DND) [PR17]. To produce a pair, the same scene is captured twice by the same camera: the reference image is captured at a low ISO, and the corresponding noisy image is captured at a higher ISO (with an appropriately adjusted higher shutter speed to keep the same exposure). A DND scene is illustrated in Figure 2.5. This benchmark is still used in the early 2020s to compare the performance of state-of-the-art denoising methods. A similar concept was used for the creation of the See-in-the-Dark (SID) dataset [Che+18] with a bigger emphasis on low-light images, where the SNR is lower. This dataset was used in the same publication to train a CNN model to directly map a noisy raw image to a clean sRGB image (using a processed version of the raw image as target during training).
  The main drawbacks of this data generation paradigm are the time-consuming aspect to the construction of these datasets (especially at large scale, which is usually required for learning-based methods to generalize well to images not seen during training), and their inherent bias towards static scenes.

- *create synthetic pairs of clean and noisy raw data.* This other approach was introduced in 2018 and refined in 2019 in two publications of UC Berkeley and Google Research [Mil+18; Bro+19]. The idea is to convert sRGB images considered clean to synthetic clean raw images using a series of mathematical operations representing the inverse of a typical raw to sRGB pipline. Noise is then added to these synthetic clean raw images according to a model that approximates real image noise. Neural networks can then either be trained using these synthetic clean and noisy raw pairs [Mil+18], or on the original sRGB clean image and reprocessed sRGB versions of the noisy raw

image [Bro+19; Zam+20].
The paradigm of realistic synthetic noisy data generation has been found and improved in many publications since then. When using a model trained on synthetic data for real denoising, the quality of the restoration is heavily dependent on the "realistic" properties of the added noise itself. While several works have used a combination of a Poissonian and a Gaussian distribution (or its signal-dependent Gaussian approximation), this model is a limited aproximation of real raw image noise. For example, thermal fluctuations of the sensor can cause spatial variations in noise statistics; quantization of noise comes into play during sensor readout; and additional, not i.i.d noise sources are non-negligible in some specific capture scenarios such as the presence of row noise in very low light. Two main techniques have been proposed to synthesize more realistic noisy data: leverage more complex and realistic noise models [Wei+20] or leverage the power of data-centric generative models [ABB19; Cha+20; Mon+22a].
The main benefit of synthetic data generation is its reduced cost when compared to the capture of pairs of real images, and one of its main risks is the potential domain gap between synthetic and real images. Some publications actually bridge the gap between the synthetic and the real raw approaches by adding synthetic noise to real raw data. In [Wan+20], synthetic noise is added to raw long exposure images of the SID dataset according to calibrated noise parameters of a target device, the Oppo Reno 10x smartphone. A mobile-friendly U-Net like architecture is then trained on the generated pairs of noisy and clean raw images and is shown to provide good restoration performance with reasonable memory and compute requirements, allowing it to be included as the night shot feature of several flagship smartphones released in 2019.

2.1.5 Deep networks for burst and video restoration

As with classical methods, some learning-based restoration methods also try leveraging the information of multiple frames to further improve the quality of their output. In [Mil+18], an architecture based on U-Net takes a concatenated noisy raw burst as input, and outputs per pixel 3D kernels that are convolved with the input to output a single image. In [AD18], an architecture also based on the U-Net is used for deblurring bursts of abitrary sizes, where copies of the autoencoder process each frame, but information is shared between copies using max pooling and concatenation of these common features to local features of each copy. These publications can be categorized as *multi-frame* approaches in the sense that they directly take several consecutive images as input. By contrast, other publications have proposed *recurrent* architectures to leverage the temporal redundancy of image sequences. Deep Burst Denoising [GMU18] uses a "two track recurrent architecture" for sRGB denoising. The first track is dedicated to single frame denoising, and features produced in this track are fed as inputs to the second

track, which uses recurrent connections to combine these features with the previous state of this track (which encompasses features of the previous frame of the burst) Efficient Multi-Stage Video Denoising with Recurrent Spatio-Temporal Fusion [Mag+21] also proposes a recurrent CNN architecture for raw video denoising. It combines learnable color and frequency transforms, temporal fusion of frames using a recursive convex combination at two different scales, and a spatial denoising and refinement modules. Variations of the architecture with different numbers of convolutions and filters are proposed, and while the bigger variants are competitive with state-of-the-art methods such as [Wan+19; Yue+20], the publication emphasizes on smaller variants that retain acceptable performance for a significantly smaller computational footprint: their 5 GFLOPs model yields 42.63dB of PSNR on the CRVD raw multi-frame dataset [Yue+20], compared to the 1965 GFLOPs of RViDeNet [Yue+20] and 3089 GFLOPs of EDVR [Wan+19] that yield a PSNR of 44.08 and 44.71 respectively. This smaller model is ported to a Huawei p40 Pro smartphone and runs locally at around 30fps for a single-precision 720p.

An additional complexity of multi-image restoration algorithms is misalignment of images. Some methods require registration by an external, off-the-shelf algorithm before or during network inference [GMU18; Mil+18; TDV19b]. Other methods include dedicated motion estimation / alignment modules within their network architecture [Xue+19; Yue+20; Bha+21; Dud+22]. Other methods have skipped the image registration step altogether, arguing that a CNN with a sufficiently large receptive field should be able to implicitly align images or feature maps thanks to its many successive learned convolution filters. For example, the two-stage architecture with U-Net like blocks of FastDVDnet [TDV20] has shown remarkable performance in Gaussian video denoising, both in terms of restoration quality and temporal stability, without any explicit alignment. It has been shown in a publication by different authors [She+21] that FastDVDnet performs adaptative spatio-temporal filtering, and that the shift induced by said adaptive filter (obtained by differentiating the network) matches optical flow-based motion estimation applied on the same noise-free sequence fairly well.

Trends of image restoration literature of the early 2020s have also trickled down to multi-image publications. The unsupervised Noise2Noise [Leh+18] paradigm was a direct inspiration of Frame-to-Frame [Ehr+20] for video denoising, considering that consecutive video frames are observations of the same underlying clean image provided one frame is warped to the other after optical flow estimation. In a follow-up work, a self-supervised approach was proposed in [Dew+21] to fine-tune pre-existing multi-image denoising networks so that they better generalize to noise models not seen during training. Inspired by the blind-spot training of Noise2Void [KBJ19], a multi-image denoising network based on FastDVDnet was trained without requiring clean data [She+21]. Transformer-based architectures are also gaining popularity in video restoration and outperforming CNN-based architectures: given that patch-based attention can be computed between elements that are arbitrarily far away from each other, these architectures can easily leverage large spatio-temporal windows. Transformer architectures have been pro-

posed for tasks such as video denoising [Lia+22b], super-resolution [Lia+22a], deblurring [JY22], or frame interpolation [Shi+22], but remain computationally expensive for embedded applications.

2.2 Designing a Raw Burst Denoising Network

Having presented some of the relevant publications and methods of the literature of deep learning for the real raw burst denoising problem, we will now present how we designed our own network in this Section. It builds upon the work of a previous PhD collaboration between GoPro and the MAP5 laboratory, namely Tassano, Delon, and Veit's FastDVDnet [TDV20], a network originally designed for Additive White Gaussian Noise (AWGN) removal of standard sRGB videos. We show that we can adapt it to real raw burst denoising with little changes in the architecture by focusing on relevant training data for multi-image raw denoising. Two main approches are studied: synthetic noisy raw data generation from sRGB image sequences, and the use of real paired clean and noisy raw data captured in a limited and controlled setting.

2.2.1 Background: FastDVDnet architecture

To produce a denoised version of the video frame at time t, FastDVDnet uses frames at times $t-2$, $t-1$, t, $t+1$ and $t+2$. Because of this, in order to denoise a whole video while ensuring the output video retains the same size, the noisy input video $\boldsymbol{v}$ is increased by 4 frames by mirroring: $\boldsymbol{v} = \{f_0, \dots, f_{N-1}\} \rightarrow \boldsymbol{v}' = \{f_2, f_1, f_0, \dots, f_{N-1}, f_{N-2}, f_{N-3}\}$.
To produce a denoised estimate of the center frame from a 5 frame window, the network uses two instances of a Denoising Block, a U-Net like autoencoder with residual connections (Fig. 2.6a). These blocks are cascaded in a two-stage manner (Fig. 2.6b):

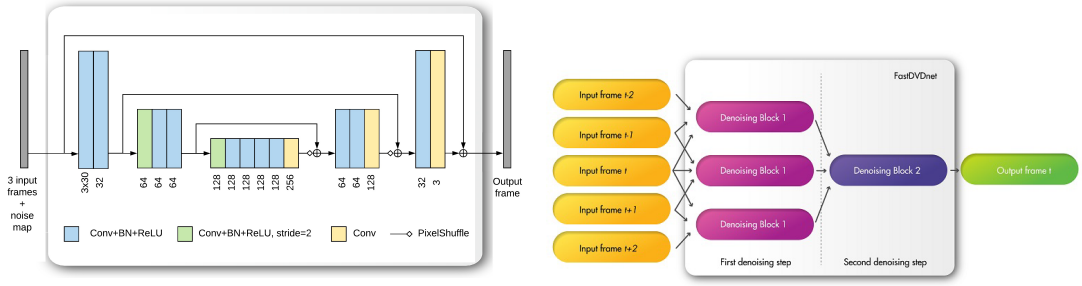
1. the first block is used to denoise 3 sequences of 3 consecutive frames: $[f_{t-2}, f_{t-1}, f_t]$, $[f_{t-1}, f_t, f_{t+1}]$, and $[f_t, f_{t+1}, f_{t+2}]$
2. the second block uses the three outputs of the first stage as input and outputs the final denoised version of f_t.

FastDVDnet also takes an estimation of the standard deviation of the noise as additional input, implemented as a constant per-pixel noise map. This noise map can also be used ad hoc to control the strength of the denoising [TDV19a].

2.2.2 From noisy sRGB videos to noisy raw bursts

Domain gap: sRGB video vs raw bursts.

The differences between processed videos and sRGB image bursts are pretty minor: the main difference is compression artifacts of video files typically not present



(a) architecture of a Denoising Block (b) two-stage architecture

Figure 2.6: FastDVDnet architecture. It comprises two instances of a Denoising Block, used in a cascaded, two-stage fashion (extracted from [TDV20])

in bursts, but we'll consider these negligible for the rest of this chapter. However, the domain gap is larger between processed sRGB images corrupted with AWGN, and raw images containing real noise caused by the photon acquisition process and the physical properties of the imaging device. Raw images also have greater, uncompressed dynamic range, and typically do not feature degradation and artifacts due to subsequent image processing steps (demosaicking, color correction, gamma compression, sharpening, etc.), while sRGB images do.

Noise model: Gaussian vs clipped heteroscedastic Gaussian.

While ubiquitous in the denoising literature, the AWGN model is not suitable to approximate the noise present in real images. A better approximation is the Poisson-Gaussian (PG) model [Foi+08], where the noise is composed of two mutually independent parts: a Poissonian signal-dependent component, and a Gaussian signal-independent component. This model can be further approximated by a heteroscedastic Gaussian distribution: a normal distribution whose variance can be expressed as an affine function of the signal level. Formally, given x_p, the intensity of a clean image at pixel location p, we have:

$$\sigma^2(x_p) = \lambda_s x_p + \lambda_r \tag{2.4}$$

where (λ_s, λ_r) are camera and ISO-specific noise parameters: λ_s is characteristic of photonic / shot noise; λ_r characterizes read noise (encompassing various sources such as sensor readout noise, thermal noise, dark current noise, etc.)[3]. Notice that Gaussian noise is a subset of that model with $\lambda_s = 0$: in theory, a network trained for Poisson-Gaussian noise should be able to denoise images corrupted with Gaussian noise, while the opposite is not guaranteed. Note that we clip images after adding noise in order to remain consistent with the black level and white level values of the original image.

[3]Poisson-Gaussian and heteroscedastic Gaussian noise will be used interchangeably to describe the model of Equation 2.4 for the remainder of this chapter.

Figure 2.7: Unprocessing pipeline as performed in [Bro+19].

2.2.3 Training a raw burst denoising model

We have presented the design principles of RBDnet, our multi-frame raw burst denoising model. Other than the number of input noise maps and the number of input image channels, the architecture of RBDnet is identical to FastDVDnet: it still uses 5 consecutive images and two cascaded U-Net blocks with skip connections, and residual learning to produce an estimate of the noise-free central image. We now present the training data and parameters of the two main versions of the network: one is trained on synthetic noisy raw data created from sRGB videos, while the other is trained on a supervised dataset of paired noisy and noiseless raw stop-motion sequences.

Synthetic noisy raw bursts

For our first version, we use the same sequences as in the original FastDVDnet publication for training, namely 480p versions of videos of the DAVIS 2017 dataset [Pon+17]. We transform them into synthetic raw image sequences before adding heteroscedastic Gaussian noise. To do so, we use the sRGB to raw unprocessing pipeline presented in [Bro+19]. In this paper, a camera Image Signal Processing pipeline (ISP) that transforms a raw image to a processed sRGB image is simulated through a series of simple mathematical operations. These operations are representative of typical image processing blocks: demosaicking, white balance, digital gain, camera RGB to sRGB color matrix, gamma, contrast-enhancing tone curve. The reverse mathematical operations are also described, and these can be used to create synthetic data similar to raw images from standard sRGB data. In such images, intensities change linearly with respect to illumination change, only one color channel is available per pixel according to a Bayer color filter array, and neighboring pixels are less correlated by subsequent image processing steps. This process, which is called *unprocessing*, is illustrated in Figure 2.7.

We thus generate synthetic raw training patches from standard sRGB videos with the following operations:

- we extract 5 consecutive frames from a video
- we extract sRGB patches of size $(w, w, 3)$ by cropping patches at a random

spatial location (same location for all 5 frames)

- we unprocess the patches to obtain $(w/2, w/2, 4)$ synthetic clean raw patches that will be the ground truth
- we add clipped heteroscedastic Gaussian noise to obtain the corresponding noisy patches. Consistent with other algorithms of the literature [Mil+18; Bro+19], the corresponding noise map is constructed with a per-pixel estimation of the standard deviation of the noise given the intensity of the observed noisy pixel $\tilde{x_p}$:

$$\hat{\sigma}(\tilde{x_p}) = \sqrt{\lambda_s \max(0, \tilde{x_p}) + \lambda_r}. \tag{2.5}$$

  During model training, we use this estimate instead of the ground truth variance to allow denoising of images where the underlying clean image is not known (which is obviously the case for real images).

Noise parameters of real raw datasets. In order to carefully select (λ_s, λ_r), the parameters of the PG noise we add to our training samples, we conducted a small survey of the noise parameters of raw images of publicly available datasets (both mono-frame and multi-frame). The datasets surveyed here are the Darmstadt Noise Dataset (DND) [PR17], CRVD [Yue+20], the Smartphone Image Denoising Dataset (SIDD) [ALB18], and a subset of the HDR+ dataset [Has+16]. For each dataset, we retrieve the noise parameters either from the original article, the associated code, or directly in the raw image metadata (e.g. in some files of the HDR+ dataset, λ_s and λ_r can be directly found in the `NoiseProfile` DNG tag). Noise parameters of the aforementioned datasets are shown in Figure 2.8, which clearly shows that the space of noise parameters featured in popular denoising datasets is pretty broad. In order to improve the generalization capabilities of our model, we thus decided, for each training sample, to randomly draw noise parameters in the rectangle $(\lambda_s \in [10^{-4}, 10^{-2}], \lambda_r \in [10^{-8}, 10^{-3}])$, which covers most of the samples featured in Figure 2.8. Covering a wide range of noise parameters might yield lower performance when compared to a model trained for a specific noise level or camera-specific noise parameters, but said model might perform worse at noise levels not seen during training.

Training details. We train our model using 128×128 RGB crops (which produces 4 64×64 Bayer [R, Gr, Gb, B] channels after unprocessing). This crop size is increased when compared to the 96×96 RGB crops of the original FastDVDnet, because the generated raw patches are 2 times smaller after unprocessing, and larger patches typically yield better restoration performance. We also train our model with L1 loss (instead of L2 for the original FastDVDnet) because it has been shown to be beneficial for image restoration tasks [Zha+17b] and is commonly used in the raw denoising literature [Bro+19; Yue+20]. Other training parameters are unchanged compared to the original FastDVDnet model (80 epochs; batch size of 64; random 90° rotations and flips; learning rate of 10^{-3} for the first 50 epochs, 10^{-4} until epoch 60 then 10^{-6} until end of training; bias-free convolutions; SVD orthogonalization of filters every 50 batches until epoch 60).

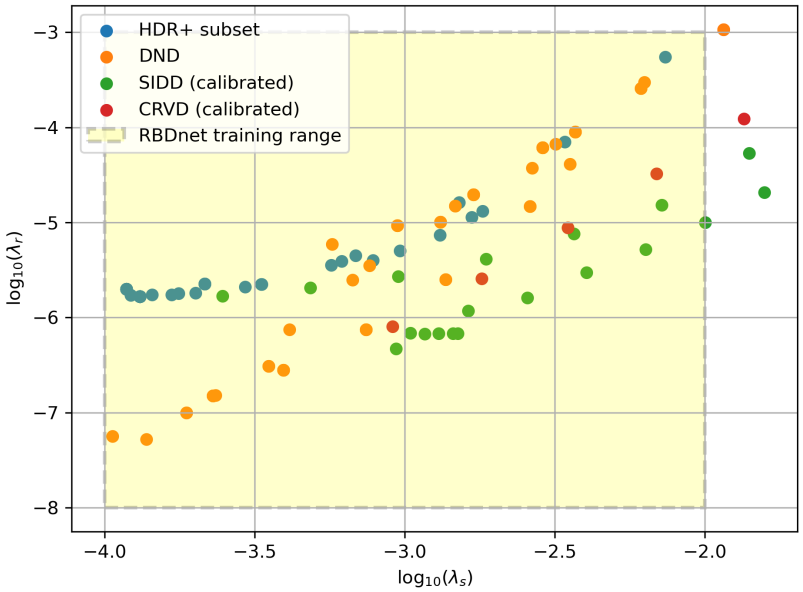


Figure 2.8: Noise parameters of images in the SIDD, DND, CRVD and HDR+ datasets

Real noisy raw bursts

We train a second version of our model on real raw image sequences. To do so, we use the Captured Raw Video Dataset (CRVD) [Yue+20]: it features 11 indoor 7-frame raw sequences, and each sequence is repeatedly captured at ISO 1600, 3200, 6400, 12800 and 25600. This data can be used to train a multi-frame denoising model in a supervised fashion, as each noisy raw image has a corresponding "ground truth" image, generated by averaging multiple captures of the exact same image content combined with slight additional spatial denoising performed using BM3D. Two noisy frames of one such scene and the corresponding ground truth are illustrated in Figure 2.9.

Contrary to the noise added in our synthetic data generation pipeline, the noise in CRVD images is actually part of the capture process. While training a model on real raw image sequences might seem unconditionally better than synthetic raw data at first glance, the CRVD dataset is not without its own drawbacks. Indeed, because of the strenuous protocol required to obtain pairs of noisy and noise-free multi-image sequences, the number of scenes with ground truth is limited to 10 sequences of 7 consecutive frames, shot indoors, with static illumination, no camera or object motion during capture, and at a finite set of camera and ISO specific noise levels. This could impair model generalization to real raw bursts captured in the wild.

Training details. Given that we use a different dataset with a different number of training samples, we set training hyperparameters of this version of RBDnet by mixing-and-matching those of our synthetic version with UDVD, another multi-

Figure 2.9: 2 consecutive 1920 × 1080 frames of a CRVD indoor scene at ISO 25600. Top row: noisy images. Second row: clean images. (best viewed digitally; images processed to sRGB for visualization)

image model trained on CRVD [She+21]. We use all training sequences, *i.e.* scenes 1, 2, 3, 4, 5, and 6 at ISO 1600, 3200, 6400, 12800 and 25600. Even though the real noise in these images does not stritcly follow the PG model [Wei+20], we still construct noise maps using the per-pixel standard deviation estimation of Equation 2.5 and calibrated, ISO-specific values of (λ_s, λ_r) for the camera used in the CRVD dataset[4]. We extract all possible 128 × 128 crops with a stride of 64 pixels of the original 1920 × 1080 Bayer images (yielding 4 64 × 64 Bayer channels after stacking). The other training parameters are the following: L1 loss; 4 epochs; batch size of 8; learning rate of 10^{-4} for the first 2 epochs, 5×10^{-5} until epoch 3 then 2.5×10^{-5} until end of training; bias-free convolutions; SVD orthogonalization of filters every 50 batches until epoch 3.

2.3 Results

In the following Section, we compare the denoising performance our RBDnet models trained on synthetic versus real raw data, both quantitatively and qualitatively.

[4] https://github.com/cao-cong/RViDeNet/blob/master/pdf/supplementary_material.pdf
https://github.com/cao-cong/RViDeNet/blob/master/train_pretrain.py

Table 2.1: Denoising: PSNR on unprocessed `gopro_540p`

ISO	1600	3200	6400	12800	25600
noisy	39.47	36.49	33.58	30.58	27.58
RBDnet (DAVIS)	**43.50**	**41.52**	**39.67**	**37.85**	**36.13**
RBDnet (CRVD)	41.54	39.80	38.09	36.37	34.64

2.3.1 Quantitative results

Raw bursts synthesized from `gopro_540p` sequences

We first evaluate the performance of both RBDnet variants on a dataset of synthetic raw bursts. This dataset is created using the sRGB videos of the `gopro_540p` dataset originally used as validation dataset in the original FastDVDnet paper. We start by extracting all possible bursts of 5 frames from the `hypersmooth`, `motorbike`, `rafting` and `snowboard` sequences. Each burst is then turned into a synthetic raw burst using the "unprocessing" methodology described in Section 2.2.3 (on full-size 960×540 images instead of crops), and PG noise is added to produce the noisy versions. We select the central frame of each burst in its noise-free, synthetic raw version as ground truth. Given that our models are trained to handle multiple levels of noise, we repeat the noisy burst creation process at 5 noise levels, each corresponding to a certain ISO setting of a certain camera. We re-use the camera-specific noise parameters of the CRVD dataset corresponding to ISOs 1600, 3200, 6400, 12800 and 25600. The results are shown in Table 2.1. While both versions perform well on images not seen during training, it is clear that on this synthetic dataset, the version trained on synthetic raw data performs better (around +1.5dB PSNR) than the model trained on the real noisy images of CRVD. This performance delta can be explained by a smaller domain gap between unprocessed video sequences of DAVIS and `gopro_540p` when compared to the stop-motion CRVD Bayer data, as well as the domain gap between heteroscedastic Gaussian noise and real raw image noise; the values of the calibrated noise parameters of the CRVD dataset (which were provided by its authors) can also negatively impact performance when using synthetic data, as the by nature imperfect estimation of said parameters is performed a posteriori on real data, while they are considered perfectly calibrated when used to generate synthetic noise.

Real raw bursts of the CRVD dataset

We also compare the denoising performance of both versions of RBDnet on indoor test sequences of the CRVD dataset. While neither model has seen these images during training, these sequences are very similar to the ones seen by the model trained on the other CRVD sequences. Unsurprisingly, Table 2.2 shows that the model with the smallest domain gap performs better. That said, it is worth noting that the other version of RBDnet also performs quite well (the performance delta

Table 2.2: Denoising: PSNR on CRVD indoor test sequences

ISO	1600	3200	6400	12800	25600
noisy	38.57	35.15	31.98	28.08	26.32
HDR+ [MDV21]	41.20	38.77	37.77	35.48	33.28
RBDnet (DAVIS)	47.10	45.26	43.35	40.51	40.38
RBDnet (CRVD)	**47.62**	**45.88**	**43.91**	**41.34**	**41.21**

being smaller than 1dB at every ISO), even though it was only trained on synthetic data that is far easier to acquire, at a larger range of possible noise levels to boot.

As a bonus, we added to our comparison a classical, non-learning-based method in the form of HDR+ [Has+16] (more specifically our open-source reimplementation of Chapter 1). The algorithm is fed with 5 raw frames and the associated noise parameters. While this method does improve PSNR over the noisy input significantly at every ISO, its denoising performance is significantly lower than both RBDnet models (the PSNR gap being systematically greater than 5 dB). Bear in mind that contrary to our deep models, HDR+ is a classical method that requires hyperparameter setting, and that its default values have been selected empirically on a limited number of examples (and are not guaranteed to be optimal regardless of the capture setting and/or ISO).

2.3.2 Qualitative / visual results

Our quantitative evaluation shows that the data selected for training will directly impact the denoising performance of our multi-frame raw denoising model, but this also depends on the data selected for said quantitative evaluation, making a conclusive pick of a "better model" difficult. In this Section, we show through visual assessment that both RBDnet variants do remove noise, but have their own strengths and weaknesses. We also show that both models generalize well to a dataset neither has seen during training.
Before we dive in comprehensive visual comparison, we wanted to let the reader know that out of all models we trained for this chapter, we noticed that only one of them (RBDnet trained on unprocessed DAVIS with per-channel per-frame noise maps) sometimes produced strange artifacts near high intensity and high contrast regions. Examples of said artifacts are shown in Figure 2.10. In our experiments, training the model during additional iterations or intializing training with a different random seed might have reduced the occurence of these artifacts, or changed the bursts they appear on, but did not make them disappear completely. These artifacts did not occur with any of the other models we trained (models trained on real data, models with different noise maps, single-frame models, etc.). We did find that adding dithering to the synthetic raw data generation pipeline made those artifacts disappear, which we'll discuss in Section 2.4.4. Nevertheless, we will focus on images that do not exhibit these artifacts for the rest of this

Figure 2.10: Artifacts observed on some bursts after denoising with RBDnet trained on unprocessed DAVIS with per-channel per-frame noise maps.

section.

CRVD outdoor noisy sequences

We first inspect the denoising capabilities of our models on another subset of CRVD: its outdoors sequences. Unlike the indoors subset, these raw multi-image sequences are captured in an uncontrolled setting and thus feature illumination change and object motion (but no apparent camera motion, as it is likely fixed on a tripod). This subset features 10 sequences, each being shot at ISO 1600, 3200, 6400, 12800 and 25600 (two versions of a given scene shot at a different ISO might feature different image content, because of different exposure times and potentially different moments of capture). Most of the scenes are captured at night and/or in low light environments to emphasize lower-SNR scenarios, and images have no noise-free / ground truth counterparts because of the uncontrolled capture setting. In order to visualize the noisy and denoised raw images, we perform minimal processing with fixed arbitrary white balance, bilinear demosaicking, identity color matrix, sRGB gamma compression and 8 bit quantization.

An example of denoising a raw burst of 5 frames extracted from a CRVD sequence with our two RBDnet variants is shown in Figure 2.11. We again include the HDR+ implementation of [MDV21] with the hyperparameter values of its online demo as a non learning-based comparison point. We typically notice that

(a) noisy (b) HDR+ [MDV21]

(c) RBDnet (DAVIS) (d) RBDnet (CRVD)

Figure 2.11: Burst denoising of frame 2 of CRVD sequence 4 (ISO 6400; all models take 5 1920 × 1080 Bayer frames as input; 500 × 420 crops are shown here)

(a) noisy (b) HDR+ [MDV21]

(c) RBDnet (DAVIS) (d) RBDnet (CRVD)

Figure 2.12: Burst denoising of frame 2 of CRVD sequence 3 (ISO 25600; all models take 5 1920 × 1080 Bayer frames as input; 720 × 480 crops are shown here)

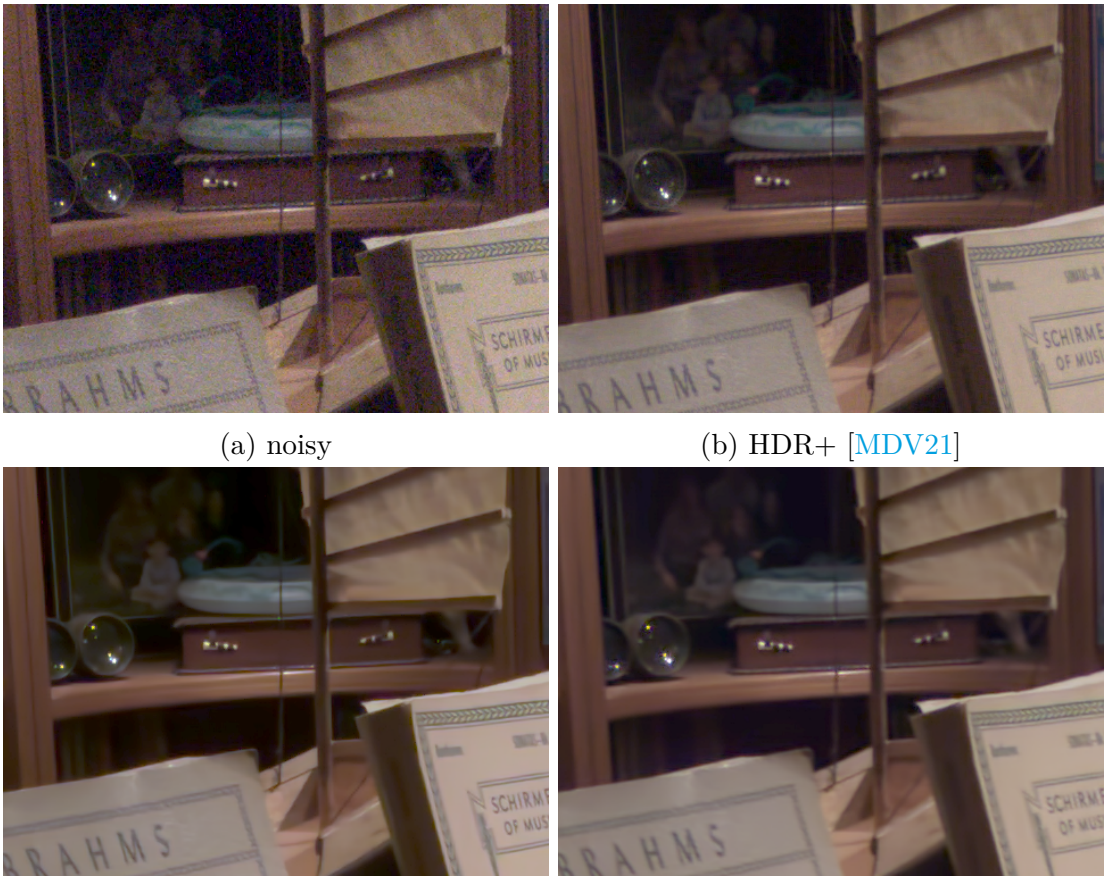
the model trained on the synthetic raw data tends to keep a bit more high frequency detail, even though it was trained on lower resolution data. At high ISOs, HDR+ sometimes produces ghosting artifacts (characteristic of too agressive temporal denoising, as in Figure 1.4c of Chapter 1) and ringing artifacts on high contrast edges (characteristic of too agressive spatial denoising, similar to Figure 1.4c of Chapter 1), as mentioned in [Has+16] and [MDV21].

Real raw bursts of the HDR+ dataset

We also visually assess the performance of our RBDnet models on 8 real raw bursts of the HDR+ dataset [Has+16][5] that have at least 5 frames and noise parameters (λ_s, λ_r) embedded in raw image metadata. Like the outdoor CRVD sequences, images have no ground truth counterparts. The bursts of the HDR+ dataset differ from CRVD outdoor sequences in that they are captured both indoors and outdoors, at various noise levels, lighting conditions and time of day, using several different smartphone cameras in a handheld fashion. As such, they can feature various amounts of noise, camera and scene motion, and illumination change.

Visual results are showcased in Figures 2.13 and 2.14. Both our RBDnet models perform very well on this dataset. HDR+ (tuned with rather conservative hyperparameters) tends to retain as much high frequency detail as possible, at the expense of stronger residual noise. As with CRVD outdoor sequences, we notice that the model trained on the synthetic raw data tends to produce sharper images; recall that said model was trained on a much broader range of noise levels, while the model trained on CRVD is limited to the 5 camera-specific noise levels of its dataset.

[5] http://www.hdrplusdata.org/



(a) noisy (b) HDR+ [MDV21]

(c) RBDnet (DAVIS) (d) RBDnet (CRVD)

Figure 2.13: Burst denoising of frame 2 of burst `9bf4_20150824_210544_967` (ISO 816; all models take 5 4080 × 3028 Bayer frames as input; 480 × 360 crops are shown here)

(a) noisy (b) HDR+ [MDV21]

(c) RBDnet (DAVIS) (d) RBDnet (CRVD)

Figure 2.14: Burst denoising of frame 2 of burst `33TJ_20150820_190702_015` (ISO 1877; all models take 5 4208×3120 Bayer frames as input; 540×540 crops are shown here)

2.4 Ablation studies and additional experiments

In this Section, we evaluate individual components of RBDnet, and conduct additional experiments that give us more insights on how to design a raw burst denoising network.

2.4.1 Ablation: 1 vs 5 frame models

Intuitively, and because of how the architecture of RBDnet is designed (it takes 5 frames as stacked inputs), we think that the network uses additional information in the surrounding 4 frames to produce an estimation of the denoised center frame. It has also been shown in [She+21] that a network trained on 1 or 3 frames performs worse than a 5 frames one. In a similar fashion, we evaluate the usefulness of using multiple frames by training RDnet, a single frame denoising network made of a single denoising block. The original denoising block takes three frames concatenated with the noise maps as input, so we modify the first convolution layer input to a single frame concatenated with its corresponding noise map. Given that RBDnet uses two blocks, RDnet has approximately 2x less parameters, which is expected to yield lower denoising performance. In order to make sure that a smaller number of parameters is not the sole cause of degraded performance, we also train another single frame model made of a single denoising block that features more convolutions (48 channels at the first U-Net scale, 92 channels at the second scale, and 184 at the third scale, instead of 32, 64 and 128 channels in the standard denoising block). This model, which we call RDnet_x2, has approximately the same number of parameters as RBDnet . We train both RDnet and RDnet_x2 using the exact same hyperparameters and training samples as RBDnet, but we only feed the center frame to the network instead of the 5 frame sequence.

Quantitative denoising performance of RDnet and RDnet_x2 on synthetic raw bursts of `gopro_540p` and on CRVD indoor test sequences are presented in Table 2.3 and Table 2.4. The performance of RBDnet is presented again here for comparison purposes. We also report the number of learnable parameters, as well as the number of Multiply-ACcumulate operations (MACs)[6] of each model to indicate respectively architecture and numerical complexity. We can see that RBDnet consistently outperforms both single-frame models, even the one with a similar number of parameters, validating the intuition that it leverages the information of surrounding frames to produce a better estimation of the denoised center frame. The quantitative performance gap between RBDnet and RDnet_x2 is small on synthetic data (typically between 0.1 and 0.2 dB), but larger for real, higher resolution data (typically more than 0.5 dB on CRVD). This better denoising performance comes at the expense of additional numerical complexity (and thus higher runtimes), which is expected given that RBDnet is a convolutionnal neural network that takes 5 frames (and their associated noise maps) as input

[6] 1 MAC $\simeq$ 2 FLOating Point operations (FLOPs) as discussed here: https://github.com/Lyken17/pytorch-OpCounter/tree/master/benchmark.

Table 2.3: Denoising: PSNR of single vs multi frame models on unprocessed `gopro_540p`

ISO	1600	3200	6400	12800	25600	# params	MACs
noisy	39.47	36.49	33.58	30.58	27.58		
RDnet	42.90	41.00	39.16	37.30	33.51	1.22M	18.4G
RDnet_x2	43.28	41.39	39.56	37.72	35.97	2.53M	38.6G
RBDnet	**43.50**	**41.52**	**39.67**	**37.85**	**36.13**	2.49M	85.2G

Table 2.4: Denoising: PSNR of single vs multi frame models on CRVD indoor test sequences (all models trained on DAVIS+unprocessing)

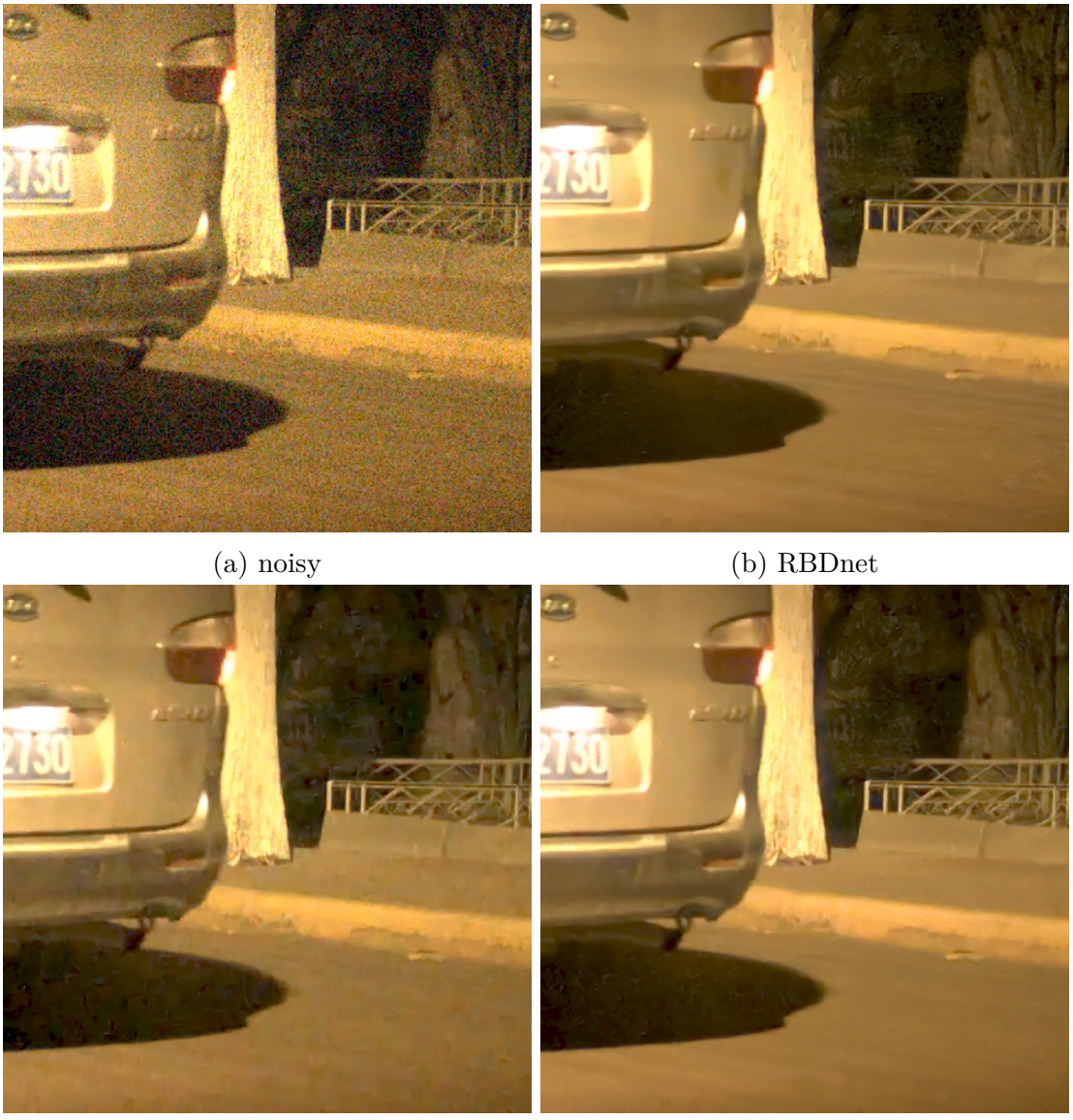
ISO	1600	3200	6400	12800	25600	# params	MACs
noisy	38.57	35.15	31.98	28.08	26.32		
RDnet	46.25	44.13	41.97	39.12	38.47	1.22M	73.3G
RDnet_x2	46.70	44.61	42.64	39.49	39.86	2.53M	153G
RBDnet	**47.10**	**45.26**	**43.35**	**40.51**	**40.38**	2.49M	338G

and performs three separate forwards in the first denoising stage, while the other models only take a single frame and perform a single forward. Not discussed here is the ability of the cascaded 5 frames architecture to produce more temporally consistent results when dealing with long frame sequences in a sliding window fashion; while this is a key characteristic for video restoration (the original use case of FastDVDnet), it is less relevant for burst denoising (temporal consistency is not relevant when dealing with one burst at a time), although the architecture of RBDnet could very much be used for Bayer video denoising.

A visual comparison of the output of RDnet and RDnet_x2 compared to that of RBDnet is presented in Figure 2.15. Although both RBDnet and RDnet_x2 vastly outperform RDnet (recall that it has twice as few parameters), the multi-frame model retains a bit more high frequency detail without featuring more residual noise.

2.4.2 Ablation: noise maps

Another experiment we wanted to conduct was evaluating the influence of the way noise maps are constructed on the final raw denoising performance of the model. In the original FastDVDnet, the estimation of the input noise level is constructed as a constant, single channel noise map, concatenated with each input frame of each denoising block. Given that under the heteroscedastic Gaussian model, the estimated noise variance is dependent on the input image intensity values, and that said values are different accross all frames and channels of a burst, the intuition is that giving per frame, per channel noise maps (i.e. $5 \times 4 = 20$ noise maps in the case of raw burst denoising) as additional input to RBDnet should lead to maximal



(a) noisy (b) RBDnet

(c) RDnet (d) RDnet_x2

Figure 2.15: Multi-frame vs single-frame denoising of frame 2 of CRVD sequence 1 (ISO 25600; all models are trained on the same DAVIS samples; 480×480 crops are shown here)

Table 2.5: Denoising: PSNR vs number of noise maps on unprocessed `gopro_540p`

ISO	1600	3200	6400	12800	25600
noisy	39.47	36.49	33.58	30.58	27.58
RBDnet (blind)	42.98	41.24	39.50	37.75	36.04
RBDnet (5× cst. (λ_r, λ_s))	43.32	41.47	39.67	37.87	36.12
RBDnet (5×1 maps)	**43.59**	**41.60**	**39.74**	**37.92**	**36.20**
RBDnet (5×4 maps)	43.50	41.52	39.67	37.85	36.13

Table 2.6: Denoising: PSNR vs number of noise maps on CRVD indoor test sequences (all models trained on DAVIS+unprocessing)

ISO	1600	3200	6400	12800	25600
noisy	38.57	35.15	31.98	28.08	26.32
RBDnet (blind)	47.01	45.26	43.35	40.50	39.91
RBDnet (5× cst. (λ_r, λ_s))	47.00	45.24	**43.39**	**40.57**	**40.44**
RBDnet (5×1 maps)	**47.20**	**45.29**	43.34	40.43	40.14
RBDnet (5×4 maps)	47.10	45.26	43.35	40.51	40.38

performance[7]. We verify that intuition by comparing the denoising performance of a RBDnet model that only uses a single noise map per input frame (we arbitrarily set it to an estimate of the noise variance of the first green Bayer channel of each input frame), and we train a third, blind model to verify if any number of noise maps is better than none for raw burst denoising. We also experiment with feeding the network with constant maps of λ_r and λ_s (2 constant maps per input frame) instead of augmenting the burst frames with per-pixel, signal-dependent estimates of the noise; in theory, given the noise model of 1.7, those maps are sufficient for the network to get the exact same information about the input noise.

Table 2.5 and Table 2.6 showcase the denoising performance of RBDnet models depending on the number and the kind of input noise maps. A first observation is that the blind model is typically outperformed by the other versions, confirming that our model leverages addtional information through the noise estimation. That being said, its performance remains very competitive. Another observation is that all models with noise maps perform very similarly to one another. Interestingly, the model with a single noise map per frame, which incidentally has the smallest number of parameters, performs better on synthetic data than the models with more parameters: when dealing with synthetic data, the information contained in a single noise map is sufficient for the network to have a good estimation of the noise level as the noise behaves identically for each channel. The opposite occurs on real raw data at high ISO values: when dealing with real data, where the noise does not strictly follow the heteroscedastic Gaussian model, having a larger number of parameters might slightly compensate for the domain gap between the

[7] All models discussed outside of this section do use per channel, per frame noise maps.

theoretical noise model and the actual noise samples.

2.4.3 Training on single-channel vs stacked Bayer patches

Our raw denoising models have Bayer images or patches as input. For computational efficiency and to adopt a similar strategy to the original sRGB FastDVDnet, we have up to this point fed the network Bayer images as stacked 4-channel tensors. However, because of the geometrical structure of the Bayer Color Filter Array, two pixel values at the same spatial location of two different stacked channels will pertain to different locations in the original Bayer raw image and the underlying scene.
Instead of stacking channels, we could also consider Bayer patches or images directly as single channel images, and train our raw model using these kinds of images as input. If we only change how the input is structured and keep the rest of the network architecture unchanged, it will have a smaller receptive field, as the first convolution kernel (if we maintain its 3×3 size) will effectively be applied to an area that is two times smaller in the original Bayer / scene space. Moreover, single channel Bayer images have their own set of peculiar properties, particularly in terms of dynamic range and pixel values, as intensity values of pixels corresponding to the same underlying Bayer channel would typically have higher correlation than other channels. This suggests that training a model on either arrangements of Bayer patches on the same underlying data would lead to very different kinds of kernels, which combined with different receptive fields might lead to nonidentical performance.
We thus set out to compare the performance of a RBDnet model trained on $H \times W \times 1$ patches versus $H/2 \times W/2 \times 4$ patches. All other training parameters are unchanged, and we only train the model on synthetic data for the purposes of this experiment (DAVIS+unprocessing).

Quantitative denoising performance of RBDnet trained on stacked versus single channel Bayer patches are available in Tables 2.7 and 2.8. The single channel model outperforms the stacked model regardless of ISO on synthetic data. When looking at real raw data of the CRVD dataset however, the stacked model outperforms the single channel one for ISOs 6400, 12800, and 2560, and the performance gap widens as ISO increases. The smaller receptive field hinders performance at higher noise levels, where it is effective to look at a spatially larger neighborhood of pixels, especially for a multi-image model where the input images are not explicitly aligned. As a sidenote, using Pytorch, the $H \times W \times 1$ model takes roughly 30% longer to train: while the first convolution kernel of the network has shallower depth (*i.e.* the model also has slightly fewer parameters), all convolutions must be performed on four times as many pixels.

Table 2.7: Denoising: PSNR with $H \times W \times 1$ patches versus $H/2 \times W/2 \times 4$ patches on unprocessed `gopro_540p`

ISO	1600	3200	6400	12800	25600	#params	MACs
noisy	39.47	36.49	33.58	30.58	27.58		
RBDnet ($H/2 \times W/2 \times 4$)	43.50	41.52	39.67	37.85	36.13	2.49M	338G
RBDnet ($H \times W \times 1$)	**43.90**	**41.90**	**40.03**	**38.19**	**36.46**	2.48M	1.31T

Table 2.8: Denoising: PSNR with $H \times W \times 1$ patches versus $H/2 \times W/2 \times 4$ patches on CRVD indoor test sequences (all models trained on DAVIS+unprocessing)

ISO	1600	3200	6400	12800	25600	#params	MACs
noisy	38.57	35.15	31.98	28.08	26.32		
RBDnet ($H/2 \times W/2 \times 4$)	47.10	45.26	**43.35**	**40.51**	**40.38**	2.49M	338G
RBDnet ($H \times W \times 1$)	**47.22**	**45.30**	43.29	39.89	40.08	2.48M	1.31T

2.4.4 Tone-curve and quantization aware dithering

The original sRGB images used by the models trained on synthetic are quantized to 8 bit. After unprocessing, the training patches have intensity histograms that are more sparse than typical real Bayer patches. We illustrate this in Figure 2.16 by comparing the 12 bit histograms of a CRVD Bayer image and the histogram of the same image processed, quantized to 8 bit, unprocessed then requantized to 12 bits.

These more sparse histograms might entail lower expressivity and dynamic range than using real data. We experiment with adding dithering to our synthetic raw generation pipeline to see if it can improve the denoising performance of our model. Simply adding uniform noise of amplitude 1/255 to our normalized 8 bit images before unprocessing is not effective, because different intensity values of the sRGB images are changed differently according to the mathematical operations of the pipeline. We isolate two deterministic, color-independent steps of the pipeline: the inversion of the tone mapping / smoothstep and the inversion of gamma compression (gamma expansion), which we call ζ^{-1} and Γ^{-1} respectively:

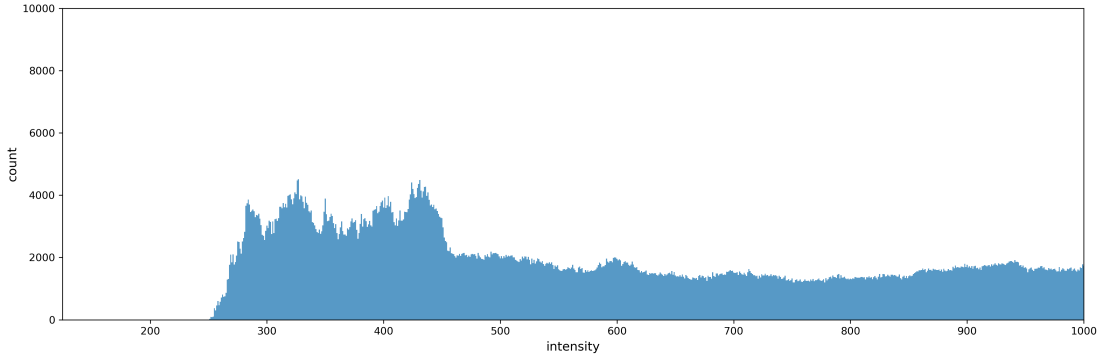
$$\zeta^{-1}(y) = \frac{1}{2} - \sin\left(\frac{\sin^{-1}(1-2y)}{3}\right) \tag{2.6}$$

$$\Gamma^{-1}(y) = \max(y, \epsilon)^{2.2} \tag{2.7}$$

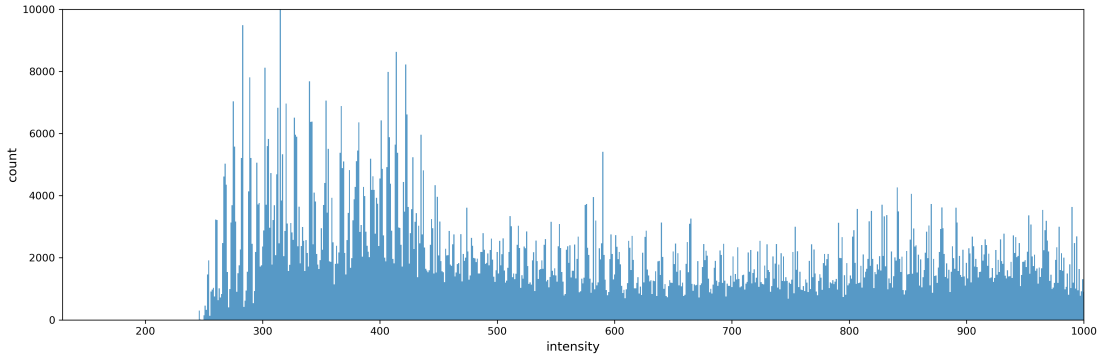
where $\epsilon = 10^{-8}$ prevents numerical instabilities. These unprocessing operations are the first to be applied to the 8 bit quantized sRGB image x_q:

$$y = \Gamma^{-1}(\zeta^{-1}(x_q)). \tag{2.8}$$

Processed image.



Raw image histogram.



Processed, quantized then unprocessed raw image histogram.

Figure 2.16: Comparison of the 12 bit histograms of frame 2 of CRVD indoor scene 9 (ISO 6400). Processing, quantization and unprocessing yield sparser histograms.

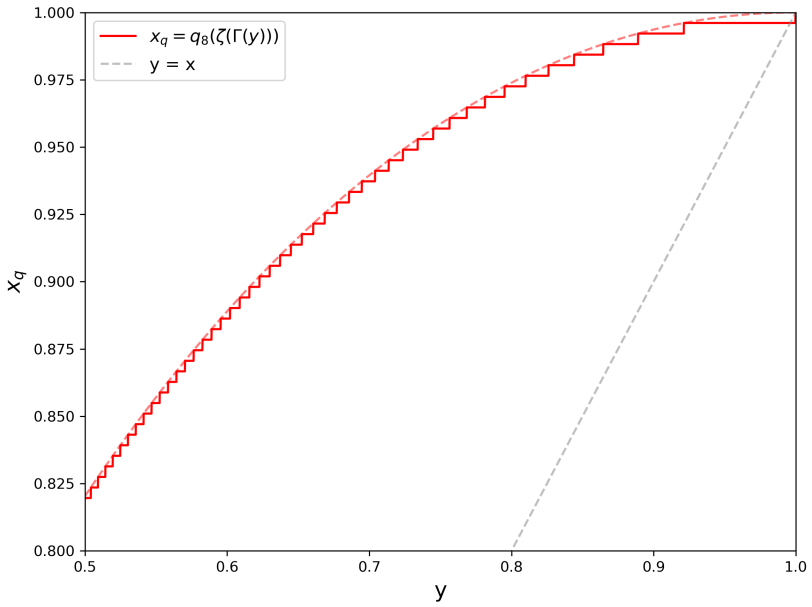


Figure 2.17: Effect of 8 bit quantization after gamma compression and tone mapping.

If we assume that all 8 bit sRGB images were obtained after gamma compression, tonemapping and quantization of some image y, we have

$$x_q = q_8(\zeta(\Gamma(y))) \tag{2.9}$$

where q_8 is the 8 bit quantization process. The main issue here is that because of the tone curve and quantization process, many values of y will map to a single value of x_q. This is illustrated in Figure 2.17.

In turn, when unprocessing, Equation 2.8 maps each possible value of x_q to a single value of y, instead of any value that would map to said value of x_q according to Equation 2.9. To mitigate this problem, we propose adding adaptive uniform noise that depends on the intensities of y after smoothstep inversion and gamma expansion. The uniform noise has a density d defined as

$$d(y) = \begin{cases} \frac{1}{\Gamma^{-1}(\zeta^{-1}(x_q+s_q))-\Gamma^{-1}(\zeta^{-1}(x_q))} & \text{for} \quad y \in \left[\Gamma^{-1}(\zeta^{-1}(x_q+s_q)), \Gamma^{-1}(\zeta^{-1}(x_q))\right[\\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

where s_q is the quantization step. We illustrate this "gamma-smooth aware dither" in Figure 2.18.

We assess the appeal of our dithering technique by comparing the performance of a RBDnet model trained with and without it. We also train a model with uniform 8 bit dithering noise added before unprocessing to showcase its lack of positive impact on denoising performance. Quantitative results on synthetic

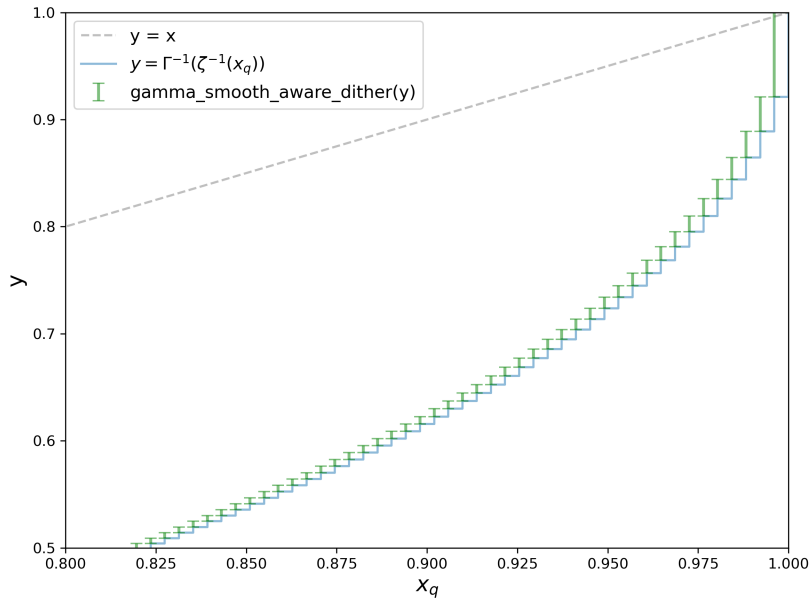


Figure 2.18: Gamma-smooth aware dithering. Intervals shown in green correspond to the range of possible values of y after dithering for a given 8 bit quantized value.

Table 2.9: Denoising: PSNR of RBDnet with and without dithering on unprocessed `gopro_540p` (no dithering added for testing)

ISO	1600	3200	6400	12800	25600
noisy	39.47	36.49	33.58	30.58	27.58
RBDnet (no dith.)	43.50	41.52	39.67	37.85	36.13
RBDnet (uniform dith.)	43.57	41.59	39.74	37.92	36.20
RBDnet (gamma-smooth aware dith.)	**43.66**	**41.65**	**39.79**	**37.96**	**36.23**

data (unprocessed `gopro_540p` with no dithering) and CRVD are available in Tables 2.9 and 2.10. Interestingly, both dithering techniques improve denoising performance on synthetic data, possibly because of the less sparse and thus more diversified training data. When denoising real raw data however, uniform dithering decreases performance when compared to our non-dithered baseline, while our gamma-smooth-aware dithering improves it, validating it as a practically cost-free technique to improve training on synthetic data created from 8 bit sRGB images. The performance gap is more significant at higher ISOs, which is expected because of the larger impact of quantization on high intensity values as shown in Figure 2.17.

From a visual standpoint, it seems that this dithering technique can help RBDnet produce smoother gradients of color when compared to the undithered version, as shown in Figure 2.19. We also observe that models trained with synthetic data and dithering do not present the artifacts shown in Figure 2.10. We illustrate a few examples of artifacts present in the baseline but not the gamma-smooth aware

Table 2.10: Denoising: PSNR of RBDnet with and without dithering on CRVD indoor test sequences (all models trained on DAVIS+unprocessing)

ISO	1600	3200	6400	12800	25600
noisy	38.57	35.15	31.98	28.08	26.32
RBDnet (no dith.)	47.10	45.26	43.35	40.51	40.38
RBDnet (uniform dith.)	47.05	45.09	43.16	40.22	40.32
RBDnet (gamma-smooth aware dith.)	**47.15**	**45.32**	**43.45**	**40.65**	**40.61**

noisy | RBDnet (DAVIS, no dithering) | RBDnet (DAVIS, gamma-smooth aware dithering)

Figure 2.19: CRVD outdoor, frame 2 of sequence 7, ISO 25600: our gamma-smooth aware dithering produces a smooth gradient between two light sources of different color, while the model without dithering produces a more quantized result (360×360 crop shown).

RBDnet (DAVIS, 5×4 maps, no dithering)

RBDnet (DAVIS, 5×4 maps, gamma-smooth aware dithering)

Figure 2.20: Dithering prevents the kinds of artifacts encountered in Figure 2.10 (the artifacts are slightly different because the RBDnet model was retrained from scratch on a different system, with a different seed).

dithered model in Figure 2.20; this probably accounts in part for the difference in PSNR.

2.4.5 Comparison to state-of-the-art methods

In this section, we compare our RBDnet models trained on synthetic and real raw data to two state-of-the-art multi-frame raw denoising methods: RViDeNet [Yue+20] and UDVD [She+21]. While both are initially described as raw video denoising methods, they take multiple raw images as input and return a denoised version of the center one, which is not very different from our own method.
RViDeNet was presented in [Yue+20] along with the CRVD dataset in 2020. It takes three images as input, treats Bayer channels separately, performs channel motion compensation using guided deformable convolutions, followed by feature refinement using spatial, channel and temporal attention, then temporal and spatial fusion to produce the denoised raw image (which can be turned into a sRGB image with a learned ISP). The architecture of the network is shown in Figure 2.21.

Presented in 2021, Unsupervised Deep Video Denoising [She+21] is effectively an unsupervised version of FastDVDnet. Inspired by Noise2Void [KBJ19], it creates a blind spot architecture by leveraging vertically causal convolutions and the creation of 4 rotated versions of the 5 input frames. In a nutshell, each rotated version of the 5 frames is separately fed to the network, and because of the vertically causal convolutions, the output only depends on the pixels above ($0\circ$ rotation), to the left ($90\circ$), below ($180\circ$) or to the right ($270\circ$) of the noisy input pixel

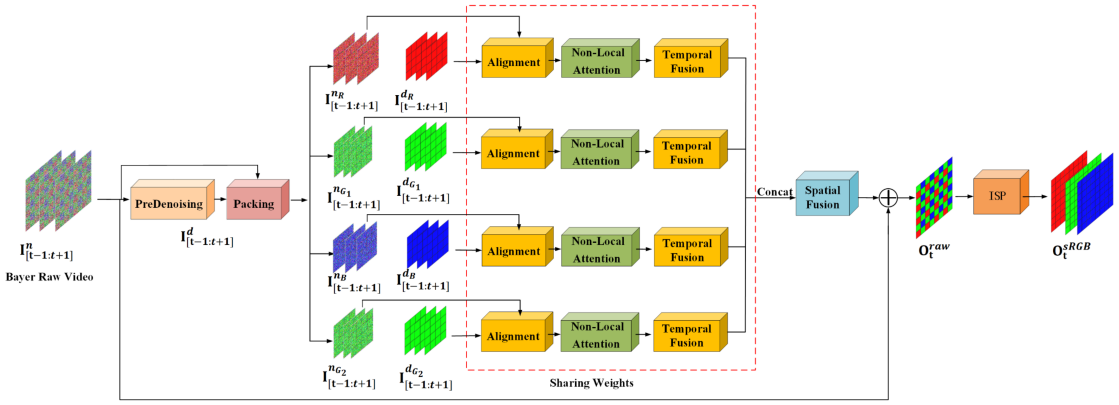


Figure 2.21: RViDeNet architecture (extracted from [Yue+20])

respectively. As in [KBJ19], early stopping is performed during training to avoid learning an identity mapping. The blind spot architecture of UDVD is illustrated in Figure 2.22.

For quantitative comparison of the burst denoising performance of RBDnet against UDVD and RViDeNet, we use CRVD yet again since it is the only available multi-frame raw dataset with real noisy images and clean ground truth. We use the networks off-the-shelf, with weights provided by the authors of the respective publications. Worth mentioning here is the data used to train the models:

- RViDeNet was trained using a supervised synthetic raw dataset which the authors call SRVD, then fine-tuned on the CRVD training set (indoor sequences 1 to 6). Since the author provide the weights of both the model trained on SRVD and the model trained on SRVD then fine-tuned on CRVD, we use both versions in our quantitative comparison.

- On the other hand, while UDVD is trained in an unsupervised fashion and thus never uses clean images, *it is trained on noisy images of the CRVD test set* (indoor sequences 7 to 11). We argue that this training procedure mentioned and used in [She+21] *for performance comparison on the very same test set* against RViDeNet (which did not see these sequences during training) is an unfair overfitting scenario, and are surprised that it was permitted in the review process of a nevertheless very interesting publication. A probably more fair comparison would require to retrain UDVD in an unsupervised manner on the CRVD training sequences, but we did not have the time to do that so we used the provided overfitted weights instead.

Table 2.11 showcases the performance of our two main RBDnet models against RViDeNet and UDVD. Unsurprisingly, UDVD performs the best since it has seen the same kind of images that the ones used for testing. It is also worth noting that RBDnet and RViDeNet have very similar performance: both models trained only on synthetic data perform roughly the same (*i.e.* well but slightly worse than the

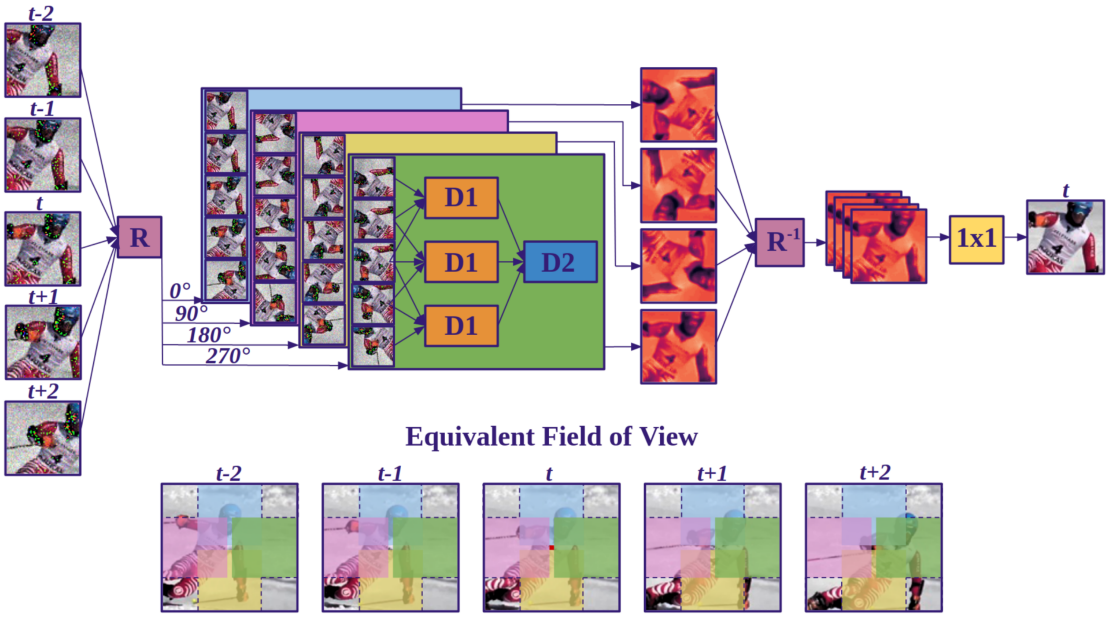


Figure 2.22: UDVD and its blind-spot architecture (extracted from [She+21])

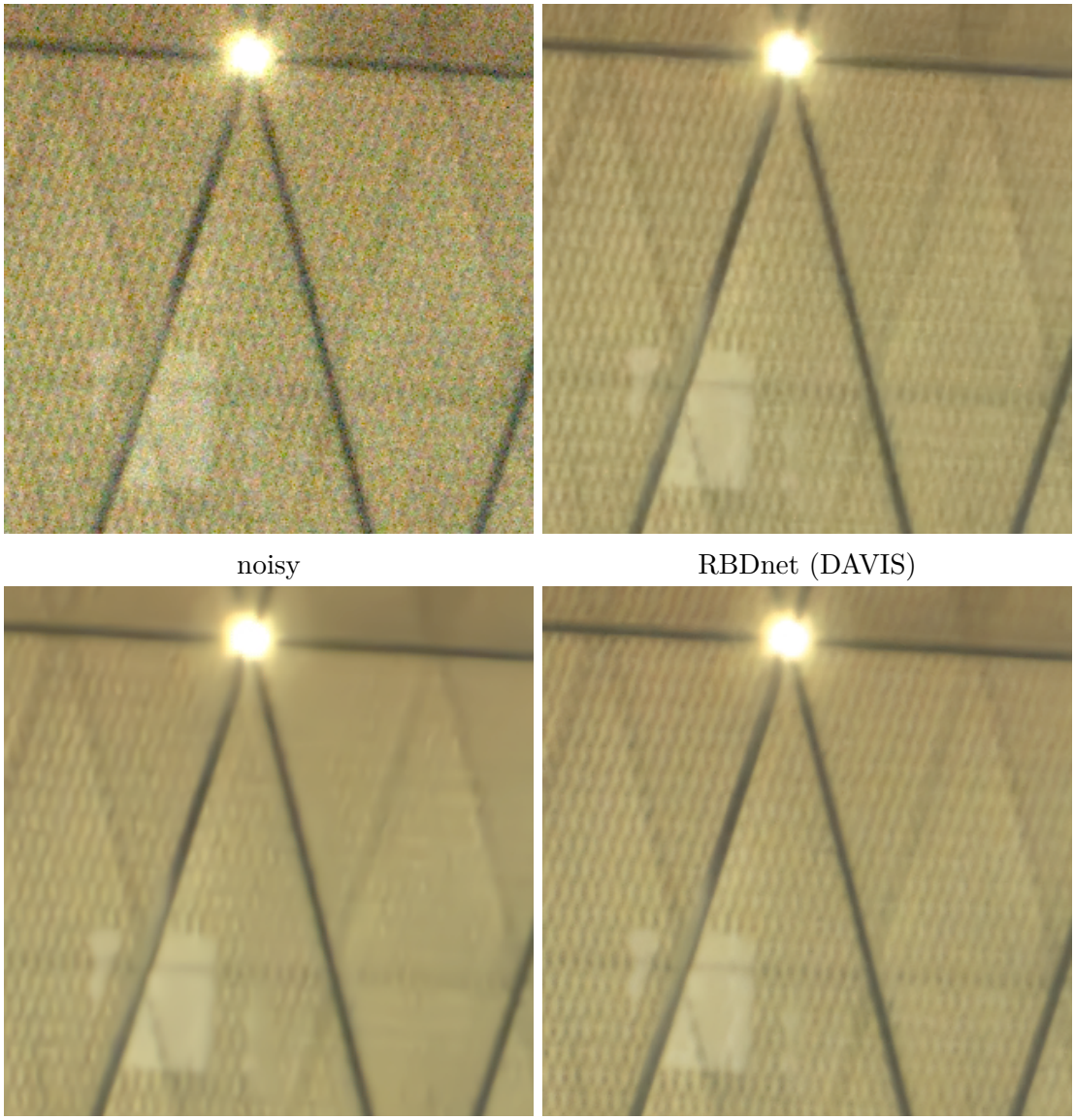
model trained on real data), and the performance gap between synthetic and real data is very similar for both models.

Also worth mentioning here is performance with respect to complexity, both in terms of architecture and computational cost. We report the number of learnable parameters and MACs in Table 2.11 as indicators of said complexity. While RViDeNet is disadvantaged against RBDnet by only taking 3 raw frames as input, it compensates by having around 3.5× more learnable parameters and a far more complex architecture. This translates to around 6.2× more operations than RBDnet to perform burst denoising of a 1920×1080 CRVD image with an almost identical PSNR. UDVD has around 14% less parameters than RBDnet: they have a very similar architecture, but UDVD inputs are stacked single channel Bayer images with no noise maps. However, because of how its blind spot receptive field is constructed, UDVD must perform four rotations and separate forwards through the network for each target pixel. This amounts to an explosion in complexity, with around 85× more MACs than RBDnet to burst denoise a CRVD image. In practice, forwarding a 5 frame 1920×1080 Bayer burst through UDVD yields an out of memory (OOM) error on a 40GB NVIDIA A100 GPU (meaning that we had to divide input images in patches and denoise each set of patches separately). Because of their evidently increased complexity compared to our model that retains the original simplicity and efficiency of the original FastDVDnet, UDVD and RViDeNet are not realistic targets for on-device inference in embedded systems of the first half of the 2020s, even as high end devices tend to feature more and more powerful dedicated hardware for neural network applications.

Table 2.11: Denoising: PSNR on CRVD indoor test sequences

ISO	1600	3200	6400	12800	25600	Avg.	# params	MACs
noisy	38.57	35.15	31.98	28.08	26.32	32.02		
RBDnet (DAVIS)	47.10	45.26	43.35	40.51	40.38	43.32	2.49M	338G
RBDnet (CRVD)	47.62	45.88	43.91	41.34	41.21	43.99	2.49M	338G
RViDeNet [Yue+20] (SRVD)	47.14	45.21	43.24	40.53	40.51	43.33	8.57M	2.1T
RViDeNet [Yue+20] (SRVD+CRVD)	47.73	45.87	43.79	41.15	41.10	43.93	8.57M	2.1T
UDVD [She+21]	**47.90**	**46.30**	**44.62**	**42.16**	**41.91**	**44.58**	2.13M	28.7T

A visual comparison of burst denoising of RBDnet, RViDeNet and UDVD on a CRVD outdoor sequence (which in this case UDVD did not see during training) is shown in Figure 2.23. Even though it was only trained on the indoor stop-motion sequences of the CRVD training set (using only noisy images to boot), UDVD produces the most pleasing results, which are certainly helped by the "test-time data augmentation" of feeding the network four rotated versions of the input images. RViDeNet tends to denoise well but often at the expense of high frequency image detail, which is similar to what we observed when comparing RBDnet trained on CRVD to RBDnet trained on unprocessed DAVIS in Section 2.3.2 (*c.f.* Figures 2.11, 2.12, 2.13 and 2.14). RBDnet trained only on synthetic data produces pleasing results at a reduced computational cost.



noisy RBDnet (DAVIS)

RViDeNet [Yue+20] (SRVD+CRVD) UDVD [She+21]

Figure 2.23: Burst denoising of frame 2 of CRVD outdoor sequence 4 ISO 25600. RBDnet (DAVIS) and UDVD are better at preserving the grid pattern than RViDeNet.

2.5 Summary

In this Chapter, we have studied the raw burst denoising problem with a deep learning approach. The 5-to-1 frame architecture of FastDVDnet originally designed for Gaussian video denoising can be turned into RBDnet, a powerful raw burst denoising model, with very little changes in the network design. Bigger changes must be made regarding training data. We experimented with two main approaches: *(i)* generate synthetic noisy multi-image raw data from sRGB videos using a sRGB to raw unprocesssing pipeline [Bro+19], or *(ii)* use a dataset of real noisy raw sequences and their corresponding noise-free counterparts [Yue+20]. Both approaches have their strengths and weaknesses. Large amounts of paired synthetic data are easy to obtain, and our model can be trained on virtually any possible noise level, but the noise model and synthetic raw images are approximations of the real thing and thus necessarily imperfect. On the other hand, capturing paired real raw data is a very costly process that limits the variety of training images in terms of scene and camera motion, illumination, or available noise parameters. Objectively comparing the two approaches is not an easy endeavour: quantitative evaluation will favor models with the smallest domain gap, but test data suffers from the exact same limitations as training data. In 2022, a new dataset named PVDD [Xu+22] was proposed as an answer to the lack of diversity of CRVD [Yue+20], but it has its own set of shortcomings, the biggest one being that noisy raw images are obtained by adding synthetic noise on top of raw images deemed sufficiently clean. The imperfect nature of quantitative evaluation for the real raw burst denoising problem is confirmed by the fact that in our experiments, the best model quantitatively does not necessarily equate the best model qualitatively. When visualizing the ouput of RBDnet on real data captured in the wild, the version trained on synthetic data produces sharper results than the model trained on real data only.

In terms of opportunities for future work, it would be interesting to see the robustness of our models (and their single-frame or blind variants) to noise levels or even noise types not seen during training. In [Moh+20], CNNs with bias-free convolutions and no batch normalization better generalize to unseen noise levels, but these experiments are conducted with AWGN and sRGB images so the findings might not necessarily translate to real raw denoising. Another possible avenue is the use of more complex noise models than the heteroscedastic Gaussian one in our synthetic models, *e.g.* incorporating row noise and quantization noise as in [Wei+20], or sampling from real dark frames (images where the sensor is completely shielded from incident light that only contain camera-specific noise) as in [Zha+21b]. One could also experiment with adding an extra Bayer to RGB demosaicking step to our model; similar experiments are suggested in [Dew+23] and its supplementary material[8], where two approaches are proposed: perform demosaicking before feeding the RGB output to FastDVDnet, or modifiy the ar-

[8] https://openaccess.thecvf.com/content/WACV2023/supplemental/Dewil_Video_Joint_Denoising_WACV_2023_supplemental.pdf

chitecture to include upsampling operations. Assessing real multi-image joint demosaicking and denoising quality also has its own set of challenges, because only synthetic data can be used to evaluate demosaicking performance.

Part II

Plug-and-Play Video Restoration

Chapter 3

Video Restoration with a Deep Plug-and-Play Prior

In this Chapter, we explore the use of video denoisers as regularizers within a PnP scheme and how they compare to the use of image denoisers on each frame separately. Section 3.2 describes our PnP method and how it integrates image and video denoisers. In Section 3.3, we perform an extensive experimental evaluation of these algorithms on video deblurring, super-resolution and interpolation of random missing pixels, as well as comparisons to state-of-the-art PnP and deep learning methods. Concluding remarks and opportunities for future research are presented in Section 3.4.

3.1 Background and state of the art

3.1.1 Image restoration as an inverse problem.

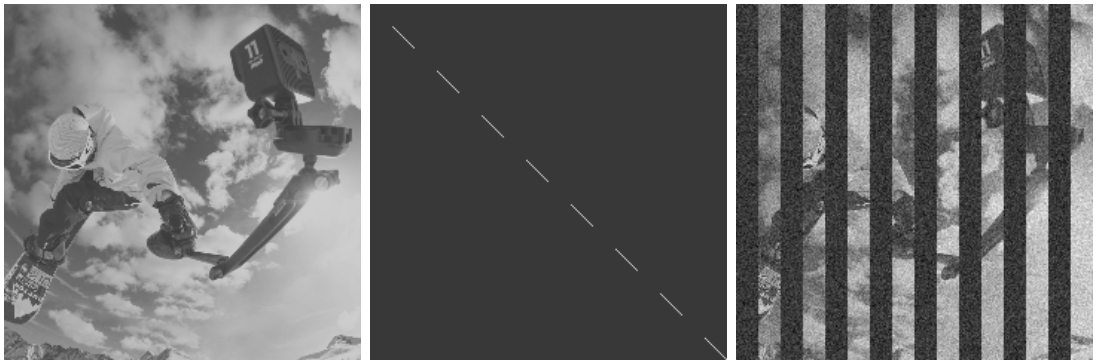
Many image restoration problems can be seen as *inverse problems*. In this context, image degradation is described through a direct form, or forward / observation model

$$y = \mathcal{A}(x) + n \tag{3.1}$$

where $y \in \mathbb{R}^m$ is the degraded *observation*, $x \in \mathbb{R}^d$ is the *unknown* image to be recovered, $\mathcal{A}$ is a linear or non-linear *degradation* operator, and $n \in \mathbb{R}^m$ is a realization from a known *noise* distribution.

As such, the goal of an inverse problem is to recover or *estimate* x *from the observation* y. These problems are often *ill-posed*: different values of x can yield the same y after degradation, making the solution not necessarily unique.

Inverse problems are also often *ill-conditioned*: the degradation $\mathcal{A}$ might be non-trivially invertible and/or low rank, and some estimators might be very sensitive to the input data or to the additional of the noise n (making *e.g.* the estimator $\widehat{x} = \mathcal{A}^{-1}(y)$ a very poor choice). It is necessary to provide extra information about x to make the problem well-posed in the Hadamard sense: a solution exists,



x A $y = Ax + n$

(a) Masking.

x A $y = Ax + n$

(b) Blurring.

Figure 3.1: Examples of image degradation with linear operators one might encounter in inverse problems ($n \sim \mathcal{N}(0, \sigma^2 \,\mathrm{Id})$, $\sigma = 0.1$).

is unique, and depends in a Lipschitz continuous manner w.r.t. y [Lau+23].

One common way to include said additional information in order to build reliable and robust estimators of x from y is to use a Bayesian formalism. In it, one assumes that the unknown x follows a law of density $p(x)$, called the *prior*. Using Bayes' rule, the marginal $p(x)$ can be combined with $p(y|x)$, the *likelihood of the observation* y given x (derived from the degradation model (3.1)) to obtain the *posterior* density $p(x|y)$:

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)}. \tag{3.2}$$

Multiple estimators can be derived from this distribution and selected as solutions of the inverse problem. One of them is the Minimum Mean Squared Error (MMSE):

$$\hat{x}_{\text{MMSE}} = \mathbb{E}_{x \sim p(x|y)}\left[x\right]. \tag{3.3}$$

Computing the MMSE is typically non-trivial and computationally expensive, as it requires computing an expectation w.r.t. $x \mid y$. It is more common in the image restoration literature to compute another estimator from the posterior distribution, the Maximum A Posteriori (MAP):

$$\hat{x}_{\text{MAP}} = \underset{x}{\text{argmax}} p(x \mid y) = \underset{x}{\text{argmax}} \log p(y \mid x) + \log p(x). \tag{3.4}$$

If n follows a Gaussian distribution $\mathcal{N}(0, \sigma_n^2 \,\text{Id})$ [1], the likelihood of y given x can be written as

$$p(y|x) = \exp\left(-\frac{\|\mathcal{A}(x) - y\|_2^2}{2\sigma_n^2}\right), \tag{3.5}$$

and estimating the MAP can be rewritten as the minimization problem

$$\hat{x} = \underset{x}{\text{argmin}} \frac{1}{2\sigma_n^2}\|y - \mathcal{A}(x)\|_2^2 + \alpha\mathcal{R}(x), \tag{3.6}$$

where $\alpha\mathcal{R}(x) = -\log p(x)$. If we speak in terms of potentials, notice that this is actually equivalent to the variational problem

$$\hat{x} = \underset{x}{\text{argmin}} \mathcal{F}(x) + \alpha\mathcal{R}(x) \tag{3.7}$$

where $\mathcal{F}$ is a least-squares *data fidelity* term that depends on the observation model, and $\mathcal{R}$ is a *regularization* term that encapsulates *a priori* information on the image.

[1] The distribution of noise observed in real images is typically not uniquely Gaussian [Foi+08; Wei+20]. However, this model is ubiquitous in the image processing literature, and it features mathematical properties that are essential to the theory presented in this Chapter.

3.1.2 Model-based methods and explicit image priors.

For a long time, Bayesian and variational restoration has relied on explicit priors (*e.g.* total variation [ROF92; LM13; DMP18] or Tikhonov regularization [EHN96]), expressing regularity assumptions on x either in the image space [KTF11] or in transformed spaces (wavelet transforms [DJ95], patch spaces [ZW11; DH18], etc.). When $\mathcal{R}$ is known and convex, there are numerous efficient numerical schemes to find the solutions of (3.6) [**chambolle2016introduction**]. These include gradient-based methods [BT09], as well as newer convex optimization algorithms introduced in the early 2010s that make use of the *proximal operator* [CP11],

$$\underset{f}{\text{prox}}(x) = \underset{z}{\text{argmin}} \frac{1}{2}||x - z||_2^2 + f(z). \tag{3.8}$$

Proximal splitting algorithms typically operate sequentially on $\mathcal{F}$ and $\mathcal{R}$ through their proximal operator.

3.1.3 Learning-based methods and end-to-end neural networks.

Over the past decade, a great success has been encountered by learning-based methods that circumvent the explicit modeling and solving of Equation 3.6 by directly applying a parametrized mapping to the observed data y that yields a solution $\hat{x}$:

$$\hat{x} = N_{\theta^\star}(y). \tag{3.9}$$

More specifically, in the case of end-to-end neural networks, N is typically a succession of learnable linear weights and non-linear activation functions, and the optimal values of the parameters $\theta^\star$ are obtained by the minimization of an empirical risk $\mathcal{L}$ over some training set[2]:

$$\theta^\star = \underset{\theta}{\text{argmin}} \sum_i \mathcal{L}\left(N_\theta\left(y_i\right), x_i\right) \tag{3.10}$$

where the set of training samples (x_i, y_i) can be constructed with an explicit degradation model (Equation (3.1)) [Zha+17a; Bro+19] or by the acquistion of real images [Che+18; Rim+20]. Solving Equation 3.10 is typically done through a large scale optimization problem. With the rise of deep and convolutional neural networks, learning-based methods have outperformed traditional model-based methods for most image and video restoration problems (see [ZZZ18; Bro+19; TDV20; Zam+21] for examples of efficient image or video restoration networks). Training these networks requires large amounts of data and computing resources (and these, along with network sizes, tend to keep increasing over the years). Moreover, a network trained for a given degradation model must be retrained as soon as the degradation model or its parameters change.

[2] Training an end-to-end image restoration neural network with a quadratic loss is an approximation of an underlying (and unknown) MMSE estimator [Lau+23].

3.1.4 Plug-and-Play methods and Regularization by Denoising.

During the same decade that saw the exponential growth of interest in neural networks, some methods have tried to bridge the gap between model-based and learning-based approaches, arguing that explicitly defined regularity assumptions might be insufficient to capture the actual distribution of complex, natural and clean images $p(x)$. These methods combine a likelihood defined explicitly according to the direct model (3.1), and a prior $\mathcal{R}$ implicitly defined by an efficient denoising algorithm. This combination is done algorithmically, typically within an alternate optimization scheme that splits the data fidelity and regularization subproblems.

Plug-and-Play (PnP) methods can be traced back to the seminal 2013 publication by Venkatakrishnan *et al.* [VBW13] that introduces a PnP version of Alternating Direction Method of Multipliers (ADMM) [Boy+11] where the proximal operator of $\mathcal{R}$ is replaced by a denoiser. A similar approach was featured a year later in [Hei+14] where a Gaussian denoising algorithm is used as the proximal operator of the regularization for multiple image restoration inverse problems.

Regularization by Denoising (RED) is another framework where the gradient of the regularization is explicitly defined by a denoiser. It proposes a Laplacian regularization term $\mathcal{R}$ proportional to the inner product between an image and its denoising residual:

$$\mathcal{R}(x) = \frac{1}{2} x^T (x - D(x)). \tag{3.11}$$

The authors of [REM17] stated that if the denoiser D is locally homogeneous and nonexpansive, the gradient of the regularization has the simple form $\nabla \mathcal{R} = x - D(x)$, and it can easily be plugged in gradient-based optimization schemes.

PnP and RED formulations of many optimization algorithms have been proposed, including (but not limited to) ADMM [VBW13; REM17; Ryu+19], Stochastic Gradient Descent [Lau+23], Primal-Dual [Mei+17; Hei+14], Iterative Shrinkage Thresholding (ISTA) [GC20; Xu+20], fast-ISTA [KMW17], Half Quadratic Splitting [Zha+21a] or Forward-Backward Splitting [Ryu+19; Pes+21; HLP22b]. All of these methods allow repurposing a single denoising algorithm for many restoration tasks such as super-resolution [BRE16; Zha+21a], denoising [RGE16; Buz+18], or image deblurring [WC17; Mei+17]. The Plug-and-Play operators framework has also been shown to be very efficient with Approximate Message Passing (AMP) algorithms [DMM09; Ahm+20], particularly for applications involving randomised forward operators, where it is possible to characterise AMP schemes in detail (see, e.g., [BM11; JM13]). The restriction on the forward operator does not hold for the inverse problems considered in this Chapter where we focus instead on classical optimization schemes such as the ones described above.

A large diversity of denoisers have been used in these frameworks. In early works, BM3D has been used the most [Hei+14; Dar+16; KMW17]. Many methods have then made the switch to deep denoising neural networks [Mei+17; Ryu+19;

Lau+23]. In [Zha+21a], an analysis of the efficiency of the different deep denoisers for multiple PnP image restoration tasks is provided. Recently, some works have suggested using denoising networks specifically designed to provide additional convergence guarantees of their PnP/RED method [Ryu+19; Coh+21; HLP22a; Pes+21; HLP22b].

3.1.5 Theoretical and practical convergence of PnP methods

Theoretical aspects of PnP/RED methods have been a very active field of research. Developing the theory has many benefits: providing additional use cases where the methods work; improving the stability and final performance of the output; relying less on hyperparameter tuning or have a better understanding of its impact; better grasp the required properties of the denoisers; and so on. First convergence proofs of PnP algorithms in the literature assumed strong properties on the denoiser, like boundedness or non-expansiveness and being the sub-gradient of a convex function [CWE16; Sre+16]. In [Ryu+19], a fixed-point convergence proof is provided for PnP-ADMM[3] and PnP-DRS using a deep CNN denoiser, if the residual of said denoiser $R_\sigma = \mathrm{Id} - D_\sigma$ is Lipschitz-continuous. In practice, they train a residual denoising network R to be 1-Lipschitz[4] by performing spectral normalization of each convolution layer of the network, through a power iteration at each training step, as done in [Miy+18]. Each layer has then spectral norm < 1, and spectral norm of the whole network is the product of the norm of all layers, so is tends to be significantly smaller than 1. This hard constraint limits the type of possible neural network architectures: Ryu *et al.* use a straightforward DnCNN architecture with convolution layers and ReLU activation functions. The proof of convergence, however, holds only if $\mathcal{F}(x) = \frac{1}{2\sigma_n^2}||\mathcal{A}x - y||^2$ is strongly convex, which is not the case in inverse problems such as super-resolution or pixel inpainting.

Rather than convergence to a fixed point of the optimization algorithm, other works have discussed the more appealing convergence to a minimum of the global objective function $\mathcal{F} + \alpha\mathcal{R}$. In [REM17], if the denoiser D is locally homogeneous and of 1-Lipschitz gradient (*i.e.* it has a symmetric Jabocian), the explicit regularization term is a convex function, and RED algorithms are guaranteed to converge to a global optimum solution if the data fidelity term is also convex. Several popular denoisers are analyzed in [REM17], however the required assumptions are only verified empirically, when in fact they typically do not have a symmetric Jacobian.

Convergence to a global optimum is also discussed in [CEM21] where regularization is performed through the fixed points of a denoiser: if a denoiser is presented with a clean image, it should do nothing, *i.e.* clean images should be

[3] PnP-ADMM and PnP-FBS share the same fixed points (Remark 3.1 of [Mei+17] and Proposition 3 of [SWK19])

[4] In [Ryu+19], the Lipschitzness of the residual is only empirically verified by tracking the expansiveness between iterates $||R_\sigma(x_{n+1}) - R_\sigma(x_n)||/||x_{n+1} - x_n||$ on a limited number of examples.

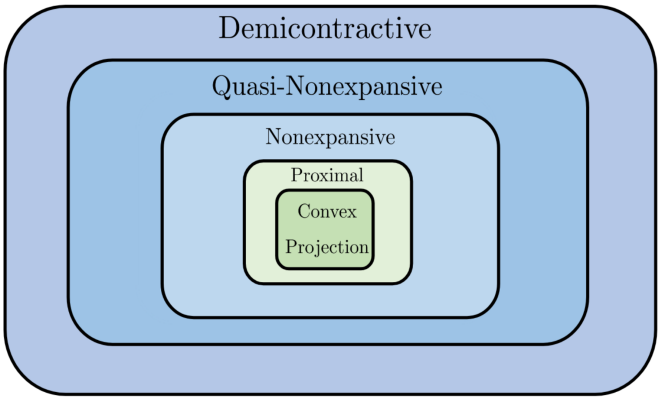


Figure 3.2: Demicontractive operators and their subclasses (extracted from [CEM21])

fixed points of D. RED [REM17] is thus reformulated as the convex optimization problem

$$\min_x \frac{1}{2}||Ax - y||_2^2 \quad \text{s.t.} \quad x \in Fix(D) \tag{3.12}$$

which is then solved through projected gradient descent. Their method, called RED-PRO, is guaranteed to converge to a solution of 3.12 if the denoiser D is demi-contractive, which is a weaker constraint than being a nonexpansive or a proximal mapping as illustrated in Figure 3.2. However, many denoisers used in PnP schemes have a limited fixed point set, making the condition $x \in Fix(D)$ unattainable in some cases. As such, Cohen *et al.* introduce a relaxed version of their method, where the solution needs only to be in a neighborhood of a fixed point[5].

Many convergence proofs of the literature assume that the denoiser is an MMSE or MAP estimator. Since such estimators are in the form of gradients, they must exhibit symmetric Jacobians. However, many denoisers used in the PnP/RED literature typically do not have a symmetric Jacobian. This problem is addressed in [Coh+21], where they enforce the condition by training a potential-driven denoiser based on DnCNN. This denoiser, which they call GraDnCNN, is the gradient of an explicit potential parametrized by a neural network. At its core lies a network $\mathcal{R}_\sigma$ whose output is a scalar, trained at multiple noise levels, including $\sigma = 0$. The network has non-negative weights and uses convex activation functions, making it a convex potential. The potential-driven denoiser is then defined as

$$D_\sigma(x) = x - \nabla \mathcal{R}_\sigma(x) \tag{3.13}$$

and using this convex potential in gradient-based optimization methods yields convergence to global minima if $\mathcal{F}$ is convex.

[5] [CEM21] also bridges some of the gaps between RED and PnP frameworks by showing that PnP-FBS and RED using accelerated gradient descent are a special case of RED-PRO.

A very similar approach is actually featured in [HLP22a]. The authors suggest training a denoising operator D_σ explicitly as a gradient step of the regularization term $\mathcal{R} : \mathbb{R}^n \to \mathbb{R}$:

$$D_\sigma(x) = x - \nabla \mathcal{R}_\sigma(x) = N_\sigma(x) + J_{N_\sigma}(x)^T (x - N_\sigma(x)) \tag{3.14}$$

where $\mathcal{R}_\sigma(x)$ is explicitly defined as $\frac{1}{2} \|x - N_\sigma(x)\|^2$ (notice that it yields a scalar as in [Coh+21]), and N_σ is parametrized by a differentiable deep neural network (which allows computing the Jacobian $J_{N_\sigma}(x)$ through automatic differentiation). Using such a denoiser in GSPnP, a relaxed version PnP-FBS, allows convergence to the minimum of the explicit functional $\mathcal{F} + \alpha\mathcal{R}$. Said convergence stills requires $\nabla \mathcal{R}_\sigma$ to be L-Lipschitz, but in practice this is controlled indirectly using a backtracking procedure that updates the step size so that it yields sufficient decrease property of the optimization algorithm.

[HLP22b] follows [HLP22a] by providing additional convergence guarantees in settings where the data fidelity term is not strongly convex. In this work, their denoiser explicitly defined as a gradient step is also rewritten as the proximal operator of a nonconvex function. $\nabla \mathcal{R}_\sigma$ still needs to be 1-Lipschitz for PnP-FBS and PnP-DRS to converge to a stationary point of an explicit functional. To do so, Hurault *et al.* fine-tune their gradient step / prox denoiser with a loss where the reconstruction term is augmented a penalty term enforcing the Jacobian of the residual $\mathrm{Id} - D_\sigma = \nabla \mathcal{R}_\sigma$ to have spectral norm < 1:

$$\begin{aligned} \mathcal{L}_S(\sigma) = \mathbb{E}_{x\sim p, \xi_\sigma \sim \mathcal{N}(0,\sigma^2)} \Big[\|D_\sigma(x + \xi_\sigma) - x\|^2 \\ + \mu \max\left(\left\| J_{(\mathrm{Id} - D_\sigma)}(x + \xi_\sigma) \right\|_S, 1 - \epsilon \right) \Big] \end{aligned} \tag{3.15}$$

where λ is a parameter controlling the tradeoff between reconstruction quality and non-expansiveness of $\mathrm{Id} - D_\sigma$, and $\varepsilon > 0$. The spectral norm is approximated using power iterations.

This soft constraint is direclty inspired by [Pes+21], where convergence properties arise from standard tools of monotone operatory. In that publication, a denoising neural network D is used as an approximation of the resolvant of a stationary maximally monotone operator A. Said operator A is maximally monotone if and only if there exists a 1-Lipschitz operator $Q = 2D - \mathrm{Id}$. If Q is non-expansive, D is firmly non-expansive, A is maximally monotone, and PnP-FBS is guaranteed to converge. In practice, the non-expansiveness of Q is obtained through a soft constraint during the training of D (change Equation 3.15 by replacing $J_{(\mathrm{Id} - D_\sigma)}$ with $J_{(2D_\sigma - \mathrm{Id})}$).

3.1.6 Extensions of PnP methods

Deep Unfolding / Unrolling Networks.

More recently, deep unfolding networks (DUNs) [Dia+18] have been proposed for specific image restoration tasks, such as super-resolution [ZVT20], or joint de-

blurring, demosaicking and denoising [ESP21]. Just like PnP methods, these approaches (like unrolling algorithms in general [MLE21]) incorporate some advantages of both learning-based and model-based methods. Compared to standard learning-based methods, unrolled algorithms can be trained correctly on much smaller datasets [GOW19]. Compared to PnP methods, DUNs usually yield better results in fewer iterations, as they translate the truncated unfolded optimization into an end-to-end training of a deep network. An additional advantage of the latter is that the manual setting of optimization hyperparameters can be avoided in the unfolded scheme. In contrast, PnP methods remain more flexible and versatile, as DUNs need a separate training for each restoration task. The concepts of optimization theory being common to both DUNs and PnP methods, innovations in one field can impact the other.

Laroche *et al.* tackle the topic of super-resolution with non-uniform blur in [LAT22]: using a method such as PnP-ADMM for that problem would require the inversion of the forward operator to compute the proximal operator of the data fidelity term, which would in turn be very computationally heavy with non-uniform blur. As such, the authors of this paper introduce a linearized version of ADMM which substitutes the splitted variable z by a linear mapping and features a linear approximation of the regularization term of the augmented Lagrangian, which in the end allows avoiding the inversion of the blur matrix. Linearized-ADMM is used within a Deep Unfolding Network that yields impressive results in super-resolution with non-uniform blur.

PnP sampling.

In many image restoration inverse problems, the degradation operator $\mathcal{A}$ is poorly conditioned or rank deficient. One might then want to consider estimating the whole posterior distribution instead of a specific pointwise estimators such as MMSE or MAP. Sampling from the posterior can then be performed to compute said estimators, quantify uncertainties or generate multiple candidates for image restoration. In [Lau+22], both the observation y and the unknown x are seen through a Bayesian prism and thus considered random variables X and Y. A Plug-and-Play version of the Unadusted Langevin Algorithm is proposed:

$$X_{k+1} = X_k - \delta \nabla H(X_k) + \sqrt{2\delta} Z_{k+1} \tag{3.16}$$

where $(Z_k) \sim \mathcal{N}(0, \mathrm{Id})$ iid, $H = -\log p(y \mid x) - logp(x)$ and δ controls the tradeoff between asymptotic distribution accuracy and convergence speed. The algorithm is guaranteed to converge to a unique stationary distribution if ∇H is L-Lipschitz and $\delta < 1/L$. Notice the similarities and differences between ULA and Gradient Descent / Stochastic Gradient Descent, which are illustrated in Figure 3.3.

Combination of implicit and explicit priors.

While the denoisers used in standard PnP methods contain an implicit representation of the space of clean images $p(x)$, some applications might benefit from

$$X_{k+1} = X_k - \delta_k \nabla H(X_k) \quad \text{(GD)}$$
$$X_{k+1} = X_k - \delta_k \nabla H(X_k) + \delta_k Z_{k+1} \quad \text{(SGD)}$$
$$X_{k+1} = X_k - \delta \nabla H(X_k) + \sqrt{2\delta} Z_{k+1} \quad \text{(ULA)}$$

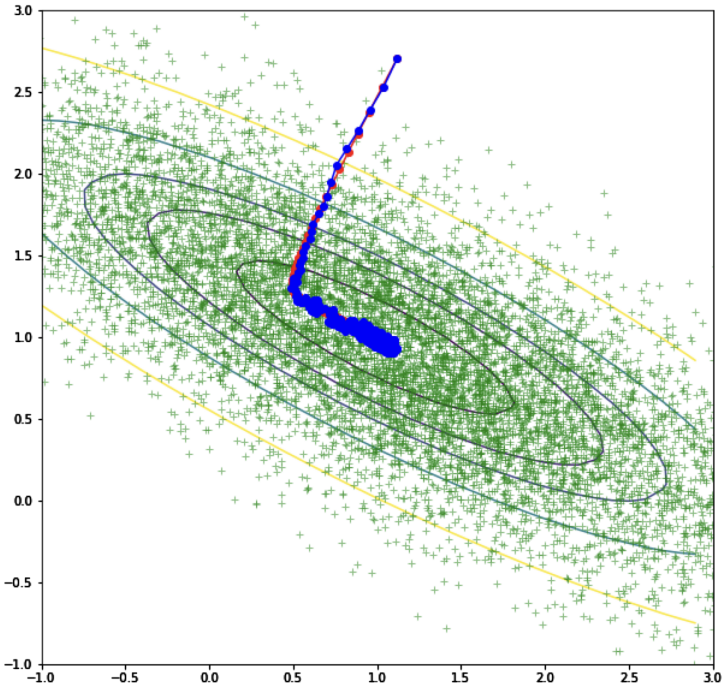


Figure 3.3: Illustration of the exploration properties of ULA compared to GD and SGD with $p_X(x) \propto e^{-H(x)}$.

additional explicit regularization to ill-conditioned inverse problems by including *e.g.* smoothness assumptions. In 2014, [Hei+14] proposed replacing the single regularization term of an optimization scheme such as ADMM or Primal-Dual by a weighted sum of three regularizers: a gradient sparsity prior, a denoising prior and a cross-channel correlation prior. The global proximal operator is then constructed by stacking the proximal operators of each prior. This technique is reused in [Mei+17], where the implicit image prior represented by the denoiser is combined to an explicit TV prior for image deblurring.

Image restoration priors beyond denoising networks.

While they veer away from the theoretical backbone of PnP and RED methods, some works have suggested substituting the denoisers by other types of restoration algorithms as implicit image priors, in order to be applied to new use cases or provide additional regularization beyond i.i.d. noise removal. In [Liu+20], a variant of RED where the denoiser is replaced by a general artifact removal network is proposed. They use a network based on DnCNN trained on heavily undersampled 4D MRI data (using only pairs of degraded images similarly to Noise2Noise [Leh+18]) as an example of implicit prior. In [EVM21], a residual U-Net is trained as a reconnective network that ensures continuity in images of blood vessels. Said network is then plugged alongside standard TV regularization in the Primal-Dual algorithm.

3.1.7 Connections to other related works

Denoisers as projections on a manifold of clean images.

In effect, the denoiser in PnP and RED schemes is used to iteratively "refine" the prediction of the data fidelity term, according to a certain image prior. But what does that refinement step entail, and what does that prior look like? The original RED paper [REM17] argues that noise-free images lie on a manifold $\mathcal{M}$, and that after contamination by additive noise, an image "pops out of the manifold along the normal to $\mathcal{M}$ with high probability". The reverse process of denoising would then be an orthogonal projection of an image onto $\mathcal{M}$. This idea is also explored in [Moh+20], where bias-free denoising networks are shown to perform interpretable adaptive linear filtering. Such networks project noisy images in a low-dimensional adaptive subspace, which can be visualized using singular value decomposition. The authors thus show that the denoiser represents an implicit prior that lies on a generalized cone manifold. In a followup work [KS21], a bias-free CNN trained using L2 loss is used in a Iterative Score Ascent scheme. This method can be seen as a gradient ascent with coarse-to-fine sampling onto the manifold of the prior that lies in the denoiser. In this work, they define the prior predictive density as

$$\tilde{x}(y) = y + \sigma^2 \nabla y \log p(y) \tag{3.17}$$

where $p(y)$ here is a blurred version of the prior on natural images $p(x)$: $p_\sigma(y) = g_\sigma(z) * p(x)$, $g_\sigma(z)$ being a Gaussian distribution of variance σ^2. Under additive

white Gaussian noise, there is equivalence to the prior predictive density and the MMSE [Miy+61]. The authors of [KS21] also propose a stochastic variant of their method by adding noise at each iteration. Using that method when starting from noise samples allows sampling from the image manifold; this method is also used for linear image restoration problems such as inpainting or super-resolution.

Image priors in self-supervised and unsupervised learning.

Since PnP / RED methods use an iterative process to map the degraded observation y to an estimate of the underlying clean image x, they can be seen as a special kind of self-supervised learning. The use of a denoiser (that is a neural networks in some works) as an implicit prior also tie them back to other works like Deep Image Prior [UVL18], where a degraded image and the handcrafted structure of a restoration neural network are sufficient to provide an estimate of the underlying clean image. Formally:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} E\left(N_\theta(z); x_0\right), \quad x^* = N_{\theta^*}(z) \tag{3.18}$$

where the network parameters θ are randomly initialized, and the minimizer is sought after using an optimization algorithm (a strategy of early stopping must be used to avoid learning an identity mapping). The combination of the information contained in degraded images to the implicit knowledge contained within denoisers can also remind of publications of the unsupervised and self-supervised denoising literature such as [Leh+18; BR19; KBJ19], which have shown competitive (if not slightly lesser) performance against denoisers trained with traditional supervised learning.

In the Deep PnP framework, denoising neural networks can be seen as implicitly encapsulating the distribution of clean images, or as projections towards a manifold. This bears similarity with other fields that make use of implicit representation through neural networks, such as neural fields. In the seminal work NeRF [Mil+20], a coordinate-based Multi-Layer Perceptron is used to map the coordinates of a point and viewing direction in 3d space to a color / radiance. The MLP encapsulates an implicit three-dimensional representation of the scene, and is only learned from a set of 2D images. Said implicit representation is gridless and continuous, and the network is connected to a fully differentiable render to synthesize novel views of the scene. Following this work, radiance fields and related implicit coordinate-based representations have been used extensively in many fields other than 3D view synthesis, including inverse problems such as Sparse-view Computed Tomography [Sun+21b], Intensity Diffraction Tomography [Liu+22] as well as general image and video restoration tasks such as spherical image super-resolution [Yoo+21], raw burst denoising [PTK22] or video frame interpoloation [Sha+22].

3.1.8 Video restoration with Plug-and-Play methods.

To the best of our knowledge, the work presented in this chapter is the first one describing the use of neural networks in a PnP method for video restoration problems. In [Yua+21], the video denoiser FastDVDnet [TDV20] is used in a PnP scheme, but to solve a specific problem of snapshot compressive imaging where the observation is a single frame. The work [KB21] uses the PnP-ADMM algorithm for video super-resolution, but employs the patch-based single-image denoiser BM3D [Dab+07].

3.2 A video Plug-and-Play method

We now present our method for PnP video restoration. To emphasize that we are now dealing with frame sequences instead of individual images, variables that represent videos will be noted in bold, *e.g.* $\boldsymbol{x}$. Our experiments in this chapter use the PnP version of ADMM because we have empirically found it to be the alternate optimization method that is the most stable and the least sensitive (but not insensitive) to hyperparameters for our use case. That said, many other schemes such as the ones described in Section 3.1 could also be used.

3.2.1 PnP-ADMM

Let us start by recalling the principle of the ADMM optimization method. Suppose that we want to solve our inverse problem of video restoration by minimizing Equation (3.6), which we rewrite here for clarity:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{1}{2\sigma_{\boldsymbol{n}}^2}\|\boldsymbol{y} - \mathcal{A}(\boldsymbol{x})\|_2^2 + \alpha\mathcal{R}(\boldsymbol{x}). \tag{3.6}$$

We start by defining the augmented Lagrangian

$$L_\varepsilon(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{v}) = \frac{1}{\alpha}\underbrace{\frac{1}{2\sigma_{\boldsymbol{n}}^2}\|\boldsymbol{y} - \mathcal{A}(\boldsymbol{x})\|_2^2}_{F(\boldsymbol{x},\boldsymbol{y})} + \mathcal{R}(\boldsymbol{z}) + \frac{1}{2\varepsilon}\|\boldsymbol{x} - \boldsymbol{z}\|_2^2 + \boldsymbol{v}^T(\boldsymbol{x} - \boldsymbol{z}) \tag{3.19}$$

that we wish to minimize in $(\boldsymbol{x}, \boldsymbol{z})$ and maximize in $\boldsymbol{v}$. In this formulation, $\boldsymbol{v}$ plays the role of a Lagrange multiplier, and this whole saddle-point problem is optimized by alternating descent steps in $\boldsymbol{x}$ and $\boldsymbol{z}$ with an update in $\boldsymbol{v}$, which yields the following scheme (setting $\boldsymbol{u} = \varepsilon\boldsymbol{v}$)

$$\begin{aligned}
\boldsymbol{x}_{k+1} &\leftarrow \underset{\boldsymbol{x}}{\operatorname{argmin}} L_\varepsilon(\boldsymbol{x}, \boldsymbol{z}_k, \boldsymbol{u}_k/\varepsilon) = \operatorname{prox}_{\frac{\varepsilon}{\alpha}F(\cdot,\boldsymbol{y})}(\boldsymbol{z}_k - \boldsymbol{u}_k) \\
\boldsymbol{z}_{k+1} &\leftarrow \underset{\boldsymbol{z}}{\operatorname{argmin}} L_\varepsilon(\boldsymbol{x}_{k+1}, \boldsymbol{z}, \boldsymbol{u}_k/\varepsilon) = \operatorname{prox}_{\varepsilon\mathcal{R}}(\boldsymbol{x}_{k+1} + \boldsymbol{u}_k) \\
\boldsymbol{u}_{k+1} &\leftarrow \boldsymbol{u}_k + \boldsymbol{x}_{k+1} - \boldsymbol{z}_{k+1},
\end{aligned} \tag{3.20}$$

where, if the degradation operator $\mathcal{A}$ is linear and represented by a matrix A,

$$\text{prox}_{\tau F(.,\boldsymbol{y})}(\boldsymbol{z}) = \left(\frac{\tau}{\sigma_n^2}A^*A + I_d\right)^{-1}\left(\frac{\tau}{\sigma_n^2}A^*\boldsymbol{y} + \boldsymbol{z}\right), \tag{3.21}$$

with $\tau \in \mathbb{R}^+$, I_d the identity matrix in dimension d and A^* the adjoint matrix of A. When the prior $\mathcal{R}$ is convex, the iterates $(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{v})$ of the algorithm satisfy $\boldsymbol{x} - \boldsymbol{z} \to \boldsymbol{0}$, and thus give us solutions of (3.6).

Consider that we know how to build a denoiser $\mathcal{D}_\varepsilon$ which can be expressed as the MAP estimator for a denoising problem (for an i.i.d. Gaussian noise of variance ε) with log-prior $\mathcal{R}$. By definition of the MAP, we then have exactly

$$\mathcal{D}_\varepsilon(\boldsymbol{x}) = \underset{\boldsymbol{z}}{\text{argmin}}\frac{1}{2\varepsilon}\|\boldsymbol{x} - \boldsymbol{z}\|_2^2 + \mathcal{R}(\boldsymbol{z}) = \text{prox}_{\varepsilon\mathcal{R}}(\boldsymbol{x}) \tag{3.22}$$

so we can directly "plug" this denoiser in the previous optimization scheme. In practice, Plug-and-Play schemes are successfully used even with denoisers that do not satisfy this property, and the study of their convergence is a very active research field [CWE16; Ryu+19; Xu+20; Lau+23; HLP22a; HLP22b]. The corresponding PnP-ADMM algorithm is summarized in Alg. 3.

Algorithm 3: PnP-ADMM scheme

Require: $\boldsymbol{x}_0 \in \mathbb{R}^d, \boldsymbol{y} \in \mathbb{R}^m, K \in \mathbb{N}^\star, \varepsilon > 0, \alpha > 0$
Initialization: Set $\boldsymbol{z}_0 = \boldsymbol{x}_0$, and $\boldsymbol{u}_k = \boldsymbol{0}$
for $k \in \{0, \dots, K-1\}$ **do**
  $\boldsymbol{x}_{k+1} \leftarrow \text{prox}_{\frac{\varepsilon}{\alpha}F(.,\boldsymbol{y})}(\boldsymbol{z}_k - \boldsymbol{u}_k)$
  $\boldsymbol{z}_{k+1} \leftarrow \mathcal{D}_\varepsilon(\boldsymbol{x}_{k+1} + \boldsymbol{u}_k)$
  $\boldsymbol{u}_{k+1} \leftarrow \boldsymbol{u}_k + (\boldsymbol{x}_{k+1} - \boldsymbol{z}_{k+1})$
return $\boldsymbol{x}_K$

3.2.2 Case of video

In the case of inverse problems on digital videos, we propose in this chapter to use the previous scheme directly on the whole video $\boldsymbol{x}$, which means that the proximal and denoising steps in the PnP-ADMM scheme are applied to the whole sequence at each step. This allows considering cases where the operator $\mathcal{A}$ cannot be written in a separable way over all images of the sequence (in the case of temporal blur, for example). This also allows the use of networks specifically designed for video denoising, such as FastDVDnet [TDV20].

Note that if one uses a single-frame denoiser (applied separately on each frame of the video), and if the degradation operator $\mathcal{A}$ is separable (*i.e.* can be written as a degradation on each frame of $\boldsymbol{x}$ separately), then the iterative ADMM scheme working on the whole video is equivalent to a succession of iterative ADMM schemes applied on each frame of the video. This is obviously no longer the case with video specific denoisers that are applied to the video in a non-separable way.

3.2.3 Performance of deep PnP Gaussian denoisers

Most of the experiments conducted in the next sections compare DRUNet [Zha+21a] and FastDVDnet [TDV20], two state-of-the-art networks for Gaussian denoising. DRUNet is a single-image network designed specifically to be integrated within a PnP approach for image restoration. FastDVDnet is a network designed for video denoising: it makes use of the additional information contained in neighboring images to provide a better and more temporally stable denoised estimate, with no explicit image alignment. Both networks use U-Net autoencoders [RFB15]: FastDVDnet combines two small U-Net blocks with residual connections and batchnorm in a cascaded architecture (*c.f.* Figure 2.6 of Chapter 2); DRUNet has a single deeper block with more subsampling steps (*c.f.* Figure 2.2) and replaces the standard convolution layers with ResBlocks [Lim+17].

We use these networks (and their weights) as provided by the authors of the original publications, without re-training them. Before studying the performance of these networks in PnP restoration, it is interesting to evaluate their denoising performance. It seems reasonable to think that the denoising performance of the network has an impact on the maximum achievable performance in PnP restoration [Zha+21a]. DRUNet denoises all the frames of the video separately. To produce a denoised version of the image at time t, FastDVDnet uses the images at times t_{-2}, t_{-1}, t, t_{+1} and t_{+2}. To ensure the output video retains the same size, the noisy input video is increased by 4 frames by mirroring: $\boldsymbol{x} = \{x_0, \dots, x_{N-1}\} \to \boldsymbol{x}' = \{x_2, x_1, x_0, \dots, x_{N-1}, x_{N-2}, x_{N-3}\}$. We include DnCNN [Zha+17a] and FFDNet [ZZZ18] to the comparison for reference; while they are not considered state-of-the-art denoisers in the early 2020s, they are both staples of the denoising literature and variants of DnCNN are still used in recent PnP/RED methods [Liu+20; Coh+21; Pes+21; Liu+21]. Also included to our comparison is GS-DRUNet [HLP22a], the denoising operator explicitly defined as a gradient step (*c.f.* equation (3.14); in [HLP22a] GS-DRUNet is used in GSPnP, their modified version of PnP-HQS, which we'll include in some of our PnP video restoration experiments). The range of noise levels seen in training is $\sigma \in [5/255, 55/255]$ for FastDVDnet and $\sigma \in [0, 50/255]$ for DRUNet, GS-DRUNet and FFDNet. DnCNN being trained at a fixed noise level of $\sigma = 25$, noisy frames where $\sigma_{\geq 25}$ are multiplied by a scale factor $(25/\sigma_{\geq 25})$ before being fed to the network. We evaluate the video denoising performance of the networks on the test set of the DAVIS-2017 dataset [Pon+17], in its 480p version. This dataset consists in 30 video sequences of varying length. None of the networks have seen these sequences in training.

Quantitative video denoising results of these networks can be seen in Table 3.1. FastDVDnet performs slightly better than DRUNet at most noise levels, even though it has around 13 times less parameters and takes about half the time to denoise one video frame. We also notice that GS-DRUNet remains very competitive with DRUNet for $\sigma_{\boldsymbol{n}} \leq 50/255$ despite featuring roughly 50% less parameters. DRUNet, however, is the network whose performance degrades the least at $\sigma_{\boldsymbol{n}} = 100/255$, a noise level not seen in training for all networks. Like FastDVD-

net, the convolution layers of DRUNet are without biases; but DRUNet has no batch-norm layers, while FastDVDnet does (with biases in them) and DRUNet was trained with L1 loss, while GS-DRUNet and FastDVDnet were trained with the L2 loss. These results are consistent with those of [Moh+20], where it was shown that unbiased L1 networks generalize better at noise levels outside the interval seen in training.

Table 3.1: Denoising: PSNR/SSIM on DAVIS-2017-test-480p [Pon+17] (8 CPU AMD 7F52 / 1 NVIDIA Tesla T4 / 16GB RAM)

$\sigma_{\boldsymbol{n}}$	10/255	25/255	50/255	100/255	# params	runtime (s/img.)
noisy	28.13/0.634	20.17/0.314	14.15/0.146	08.13/0.053		
DnCNN [Zha+17a]	35.66/0.946	31.45/0.884	28.21/0.796	25.05/0.664	0.668M	0.20
FFDNet [ZZZ18]	36.24/0.952	31.68/0.889	28.43/0.809	25.33/0.689	0.852M	0.08
DRUNet [Zha+21a]	38.90/0.967	34.40/0.921	31.27/0.861	**28.32/0.781**	32.6410M	0.48
GS-DRUNet [HLP22a]	38.81/0.967	34.30/0.919	31.11/0.856	24.39/0.495	17.010M	0.52
FastDVDnet [TDV20]	**39.20/0.969**	**35.05/0.931**	**31.97/0.878**	26.84/0.655	2.4791M	0.23

3.3 Experiments

We are now interested in the performance of our PnP-ADMM method using the networks of Section 3.2.3 for several video restoration problems such as non-blind deblurring, super-resolution, demosaicking, and interpolation of random missing pixels. These results aim at presenting the viability of our method, and at evaluating the impact of the chosen denoiser on the final result.

In the following experiments, we restrict each sequence to its first 30 frames to harmonize computation times and complexity per video.

3.3.1 Optimal setting of PnP parameters

For a given video restoration problem, our formulation of PnP-ADMM has several parameters that can have an impact on final performance:

- $\mathcal{D}_\varepsilon$, the Gaussian denoiser used as the proximal operator of the prior term.
- ε, which represents the strength of said denoiser. When the denoiser is designed to handle varying levels of noise in a non-blind fashion, (as is the case of FastDVDnet and DRUNet), ε can be set in the form of a noise map that is fed as an additional input to the denoiser.
- α, which represents the tradeoff between the data fidelity and prior terms of Equation (3.6).
- K, the total number of ADMM iterations. It needs to be large enough to allow convergence of the algorithm, but not too large in cases where convergence is not guaranteed.

For a given problem and a given denoiser (i.e. $\mathcal{A}$, $\sigma_{\boldsymbol{n}}$ and $\mathcal{D}$ are fixed), to find optimal values for said parameters, we perform a 2D grid search on ε and α for an empirically found sufficient number of iterations K. Since we have no guarantees that these values are optimal for every possible $\boldsymbol{x}$, we perform that grid search on a subset of the data we wish to evaluate our method on (in our case, 256×256 center crops of the first 30 frames of the `aerobatics`, `girl-dog`, `horsejump-stick` and `subway` of DAVIS-2017-test-480p [Pon+17]). We retain the (ε, α, K) tuple that yields the higher PSNR on that subset for a given denoiser. An example of a grid search of (ε, α) for a given video deblurring problem is illustrated in Figure 3.4. The parameter values obtained for the video restoration problems studied in this work are given in Tables 3.2, 3.3, 3.4 and 3.8.

3.3.2 Non-blind video deblurring

Our first use case is relatively straightforward: we generate blurry videos by convolving each frame by a known 2D kernel and by adding noise of standard deviation

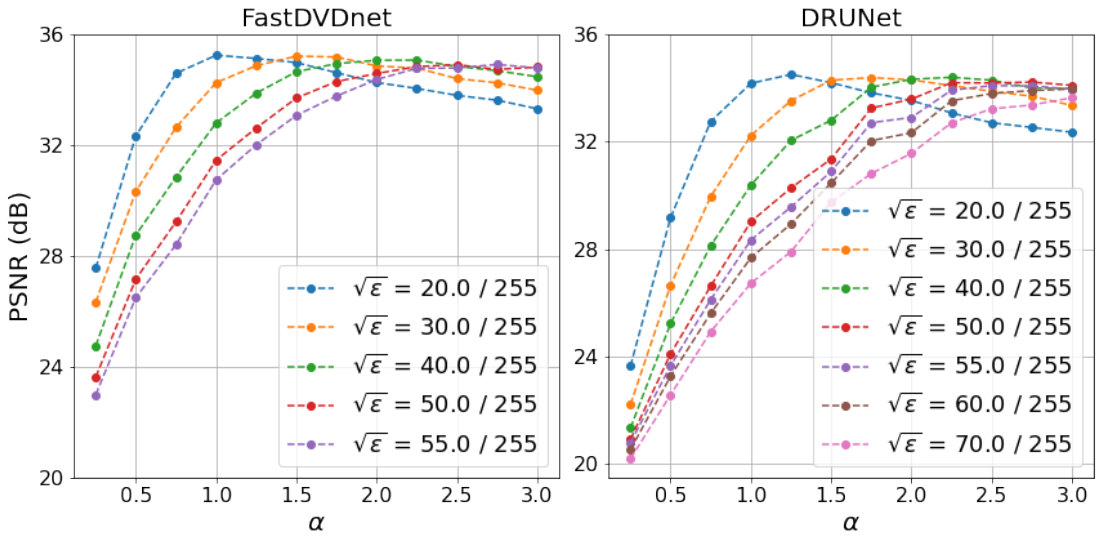


Figure 3.4: Grid search of optimal (ε, α) for video deblurring ($\sigma_{\boldsymbol{n}} = 2.55/255$)

Table 3.2: Video deblurring: selected values of (ε, α, K) PnP-ADMM parameters

$\sigma_{\boldsymbol{n}}$		2.55/255	7.65/255
DRUNet	$\sqrt{\varepsilon} \times 255$	20	50
	α	1.25	1.25
	K	10	10
FastDVDnet	$\sqrt{\varepsilon} \times 255$	20	20
	α	1	0.5
	K	20	20

Table 3.3: Video super-resolution: selected values of (ε, α, K) PnP-ADMM parameters

s.f		×2		×4		×4	
kernel		Gauss. ($\sigma = 1.6$)		Gauss. ($\sigma = 1.6$)		Gauss. ($\sigma = 3.2$)	
$\sigma_{\boldsymbol{n}}$		0	2.55/255	0	2.55/255	0	2.55/255
DRUNet	$\sqrt{\varepsilon} \times 255$	40	55	60	60	60	70
	α	0.075	1.25	1	1.25	0.025	0.25
	K	10	10	10	10	10	10
FastDVDnet	$\sqrt{\varepsilon} \times 255$	20	30	55	55	50	55
	α	0.025	0.5	0.75	0.75	0.025	0.25
	K	20	20	20	20	20	20

$\sigma_{\boldsymbol{n}}$ to the result. For the sake of simplicity, we assume a cyclic convolution operator $\boldsymbol{k}$. The convolution with $\boldsymbol{k}$ can also be represented by a multiplication with a block circulant matrix H with circulant blocks. The degradation model becomes

$$\boldsymbol{y} = \boldsymbol{k} * \boldsymbol{x} + \boldsymbol{n} = H\boldsymbol{x} + \boldsymbol{n}. \tag{3.23}$$

In order to simulate blur which varies throughout the video (*e.g.* blur caused by camera shake due to random hand tremor), we randomly sample a kernel among the 8 samples used in [Lev+09] for each frame. While this type of non-blind deconvolution problem can also be solved on a frame-by-frame basis, it is still relevant for digital video.

For the sake of simplicity, let us assume that frames of $\boldsymbol{x}$ are periodic. Applying H can then be done very simply in the Fourier domain, writing $\mathcal{F}(H\boldsymbol{x}) = \mathcal{F}(\boldsymbol{k})\mathcal{F}(\boldsymbol{x})$, where $\boldsymbol{k}$ denotes the 2D blur kernel, $\mathcal{F}(.)$ denotes the 2D Fourier transform, $\overline{\mathcal{F}(.)}$ its conjugate and $\mathcal{F}^{-1}(.)$ its inverse. Using (3.21) with $A = H$ and the fact that the operators H and H^*H are diagonal in the Fourier domain, we see that the proximal operator of the data fidelity term of (3.6) can be written as

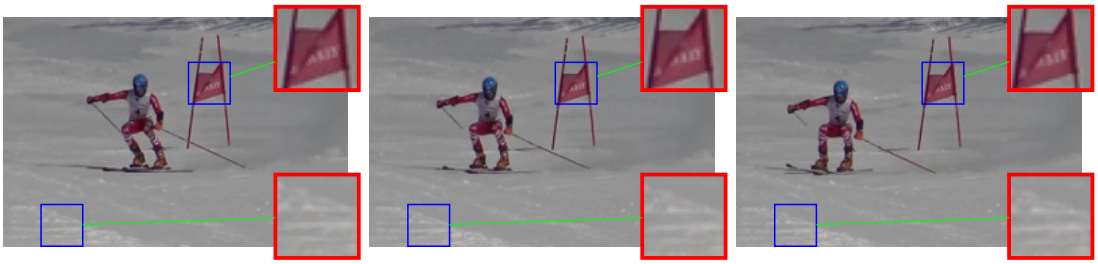
$$\operatorname{prox}_{\tau F(.,\boldsymbol{y})}(\boldsymbol{z}) = \left(\frac{\tau}{\sigma_n^2}H^*H + I_d\right)^{-1}\left(\frac{\tau}{\sigma_n^2}H^*\boldsymbol{y} + \boldsymbol{z}\right) \tag{3.24}$$

$$= \mathcal{F}^{-1}\left(\frac{\frac{\tau}{\sigma_n^2}\overline{\mathcal{F}(\boldsymbol{k})}\mathcal{F}(\boldsymbol{y}) + \mathcal{F}(\boldsymbol{z})}{\frac{\tau}{\sigma_n^2}\overline{\mathcal{F}(\boldsymbol{k})}\mathcal{F}(\boldsymbol{k}) + 1}\right). \tag{3.25}$$

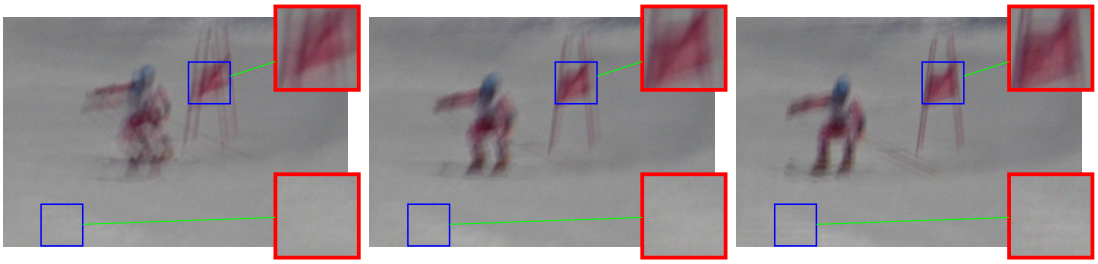
Results. The quantitative results of our PnP-ADMM method for video deblurring with the 8 randomized kernels of [Lev+09] on DAVIS-2017-test-480p [Pon+17] for $\sigma_{\boldsymbol{n}} \in \{2.55, 7.65\}/255$ are presented in Table 3.5. For reference, the table also includes the comparison of our method to DPIR [Zha+21a] and GSPnP [HLP22a], two state-of-the-art PnP image restoration methods. DPIR uses DRUNet in PnP-HQS with an annealing strategy on the noise level of their denoiser across iterations. GSPnP uses GS-DRUNet, the shallower version of DRUNet explicitly trained as a gradient step, in a slightly modified version of HQS. For both methods, we perform PnP restoration of each video frame sequentially, using the parameter values recommended by their respective authors for deblurring:

- $(\sigma_1, \sigma_K, \lambda, K) = (49, \sigma_{\boldsymbol{n}} \times 255, 0.23, 8)$ using the notations of [Zha+21a]: σ_K and σ_K are the level of the denoiser at the first and last of the K iterations, and λ controls the tradeoff between data fidelity and regularization.
- $(\sigma, \lambda_\nu, \epsilon, K) = (1.8\sigma_{\boldsymbol{n}}, 0.1, 10^{-5}, 400)$ using the notations of [HLP22a]: σ is the noise level the denoiser is set to, $\lambda_\nu = \sigma_{\boldsymbol{n}}^2\lambda$ where λ is the tradeoff parameter that relaxes the denoising step, ϵ[6] is the value of the relative difference between consecutive values of the objective function under which the algorithm is terminated, and K is the maximal number of iterations.

[6] not to be confused with ε, the variance of the denoiser in our notations.



(a) Unknown / ground truth



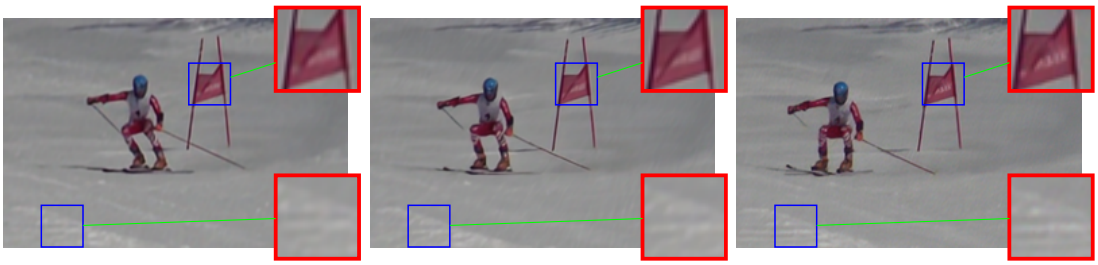
(b) Observation PSNR = 28.34 dB

Figure 3.5: Example of video blur ($\sigma_{\boldsymbol{n}} = 2.55$) with different kernels per frame on `giant-slalom` of DAVIS-2017-test-480p [Pon+17].

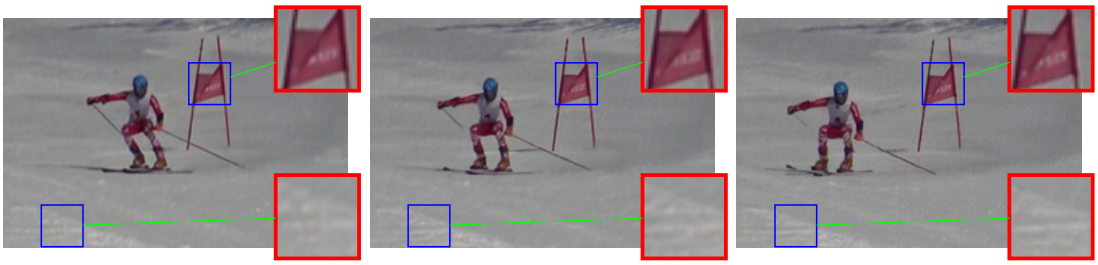
We turn off periodical geometric self-ensemble in DPIR (which consists in rotating / flipping $\boldsymbol{x}$ and $\boldsymbol{z}$ at each iteration with a period of 8 iterations to improve the performance of the denoising step), as it could also be implemented in our method but only yields marginal improvements and is not necessarily relevant to the comparisons of this chapter. With our video PnP-ADMM method, using FastDVDnet instead of DRUNet yields a higher PSNR/SSIM in almost all of the 30 videos observed at both noise levels. The performance gap is smaller between PnP-ADMM using FastDVDnet and DPIR, even though it also uses DRUNet; this suggests that using identical denoisers but a different optimization algorithm and a different set of parameters can lead to different PnP restoration results. Figure 3.5 illustrates an example of our video blur use case, and Figure 3.6 showcases the visual comparison of deblurring results. GSPnP also yields better performance than video PnP-ADMM with DRUNet, with a very similar performance profile to DPIR. It can be observed that FastDVDnet restores more image detail and its results are more temporally stable, whereas DPIR yields smoother frames with somewhat less high frequency content.

3.3.3 Video super-resolution

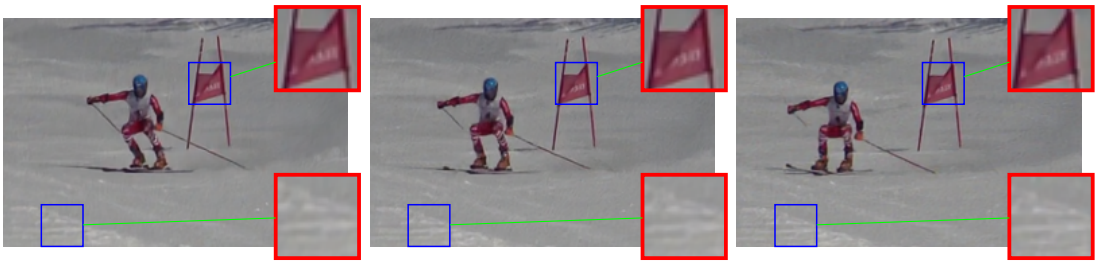
For video super-resolution, we use the classical degradation model where the low resolution video is obtained by downsampling the original high resolution video by a certain scale factor s in each dimension after convolution by a known anti-aliasing kernel $\boldsymbol{k}$. Writing H the $d \times d$ matrix representing the cyclic convolution operator and S the $m \times d$ downsampling matrix, the degradation model can be



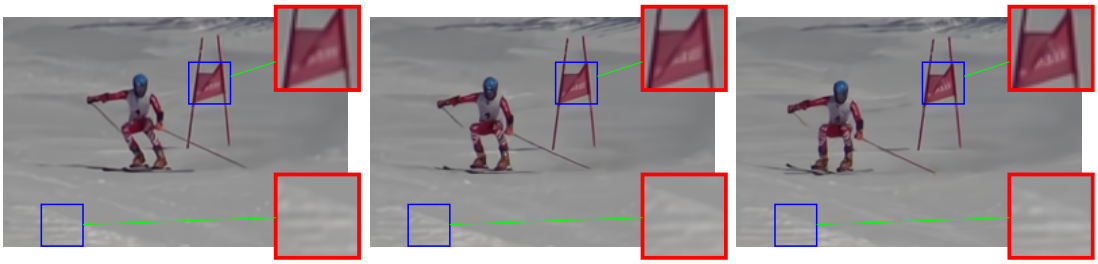
(a) Ours - DRUNet ($\sqrt{\varepsilon} = 20/255, \alpha = 1.25$) PSNR = 39.06 dB



(b) Ours - DRUNet ($\sqrt{\varepsilon} = 20/255, \alpha = 1$) PSNR = 37.84 dB



(c) Ours - FastDVDnet ($\sqrt{\varepsilon} = 20/255, \alpha = 1.0$) PSNR = 40.38 dB



(d) DPIR [Zha+21a] PSNR = 39.77 dB

Figure 3.6: Video deblurring ($\sigma_{\boldsymbol{n}} = 2.55$) on `giant-slalom` of DAVIS-2017-test-480p [Pon+17]. Ours - DRUNet and FastDVDnet are using their optimal values of (ε, α) (we also show Ours - DRUNet with $\alpha = 1$ for identical regularization weight versus Ours - FastDVDnet). While all methods perform well, Ours - FastDVDnet restores more image detail and its results are more temporally stable, whereas DPIR yields smoother frames with somewhat less high frequency content. GSPnP, while not included here because it is a late addition to the performance comparison, yields a PSNR of 40.02 dB.

written

$$\boldsymbol{y} = SH\boldsymbol{x} + \boldsymbol{n}. \tag{3.26}$$

Replacing A by SH in (3.21), we deduce that the proximal operator of the data fidelity term can be written in this case

$$\operatorname{prox}_{\tau F(.,\boldsymbol{y})}(\boldsymbol{z}) = \left(\frac{\tau}{\sigma_n^2} H^* S^* SH + I_d\right)^{-1} \left(\frac{\tau}{\sigma_n^2} H^* S^* \boldsymbol{y} + \boldsymbol{z}\right). \tag{3.27}$$

Using this formulation directly is not possible since it requires to invert a huge $d \times d$ matrix, which is not diagonal in the frequency domain. Instead, as done in [Zha+21a; HLP22a], we use the closed-form expression proposed in [Zha+16]. First, following the notations of [HLP22a], we write $\hat{z}_\tau = \frac{\tau}{\sigma_n^2} H^* S^* \boldsymbol{y} + \boldsymbol{z}$ and observe that

$$\operatorname{prox}_{\tau F(.,\boldsymbol{y})}(\boldsymbol{z}) = \left(\frac{\tau}{\sigma_n^2} H^* S^* SH + I_d\right)^{-1} \hat{z}_\tau \tag{3.28}$$

$$= \hat{z}_\tau - \frac{\tau}{\sigma_n^2} H^* S^* \left(\frac{\tau}{\sigma_n^2} SHH^* S^* + I_m\right)^{-1} SH\hat{z}_\tau. \tag{3.29}$$

The $m \times m$ matrix $SHH^*S^* + I_m$ can be inverted much more easily in the Fourier domain. Following again the notations of [HLP22a], we write $\Lambda = \operatorname{diag}(\mathcal{F}(\boldsymbol{k}))$ the $d \times d$ diagonal matrix containing the Fourier transform of the convolution kernel $\boldsymbol{k}$ on the diagonal. This matrix can also be written as a block diagonal matrix $\Lambda = \operatorname{diag}(\Lambda_1, \dots, \Lambda_{s^2})$, with blocks Λ_k (also diagonal) of size $m \times m$. Now, writing $\underline{\Lambda} = [\Lambda_1, \dots, \Lambda_{s^2}] \in \mathbb{R}^{m \times d}$, it follows easily that the operator SH corresponds to $\frac{1}{s}\underline{\Lambda}$ in the Fourier domain, *i.e.* $\mathcal{F}(SH\boldsymbol{z}) = \frac{1}{s}\underline{\Lambda}\mathcal{F}(\boldsymbol{z})$ for all $\boldsymbol{z} \in \mathbb{R}^d$. Finally, the proximal operator of the data term can be computed explicitly as

$$\operatorname{prox}_{\tau F(.,\boldsymbol{y})}(\boldsymbol{z}) = \hat{z}_\tau - \frac{\tau}{\sigma_n^2 s^2} \mathcal{F}^{-1}\left(\underline{\Lambda}^* \left(\frac{\tau}{\sigma_n^2 s^2} \underline{\Lambda}\underline{\Lambda}^* + I_m\right)^{-1} \underline{\Lambda}\mathcal{F}(\hat{z}_\tau)\right). \tag{3.30}$$

Results. The quantitative results of our PnP-ADMM method for $\times 2$ and $\times 4$ super-resolution with two Gaussian kernels for $\sigma_{\boldsymbol{n}} \in \{0, 2.55, 7.65\}/255$ are presented in Table 3.6. Once again, we also include the comparison to DPIR [Zha+21a] and GSPnP [HLP22a], with the parameter values recommended by the authors for super-resolution:

- $(\sigma_1, \sigma_K, \lambda, K) = (49, \max(s, \sigma_{\boldsymbol{n}} \times 255), 0.23, 24)$ using the notations of [Zha+21a].
- $(\sigma, \lambda_\nu, \epsilon, K) = (2\sigma_{\boldsymbol{n}}, 0.065, 10^{-6}, 400)$ using the notations of [HLP22a].

As in Section 3.3.2, we turn off periodical geometric self-ensemble for DPIR. Similarly to the previous restoration problem, PnP-ADMM with FastDVDnet produces better results than PnP-ADMM with DRUNet. We also observe that our method with FastDVDnet outperforms DPIR in the noiseless case, and is competitive with DPIR and GSPnP in noisy cases. An example is illustrated in Figure 3.7.

Table 3.4: Video interpolation of random missing pixels: selected values of (ε, α, K) PnP-ADMM parameters

ρ		0.5			0.9		
$\sigma_{\boldsymbol{n}}$		0	2.55/255	7.65/255	0	2.55/255	7.65/255
DRUNet	$\sqrt{\varepsilon} \times 255$	30	30	30	50	50	50
	α	0.5	3.0	2.5	2.75	2.5	2.75
	K	200	200	200	200	200	200
FastDVDnet	$\sqrt{\varepsilon} \times 255$	20	20	20	30	30	40
	α	1.75	3.0	1.5	2.25	2.75	1.5
	K	200	200	200	200	200	200

Table 3.5: Video deblurring: PSNR/SSIM on DAVIS-2017-test-480p [Pon+17] (each video limited to its 30 first frames)

$\sigma_{\boldsymbol{n}}$	2.55/255	7.65/255
blurred	22.75/0.599	12.66/0.533
Ours - DRUNet	36.20/0.938	32.07/0.856
Ours - FastDVDnet	**37.11**/0.949	**33.29/0.893**
DPIR [Zha+21a]	36.97/**0.950**	32.92/0.890
GSPnP [HLP22a]	36.69/0.947	32.81/0.885

Table 3.6: Video super-resolution: PSNR/SSIM on DAVIS-2017-test-480p [Pon+17] (each video limited to its 30 first frames)

s.f.	kernel	$\sigma_{\boldsymbol{n}}$	LR + bicubic	Ours - DRUNet	Ours - FastDVDnet	DPIR [Zha+21a]	GSPnP [HLP22a]
×2	Gauss. ($\sigma = 1.6$)	0	27.76/0.816	36.47/0.955	**36.86/0.958**	35.62/0.945	N.A.
		2.55/255	27.52/0.784	32.12/0.874	32.37/0.874	**32.56**/0.895	32.03/**0.896**
		7.65/255	26.06/0.615	29.86/0.806	30.25/0.817	**30.59**/0.845	30.39/**0.848**
×4	Gauss. ($\sigma = 1.6$)	0	24.74/0.714	30.00/**0.844**	**30.12**/0.839	29.75/0.840	N.A.
		2.55/255	24.61/0.689	29.23/0.809	**29.38**/0.805	29.31/**0.823**	29.06/0.817
		7.65/255	23.78/0.563	28.18/0.772	28.47/0.780	28.29/0.783	**28.48/0.786**
×4	Gauss. ($\sigma = 3.2$)	0	23.98/0.665	30.07/**0.844**	**30.15**/0.841	29.70/0.830	N.A.
		2.55/255	23.88/0.642	26.42/0.634	27.46/0.711	**27.94/0.767**	27.80/0.759
		7.65/255	23.17/0.518	26.37/0.698	26.54/0.701	26.53/0.713	**26.84/0.720**

3.3.4 Video interpolation of random missing pixels

We evaluate a final use case of our Plug-and-Play video restoration method: interpolation, which consists in estimating the values of hidden or missing pixels. In our experiments, we mask a proportion ρ of the pixels in the video according to a random spatio-temporal pattern. This inverse problem can be seen as a special case of compressed sensing [Don06] . The degradation model can be expressed as

$$\boldsymbol{y} = \boldsymbol{M} \odot \boldsymbol{x} + \boldsymbol{n} \tag{3.31}$$

where $\boldsymbol{M}$ is random with a proportion ρ of elements set to 0 and 1 elsewhere and $\odot$ denotes pixel-wise multiplication. The proximal operator of the data term for this type of problem can be written as

$$\text{prox}_{\tau F(.,\boldsymbol{y})}(\boldsymbol{x}[i]) = \begin{cases} \frac{\boldsymbol{x}[i] + \boldsymbol{y}[i]\frac{\tau}{\sigma_{\boldsymbol{n}}^2}}{1+\frac{\tau}{\sigma_{\boldsymbol{n}}^2}} & \text{if } \boldsymbol{M}[i] = 1 \\ \boldsymbol{x}[i] & \text{if } \boldsymbol{M}[i] = 0 \end{cases} \tag{3.32}$$

where $\boldsymbol{x}[i]$ is the value of the pixel i at a specific spatial location of a specific channel of a specific frame of the RGB video $\boldsymbol{x}$.

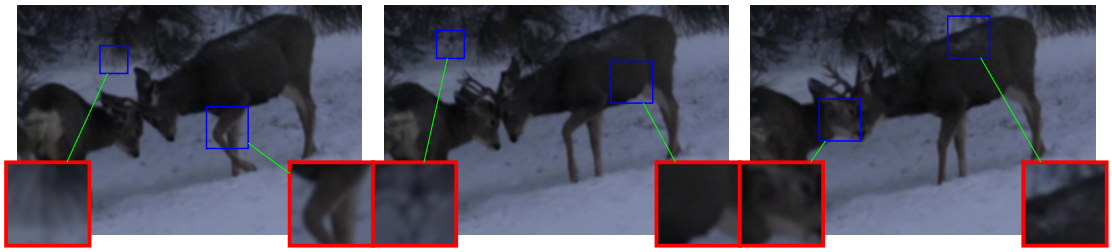
In the case where $\sigma_{\boldsymbol{n}} = 0$, the data-fitting term takes the form of a hard constraint and its proximal operator admits a closed form that is independent of τ (and thus of α in our PnP-ADMM method since we have $\tau \equiv \varepsilon/\alpha$):

$$\text{prox}_{\tau F(.,\boldsymbol{y})}(\boldsymbol{x}) = (1 - \boldsymbol{M}) \odot \boldsymbol{x} + \boldsymbol{M} \odot \boldsymbol{y}. \tag{3.33}$$

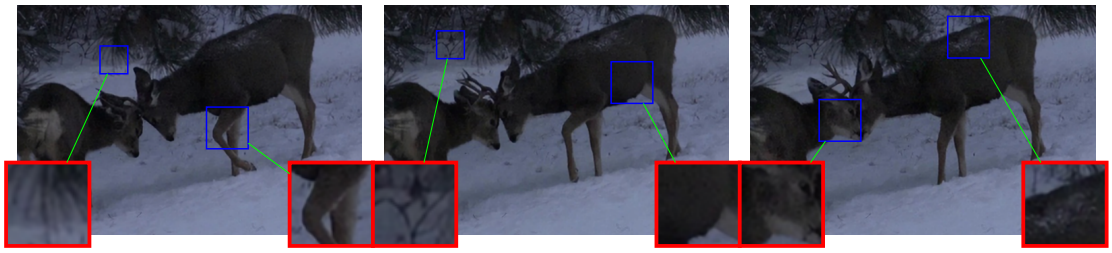
Results. Results of our method for interpolation of missing pixels for $\rho \in \{0.5, 0.9\}$ and $\sigma_{\boldsymbol{n}} \in \{0, 2.55, 7.65\}/255$ are shown in Table 3.7. We include GSPnP [HLP22a] to our comparison, with their parameter setting described for 50% of missing pixels and no noise: no backtracking procedure, $\sigma = 50$ for the first 10 iterations then $\sigma = 10$, $\lambda\tau = 1$ (where λ relaxes the denoising step and τ is the fixed step size), $\lambda = 0.1, K = 100$ using the notations of [HLP22a]. While we cannot be sure that these parameters are also suitable for the harder case where $\rho = 0.9$, we keep them as is because we have found no performance improvement by e.g increasing the number of iterations K to 200 or 400. We do not compare the performance of our method to DPIR here, as this problem is not part of the original publication. In this experiment, since the amount of missing pixels and their locations change from one frame to another, GSPnP and the DRUNet-based scheme that in effect operate on each frame separately clearly lag behind. Our FastDVDnet-based scheme takes advantage of its neighbor-using denoising step to provide a much more reliable and stable interpolation of the video. A visual example is also proposed in Figure 3.8.

3.3.5 Bayer to RGB video demosaicking

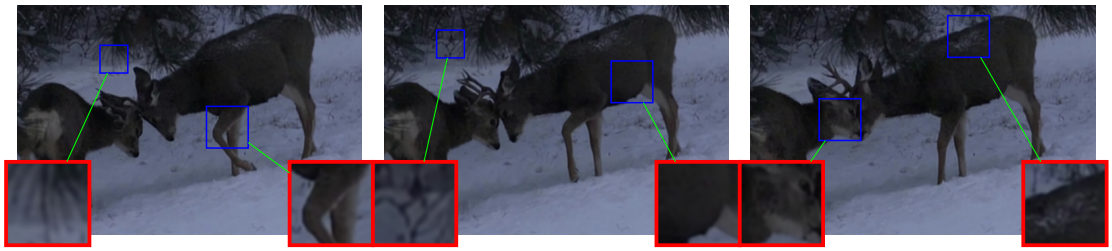
In addition to deblurring, super-resolution and interpolation of random missing pixels, we evaluate a very recurring topic of digital photography: demosaicking [Gun+05]. Indeed, the vast majority of CMOS image sensors currently used



(a) LR + bicubic PSNR = 34.17 dB



(b) Ours - FastDVDnet ($\sqrt{\varepsilon} = 20/255, \alpha = 0.025$) PSNR = 44.35 dB



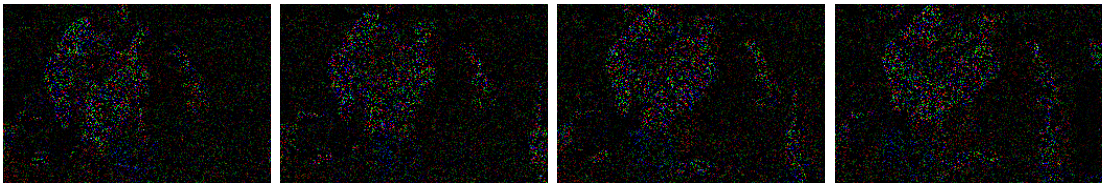
(c) DPIR [Zha+21a] PSNR = 42.91 dB

Figure 3.7: Video SR (×2/ Gauss. ($\sigma = 1.6$), $\sigma_{\boldsymbol{n}} = 0$) on the `deer` sequence of DAVIS-2017-test-480p [Pon+17]. Our video PnP-ADMM method with FastDVDnet retains slightly more detail than DPIR (Ours - DRUNet is not shown as it is similar to Ours - FastDVDnet with 43.97 dB).

Table 3.7: Video interpolation of random missing pixels: PSNR/SSIM after 200 iterations of PnP-ADMM on DAVIS-2017-test-480p [Pon+17] (each video limited to its 30 first frames)

ρ	$\sigma_{\boldsymbol{n}}$	masked	Ours - DRUNet	Ours - FastDVDnet	GSPnP [HLP22a]
0.5	0	9.44/0.142	44.73/**0.992**	**45.00**/0.991	38.87/0.965
	2.55/255	9.43/0.138	39.89/0.970	**41.83/0.981**	37.41/0.945
	7.65/255	9.40/0.117	37.37/0.954	**38.08/0.960**	32.61/0.819
0.9	0	6.88/0.047	26.14/0.820	**32.40/0.914**	28.41/0.801
	2.55/255	6.88/0.044	26.28/0.821	**32.22/0.910**	28.36/0.797
	7.65/255	6.86/0.032	26.70/0.813	**32.21/0.886**	27.97/0.766

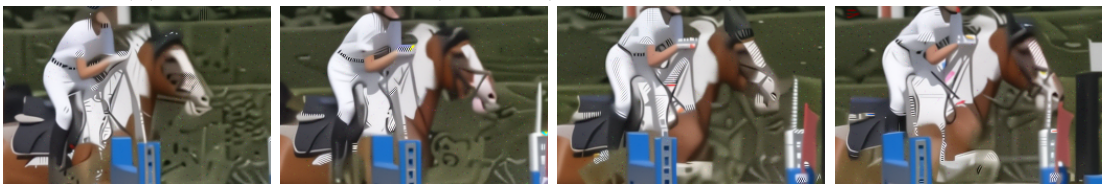
(a) Unknown / ground truth



(b) Observation PSNR = 8.23 dB

(c) Ours - DRUNet ($\sqrt{\varepsilon} = 50/255, \alpha = 2.75$) PSNR = 26.18 dB

(d) Ours - FastDVDnet ($\sqrt{\varepsilon} = 30/255, \alpha = 2.25$) PSNR = 29.55 dB



(e) GSPnP [HLP22a] PSNR = 25.35 dB

Figure 3.8: Video interpolation of random missing pixels ($\rho = 0.9, \sigma_{\boldsymbol{n}} = 0$) on the `horsejump-stick` sequence of DAVIS-2017-test-480p [Pon+17]. Our video PnP-ADMM method with FastDVDnet produces significantly better, temporally more stable results than both Ours - DRUNet and GSPnP, as it is the only method that leverages neighboring frames to solve the problem.

in smartphones, cameras and various computer vision systems are monochrome sensors stacked with an array of per-pixel colored filters called Color Filter Array (CFA). The most common array is the repetition of a 2×2 *Bayer* pattern, where two green color filters are put diagonally and the other two filters are red and blue. This means that the sensor output is undersampled in terms of colors at each pixel location, and an interpolation algorithm must be used to recover red, green and blue values at each pixel. Videos are of course not spared from that problem since they also come from CMOS image sensors. The degradation model for a synthetic video demosaicking inverse problem can be written as

$$\boldsymbol{y} = \boldsymbol{M} \odot \boldsymbol{x} + \boldsymbol{n} \tag{3.34}$$

where $\boldsymbol{M}$ is a binary mask corresponding to the CFA pattern and $\odot$ denotes pixel-wise multiplication. Notice that this is actually the same degradation model as the interpolation problem of Section 3.3.4 with a different, fixed mask.The proximal operator of the data term is thus identical to the expressions of Equations 3.31 and 3.32.

Parameter setting of the demosaicking problem. For this specific problem only, we experiment with a variant of our method of video PnP-ADMM: an annealed version, where the strength of the denoiser ε changes accross iterations. Annealing is used for demosaicking but not other problems because we have found no empirical improvements elsewhere. We are not the firsts to consider that kind of strategy, as variants of ADMM where the penalty parameter is updated at each iteration are already mentioned in [Boy+11], and similar annealing strategies were used both in PnP-ADMM [BRE16; CWE16] and other schemes such as PnP-SGD [Lau+23] and PnP-HQS [Zha+21a]. Annealing strategies such as this one are typically introduced to provide additional stability of iterative algorithms, particularly in cases that have limited guarantees of convergence.

We have seen in other restoration problems that the best value of α for a given denoiser and a given problem typically depends on the selected value of ε. This could be a problem for our annealing strategy, since we want ε to change across iterations (we might have to change α across iterations as well, making our 2D grid search strategy impossible in that case). Luckily, as shown in Equation (3.33) of Section 4.3, in both masking problems, when there is no noise (or little noise), the proximal operator of the data term does not depend on α, allowing us to consider our strategy of annealing of ε while keeping the same α across iterations. Similarly to what is done in [Zha+21a], we uniformly sample $\sqrt{\varepsilon}$ from a large noise level $\sqrt{\varepsilon}_0$ to a small one $\sqrt{\varepsilon_{K-1}}$ in log space. We set $\sqrt{\varepsilon_{K-1}} = 5/255$ for FastDVDnet since it was trained for noise levels in $[55, 5]/255$, and appropriate denoising performance is not guaranteed at noise levels outside of this range (as shown in Table 3.1). Given that DRUNet was trained for noise levels in $[50, 0]/255$ and that it generalizes better to unseen noise levels, we set $\sqrt{\varepsilon_{K-1}} = 1/255$ when using annealing with that network. An example of annealing of ε for $K = 200$ iterations (an empirically found sufficient number of iterations for PnP-ADMM applied to video demosaicking) is shown in Figure 3.9.

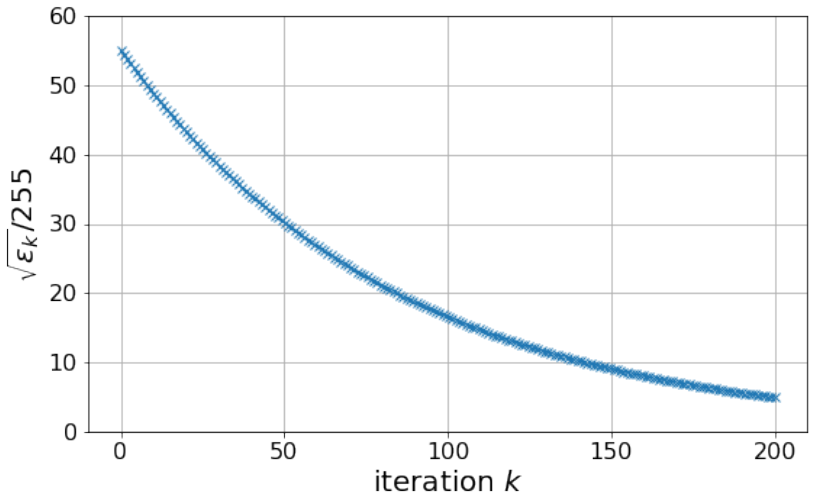


Figure 3.9: Example of annealing of ε with $\sqrt{\varepsilon_k} \in [55, 5]\,/255$ and $K = 200$

As done in other problems, we thus perform our grid search of optimal values of ε and α for FastDVDnet and DRUNet for Bayer RGGB to RGB demosaicking, at three noise levels $\sigma_{\boldsymbol{n}} = 0$, $\sigma_{\boldsymbol{n}} = 2.55/255$ and $\sigma_{\boldsymbol{n}} = 7.65/255$. We perform that grid search both with and without annealing of ε to later compare the performance of both PnP-ADMM versions. The resulting parameter values are shown in Table 3.8.

Besides from potentially increasing performance, another added benefit of decreasing ε across iterations is that it forces the distance between $\boldsymbol{x}$ and $\boldsymbol{z}$ to decrease when trying to minimize the augmented Lagrangian (Equation (3.19)), yielding additional stability of our PnP method in this problem where the data fidelity term is not strictly convex. This is illustrated in Figure 3.10.

Results. As done in Sections 4.1 and 4.2, we include DPIR [Zha+21a] to our performance comparison, with the parameter values recommended by the authors for demosaicking: $(\sigma_1, \sigma_K, \lambda, K) = (49, \max(0.6, \sigma_{\boldsymbol{n}} * 255), 0.23, 40)$ using the notations of [Zha+21a] (we disable here again the periodical geometric self-ensemble). It is interesting to note that DPIR initializes to an already crudely demosaicked version of the observation (the authors of [Zha+21a] use MATLAB's `demosaic` function). In contrast, we simply start from the mosaicked observation: $\boldsymbol{z}_0 = \boldsymbol{y}$, as initializing to MATLAB's `demosaic` output yielded little improvements to our PnP-ADMM results both in term of final performance and convergence speed. A drawback is that our method typically requires more iterations (100 to 200) than DPIR with its MATLAB initialization (40). Conversely, we found in our experiments that DPIR would also require more iterations with a mosaicked initialization, but we leave that out of our performance comparison, as no number of iterations K was provided by its authors for such a use case. Quantitative results of our PnP-ADMM methods on the 2017 test set of the DAVIS dataset [Pon+17] are available in Table 3.9. These results confirm the benefit of using an annealed

Table 3.8: Video demosaicking: selected values of (ε, α, K) PnP-ADMM parameters

$\sigma_{\boldsymbol{n}}$		0	2.55/255	7.65/255
DRUNet	$\sqrt{\varepsilon} \times 255$	60	60	60
	α	0.25	2.5	3
	K	200	200	200
FastDVDnet	$\sqrt{\varepsilon} \times 255$	20	20	20
	α	0.25	3	1.5
	K	200	200	200
DRUNet (annealing)	$\sqrt{\varepsilon_0} \times 255$	70	70	70
	α	0.25	0.75	0.5
	K	200	200	200
FastDVDnet (annealing)	$\sqrt{\varepsilon_0} \times 255$	30	55	50
	α	0.25	1	0.5
	K	200	200	200

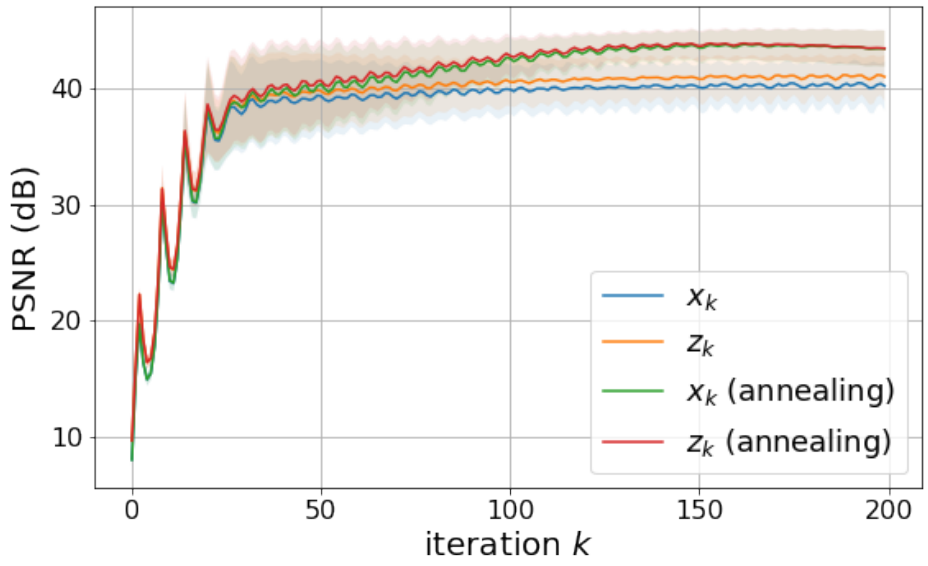


Figure 3.10: Demosaicking ($\sigma_{\boldsymbol{n}} = 0$): PSNR of split variables across iterations $\boldsymbol{x}_k$, $\boldsymbol{z}_k$ with and without annealing (FastDVDnet, $\sqrt{\varepsilon_0} = \sqrt{\varepsilon} = 20/255$). Results are computed and averaged on 256×256 crops of the first 30 frames of sequences `aerobatics`, `girl-dog`, `horsejump-stick` and `subway` of DAVIS-2017-test-480p [Pon+17]

Table 3.9: Video demosaicking: PSNR/SSIM on DAVIS-2017-test-480p [Pon+17] (each video limited to its 30 first frames)

$\sigma_{\boldsymbol{n}}$	0	2.55/255	7.65/255
mosaicked	8.21/0.100	8.21/0.096	8.18/0.079
Ours - DRUNet	33.23/0.921	32.20/0.879	31.22/0.847
Ours - FastDVDnet	39.78/0.971	38.78/0.964	36.68/**0.952**
Ours - DRUNet (annealing)	45.15/0.992	40.55/0.973	**36.75**/0.951
Ours - FastDVDnet (annealing)	43.92/0.990	**41.04/0.981**	36.40/0.951
DPIR [Zha+21a]	**45.72/0.993**	40.05/0.972	34.48/0.905

version of PnP-ADMM for demosaicking. In this non strongly convex case where the degradation pattern is identical for all frames, FastDVDnet performs better than DRUNet in some scenarios, while the opposite is true in others. It is worth noting that DPIR performs better than our method (with annealing and either denoiser) in the noiseless case, but worse than our method (with annealing and either denoiser) at $\sigma_{\boldsymbol{n}} = 7.65/255$.

3.3.6 Video 3D Deconvolution

Another experiment we conducted was 3D deconvolution. This use case is more of a toy example where the video degradation process cannot be directly applied to each individual frame. The convolution operation can again be represented by a multiplication with a block circulant matrix H with circulant blocks, but the cyclic convolution operator $\boldsymbol{k}$ will be 3D and thus have the ability to average values of pixels of multiple frames. Formally the forward degradation model is identical to Equation 3.23. If the kernel is uniform of size $d \times 1 \times 1$, where d is the number of frames of interest, we are performing frame averaging.

Results. In this case, per-frame PnP methods such as DPIR or GSPnP cannot be applied. However, we can still evaluate the usefullness of a video denoiser by comparing FastDVDnet to DRUNet in our video PnP-ADMM method. For each model, we reuse the optimal parameter setting for video deblurring (Table 3.2) in order to avoid rerunning the 2D grid search for this specific problem. Quantitative results of our method on the DAVIS test set [Pon+17] for uniform 3D kernels can be found in Table **??** As in other experiments, FastDVDnet significantly outperforms DRUNet, once again showcasing the interest of using a multi-frame denoiser for video restoration. Figure 3.11 showcases an example where FastDVDnet produces a lot less ghosting artifacts originating from the content of surrounding frames.

Table 3.10: Video 3D deconvolution: PSNR/SSIM on DAVIS-2017-test-480p [Pon+17] (each video limited to its 30 first frames)

kernel	unif. (4 frames)		unif. (8 frames)	
$\sigma_{\boldsymbol{n}}$	2.55/255	7.65/255	2.55/255	7.65/255
convolved	23.72/0.722	22.71/0.536	21.73/0.642	21.04/0.466
Ours - DRUNet	35.71/0.928	30.82/0.792	33.73/0.923	29.86/0.820
Ours - FastDVDnet	**37.51/0.946**	**32.75/0.866**	**36.05/0.938**	**30.51/0.839**

If we were to apply a particular succession of 3D uniform and zero-valued kernels, we can obtain a 3D deconvolution case akin to temporal deblurring which could be applied to datasets such as GoPro [NKL17] or REDS [Nah+19] where the blurry videos are obtained by averaging successive frames to produce a single blurry image resembling a long exposure image (the zero-valued kernels in this case are to take into account that all frames used to produce a blurry image will not be used in the subsequent blurry image, which is different from applying the same uniform 3D kernel). This experiment will be the subject of future work.

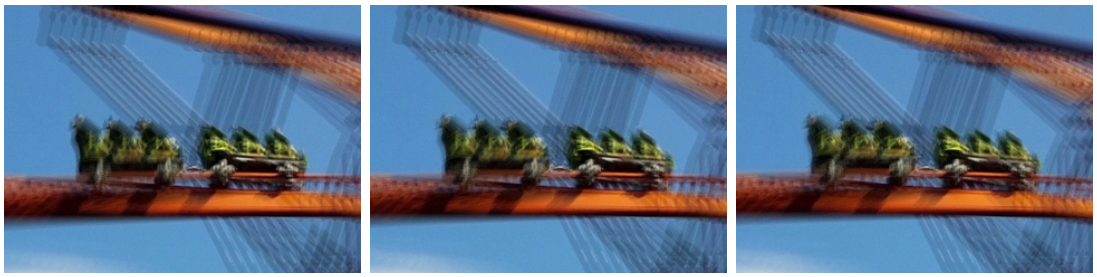
3.3.7 Expansiveness of video denoising operators

We have seen with our previous experiments that the original FastDVDnet can produce satisfactory results and early convergence in video PnP restoration. However, this network was only trained for Gaussian noise removal, and we have no guarantees that it is nonexpansive or Lipschitz-continuous. Let us experimentally study the expansiveness of FastDVDnet (with its pretrained weights provided by the authors of [TDV20]) for video denoising.

To do so, we first set ourselves to compute the spectral norm of the Jacobian of the denoising operator using power iterations until convergence as done in [HLP22b; Pes+21]. We select 256×256 center crops of the first 30 frames of videos of the DAVIS-2017-test-480p [Pon+17] as our evaluation dataset. We also evaluate the expansiveness of DRUNet for video denoising, as well as the residual of both denoisers at noise levels $\sigma_{\boldsymbol{n}} \in \{10, 25, 50, 100\}/255$. As can be seen in Table 3.11, neither vanilla FastDVDnet nor DRUNet (as discussed in [HLP22a; HLP22b]) are 1-Lipschitz, and neither are their residuals. This means that we cannot directly build upon the theoretical convergence guarantees of [Ryu+19] or [Pes+21] for PnP video restoration when using these operators. FastDVDnet consistently has a lower spectral norm than DRUNet however.

Furthermore, in our PnP restoration experiments, we have noticed that video PnP-ADMM with FastDVDnet is significantly less prone to divergence than both video PnP-ADMM with DRUNet and DPIR [Zha+21a]. To illustrate this, we perform x4 super-resolution using two different kernels at two noise levels on 4 video sequences, using our method and DPIR for 2000 iterations on purpose to assess their long-term behaviour. Results are shown in Figures 3.12, 3.13, 3.14

(a) Unknown / ground truth



(b) Observation PSNR = 16.14 dB

(c) Ours - DRUNet ($\sqrt{\varepsilon} = 20/255, \alpha = 1.25$) PSNR = 31.96 dB

(d) Ours - FastDVDnet ($\sqrt{\varepsilon} = 20/255, \alpha = 1$) PSNR = 36.69 dB

Figure 3.11: Video 3D deconvolution (unif. (8 frames), $\sigma_{\boldsymbol{n}} = 2.55$) on the `rollercoaster` sequence of DAVIS-2017-test-480p [Pon+17]. Our video PnP-ADMM method with FastDVDnet produces more temporally stable results and less ghosting than Ours - DRUNet.

Table 3.11: Maximal value of the spectral norm of the Jacobian of video denoising operators on 256×256 30-frame videos of DAVIS-2017-test-480p [Pon+17] at various noise levels

$\sigma_{\boldsymbol{n}}$	10/255	25/255	50/255	100/255
$\|\|J_{(\text{FastDVDnet})}\|\|_S$	2.94	4.08	4.78	2.67
$\|\|J_{(\text{DRUNet})}\|\|_S$	10.99	8.30	10.96	6.91
$\|\|J_{(\text{Id}-\text{FastDVDnet})}\|\|_S$	1.94	3.08	3.78	1.67
$\|\|J_{(\text{Id}-\text{DRUNet})}\|\|_S$	9.99	7.30	9.96	5.92
$\|\|J_{(2\text{FastDVDnet}-\text{Id})}\|\|_S$	4.87	7.15	8.56	4.33
$\|\|J_{(2\text{DRUNet}-\text{Id})}\|\|_S$	20.98	15.61	20.93	12.83

and 3.15. We observe on these long runs that PnP-ADMM with DRUNet is very unstable, and can easily diverge on some experiments. DPIR is usually more stable than Ours - DRUNet (probably in part thanks to its annealing strategy) but also diverges in some cases. In contrast, Ours - FastDVDnet showcases greater stability and does not diverge.

In future work, we would like to train non-expansive versions of FastDVDnet and evaluate the impact of 1-Lipschitzness on denoising as well as PnP restoration. To do so, we would adopt a strategy of training with a soft constraint very similar to [Pes+21; HLP22b]: we start by pretraining FastDVDnet for standard Gaussian denoising with L2 loss, then fine-tune the network with a training loss that has an additional penalty term enforcing the spectral norm of the Jacobian of some operator to be < 1. We would conduct experiments with contraining the denoiser D itself, its residual $\text{Id} - D$ (for fixed-point convergence guarantees as in [Ryu+19]) or the operator $2D - \text{Id}$ (the denoiser then being the resolvant of a maximally monotone operator as in [Pes+21]).

3.3.8 Runtimes

Plug-and-play methods are typically not real-time methods, because they imply multiple iterations of an alternate optimization algorithm, and each iteration itself includes a forward pass through a denoising algorithm. Our method is no exception. That said, our method is typically faster than DPIR when performing video restoration with similar initialization, since the number of iterations required by DPIR for a single frame must be multiplied by the number of frames of the video. In contrast, one iteration of our method processes the whole video at once. Actual runtimes of course depend on the required number of iterations, which can in turn depend on the scene content, the difficulty of the problem, the type of initialization and the choice of parameters. Here are the runtimes with the parameters described in the article (8 CPU AMD, 1 NVIDIA A100, 64GB RAM, DAVIS-2017-test-480p [Pon+17]):

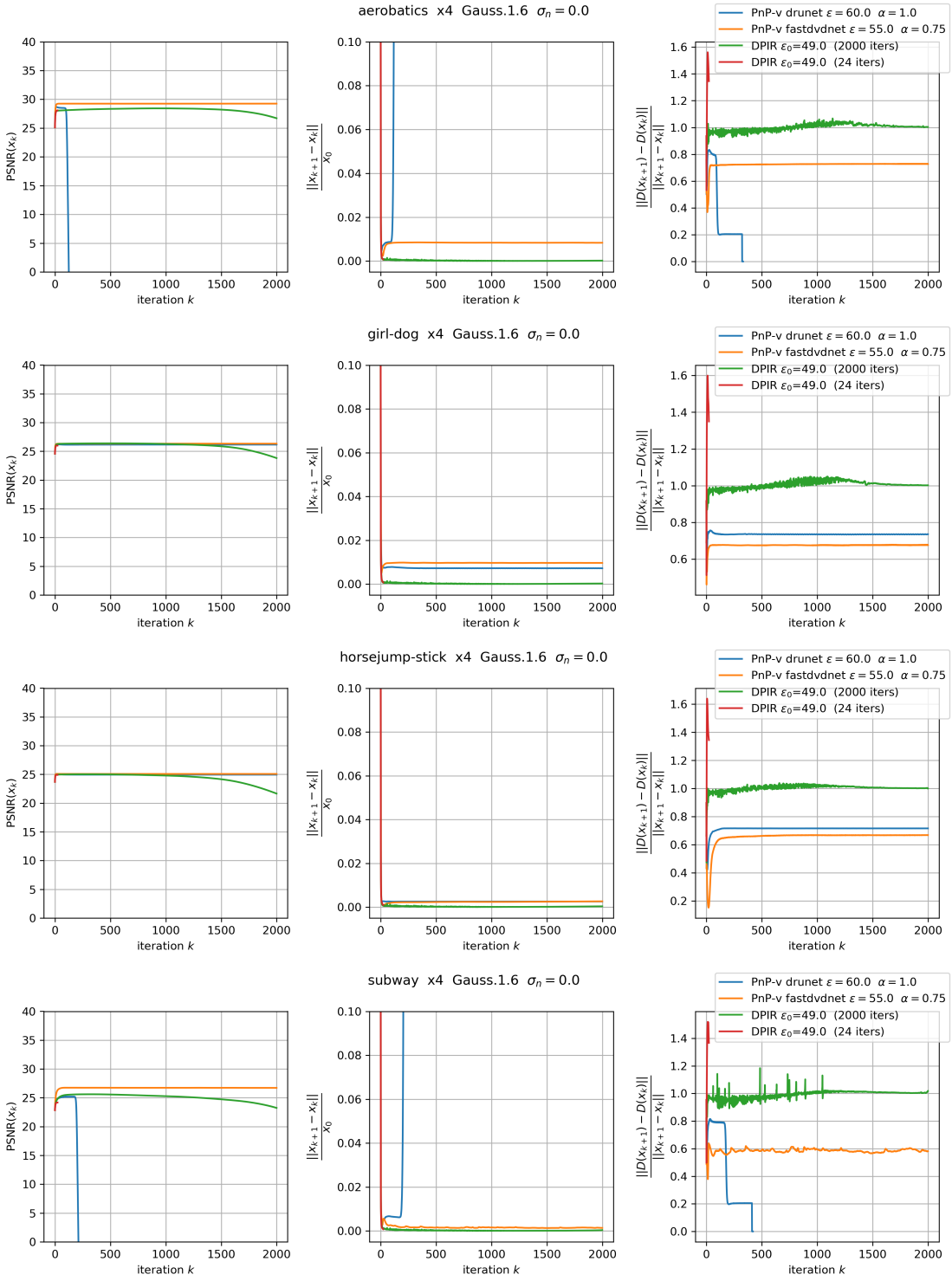


Figure 3.12: Long term behaviour of Video PnP-ADMM and DPIR [Zha+21a] for x4 super-resolution (Gauss.($\sigma = 1.6$), $\sigma_{\boldsymbol{n}} = 0$) on four center-cropped 256×256 30-frame videos of DAVIS-2017-test-480p [Pon+17]

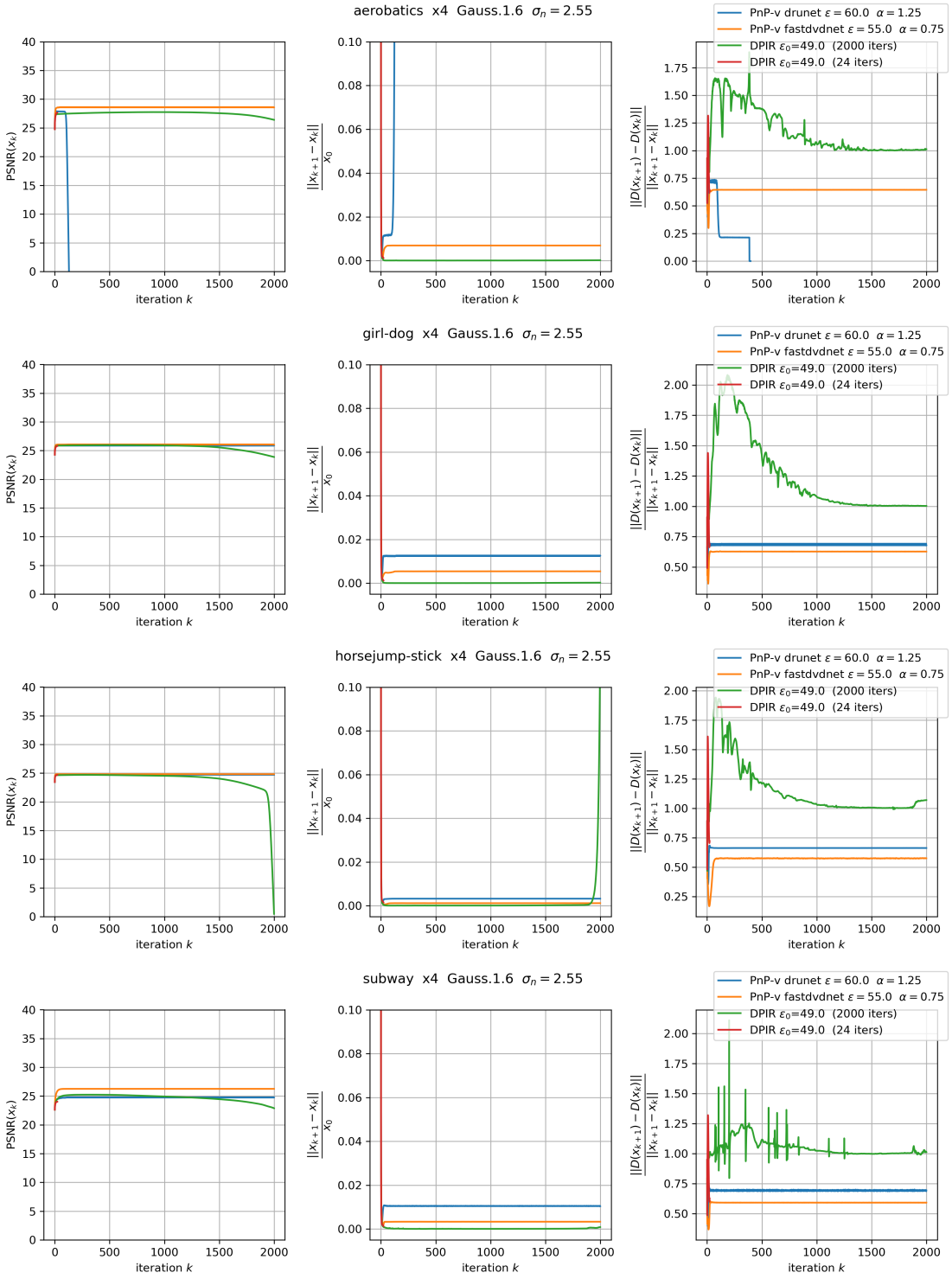


Figure 3.13: Long term behaviour of Video PnP-ADMM and DPIR [Zha+21a] for x4 super-resolution (Gauss.($\sigma = 1.6$), $\sigma_{\boldsymbol{n}} = 2.55$) on four 256×256 30-frame videos of DAVIS-2017-test-480p [Pon+17]

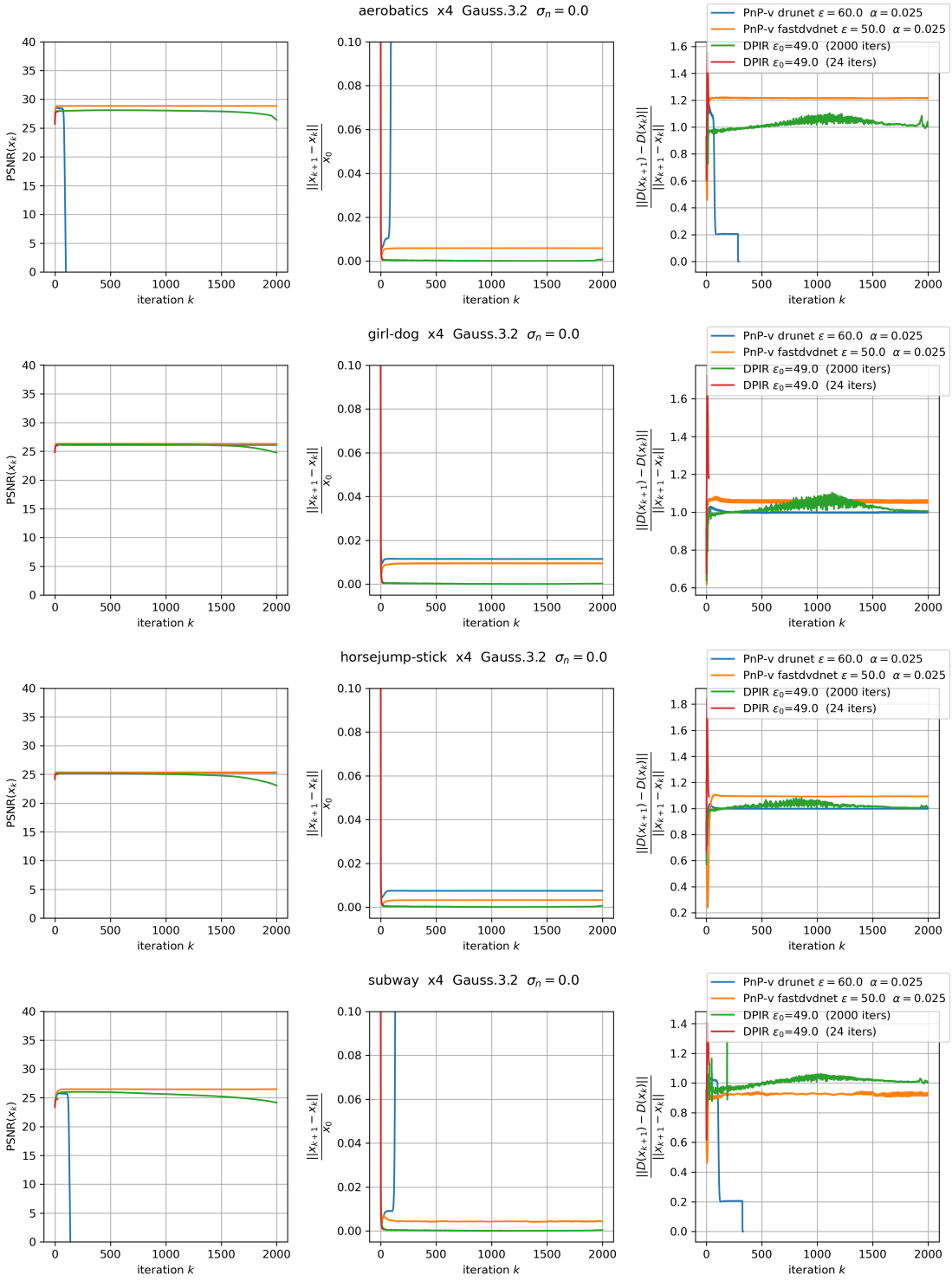


Figure 3.14: Long term behaviour of Video PnP-ADMM and DPIR [Zha+21a] for x4 super-resolution (Gauss.($\sigma = 3.2$), $\sigma_{\boldsymbol{n}} = 0$) on four 256×256 30-frame videos of DAVIS-2017-test-480p [Pon+17]

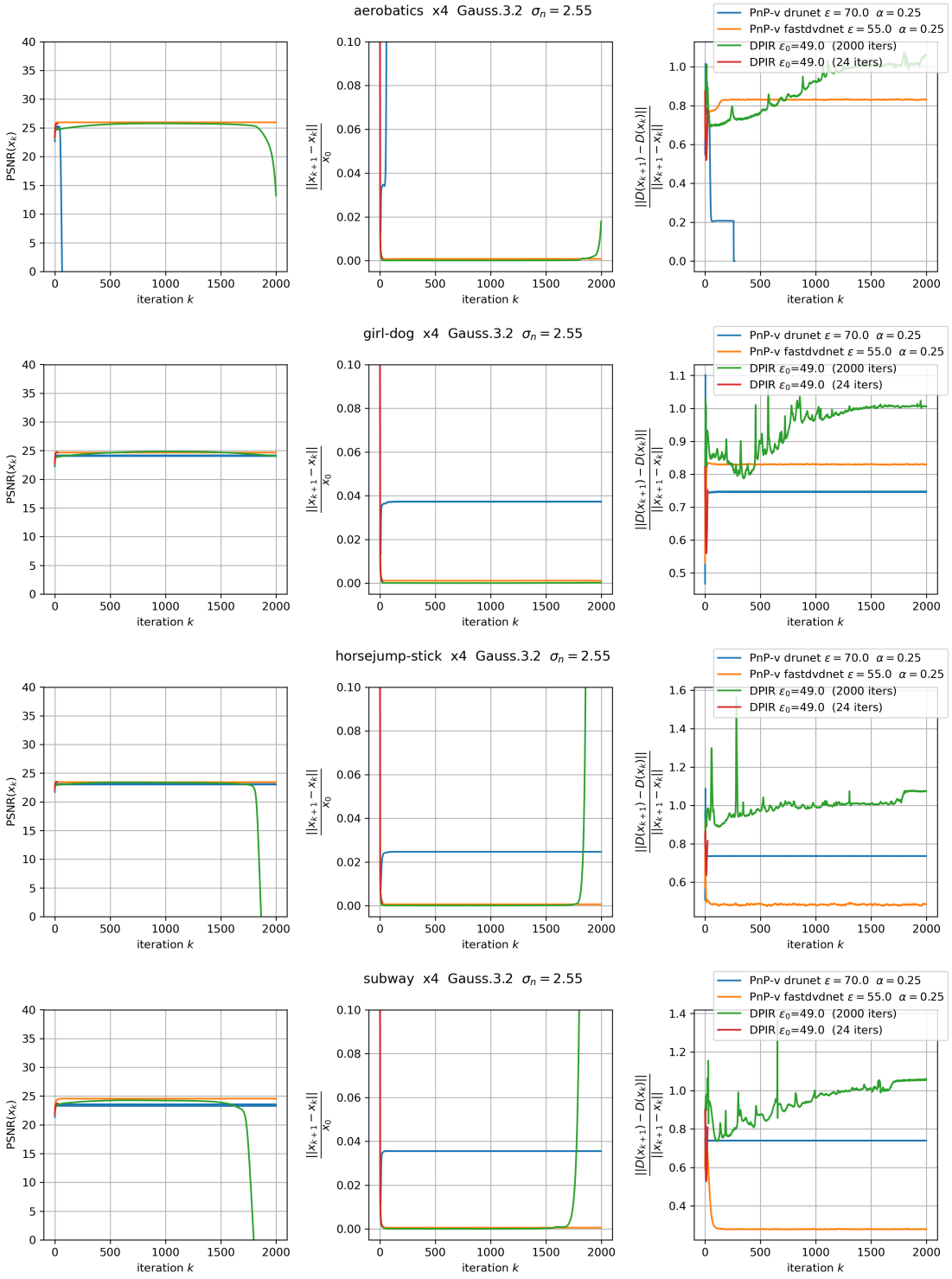


Figure 3.15: Long term behaviour of Video PnP-ADMM and DPIR [Zha+21a] for x4 super-resolution (Gauss.($\sigma = 3.2$), $\sigma_{\boldsymbol{n}} = 2.55$) on four 256×256 30-frame videos of DAVIS-2017-test-480p [Pon+17]

- **deblurring**: Ours - FastDVDnet 1.2s/frame (20 it.), Ours - DRUNet 1.4s/frame (10 it.), DPIR 2.7s/frame (8 it.)
- **super-resolution**: Ours - FastDVDnet 1.2s/frame (20 it.), Ours - DRUNet 1.4s/frame (10 it.), DPIR 8.1s/frame (24 it.)
- **interpolation**: Ours - FastDVDnet 12s/frame (200 it.), Ours - DRUNet 29s/frame (200 it.)
- **demosaicking**: Ours - FastDVDnet 12s/frame (200 it.), Ours - DRUNet 29s/frame (200 it.), DPIR 14s/frame (40 it.)
- **3D deconvolution**: Ours - FastDVDnet 1.2s/frame (20 it.), Ours - DRUNet 1.4s/frame (10 it.)

3.3.9 Comparison to state-of-the-art learning-based methods

While Plug-and-Play methods try to combine advantages of model-based and learning-based methods and have the added flexibility of being reusable for different restoration problems with a single denoising algorithm, we do not expect them to perform as good as state-of-the-art learning-based methods specifically trained for a given task. We put that expectation to the test in this Section by comparing our video PnP-ADMM method to powerful deep learning video restoration methods. Said comparison is performed for the $\times 4$ video super-resolution task, as the other experiments we have conducted (deblurring with varying per-frame kernels, interpolation of random missing pixels, 3D deconvolution) feature degradation processes that are unlikely to be found in most works of the deep learning video restoration literature (*e.g.* video inpainting methods typically mask large areas of an image instead of random pixels, and video deblurring methods usually generate blurry images by averaging frames captured at a higher framerate).

We compare our method to three state-of-the art learning-based video SR methods:

- EDVR [Wan+19] performs video super-resolution in a sliding window fashion, taking $2N + 1$ ($N \in \{2, 3\}$ depending on the version) frames as input and outputs a high resolution version of the center frame. Key features of the architecture include an alignment module that uses deformable convolutions to align feature maps in a coarse-to-fine pyramid structure, and a feature fusion module that leverages temporal and spatial attention (by computing element-wise correlation in an embedding space between different feature maps to weigh them before fusion). The network is presented as a generic architecture suitable for many restoration tasks such as denoising, deblocking, deblurring or super-resolution. EDVR won all four tracks of the NTIRE 2019 video deblurring and super-resolution challenges [Nah+19].

- BasicVSR [Cha+21] is an architecture designed around four main components of video super-resolution: propagation, alignment, aggregation, and upsampling. It is a bidirectional recurrent network (*i.e.* it performs forward propagation by looking at the current frame and the next, and backward propagation by looking at the current frame and the previous). The propagation branches contain a flow estimation module, a spatial warping module and residual blocks; forward and backward propagation outputs are inputs to the upsampling module that contains multiple pixel-shuffle and convolution operations. BasicVSR performs competitively with EDVR for a significantly smaller number of parameters and runtime.

- BasicVSR++ [Cha+22] is an enhanced version of BasicVSR whose main improvements are second-order grid propagation (*i.e.* use the previous frame and the second previous frame for backward propagation), and flow-guided deformable alignement (combine deformable convolutions and optical flow for alignment). BasicVSR++ ranked first place three times and second place once on the NTIRE 2021 video SR tracks [Son+21].

We compare our method using DRUNet and FastDVDnet as regularizers to EDVR, BasicVSR and BasicVSR++ on two datasets, DAVIS-2017-test-480p [Pon+17] which we've used in many of our experiments so far, and Vid4 [CD14], a popular video super-resolution dataset comprised of four video sequences of varying resolution and length. Said dataset is a bit less diversified than DAVIS, as it only features four videos and some of them feature very little camera or scene motion, but we still include it because these are potential use cases of video SR and is widely used in the video SR literature. It is important to emphasize that the SR degradation process used during training has a direct impact on the SR performance at test time. Sadly, off-the-shelf versions of EDVR are only trained with $\times 4$ bicubic downsampling, so performance on our use case of kernel convolution + downsampling will be reduced. On the other hand, we use the versions of BasicVSR and BasicVSR++ trained specifically for Blur Downsampling (BD), which actually uses a Gaussian blur kernel of standard deviation 1.6 (which is a kernel that we also used in some of our experiments of Section 3.3.3).

Table 3.12: Video super-resolution: PSNR↑/SSIM↑/LPIPS↓ on DAVIS-2017-test-480p [Pon+17]

sf / kernel	×4 / Gauss. ($\sigma = 1.6$)		×4 / Gauss. ($\sigma = 3.2$)	
σ_n	0	2.55/255	0	2.55/255
bicubic	25.07/0.71/0.40	24.93/0.69/0.46	24.44/0.67/0.53	24.31/0.65/0.56
Ours - DRUNet	29.93/0.83/0.29	29.18/**0.80**/0.36	30.01/**0.83**/0.30	26.42/0.65/0.61
Ours - FastDVDnet	**30.08**/0.83/0.27	**29.33**/**0.80**/0.36	**30.12**/**0.83**/**0.28**	**27.44**/**0.72**/**0.52**
EDVR [Wan+19]	29.01/0.84/0.19	28.00/0.77/**0.32**	25.53/0.70/0.50	25.18/0.66/0.54
BasicVSR [Cha+21]	29.61/0.86/0.18	28.21/0.77/0.36	25.90/0.71/0.49	25.33/0.64/0.58
BasicVSR++ [Cha+22]	29.76/**0.87**/**0.17**	28.46/0.79/0.33	25.89/0.71/0.49	25.41/0.65/0.58

Table 3.13: Video super-resolution: PSNR↑/SSIM↑/LPIPS↓ on Vid4 [CD14]

sf / kernel	×4 / Gauss. ($\sigma = 1.6$)		×4 / Gauss. ($\sigma = 3.2$)	
σ_n	0	2.55/255	0	2.55/255
bicubic	20.37/0.51/0.54	20.33/0.50/0.58	19.85/0.45/0.70	19.82/0.44/0.73
Ours - DRUNet	23.29/0.68/0.42	23.10/0.66/0.46	23.50/0.68/0.43	21.89/0.54/0.67
Ours - FastDVDnet	24.01/0.72/0.38	23.77/0.70/0.42	**23.78**/**0.70**/**0.40**	**22.46**/**0.59**/**0.58**
EDVR [Wan+19]	23.16/0.73/0.26	22.99/0.68/0.32	20.58/0.49/0.66	20.48/0.47/0.67
BasicVSR [Cha+21]	24.03/0.78/0.22	23.63/0.72/0.34	20.84/0.50/0.64	20.69/0.47/0.60
BasicVSR++ [Cha+22]	**24.40**/**0.80**/**0.21**	**23.97**/**0.74**/**0.30**	20.84/0.50/0.64	20.71/0.47/0.70

Quantitative results are presented in Tables 3.12 and 3.13. As seen in previous experiments, our method with FastDVDnet typically outperforms using DRUNet. Since the learning-based methods were not trained with additional noise and two out of three methods were trained with BD using a Gaussian kernel of standard deviation 1.6, let us first look at the first column ($\times 4$ / Gauss.($\sigma = 1.6$), $\sigma_{\boldsymbol{n}} = 0$). Learning-based methods yield higher SSIM and LPIPS than our PnP method, including EDVR that was trained using bicubic downsampling: they are better at recovering textures of the high resolution video, and generally yield higher perceptual similarity. By contrast, our PnP method tends to maximize the PSNR (*i.e.* minimize the mean-squared error) and generally produces frames that are perceived as blurrier. This is illustrated in Figures 3.16, 3.17 and 3.18. Looking at the other columns of Tables 3.12 and 3.13, when the learning-based methods encounter sequences with a LR degradation process that differs from the one seen during training (*e.g.* different blur kernel, noise added to downsampled frames), their performance tends to decrease rapidly. On the other hand, our PnP method has the added flexibility of handling any blur kernel and varying amounts of noise without retraining (another perk not shown in the results is the possibility to select a different super-resolution factor). An example of $\times 4$ super-resolution with a different kernel (Gaussian of standard deviation 3.2) and added noise with $\sigma_{\boldsymbol{n}} = 2.55$ is shown in Figure 3.19. Figure 3.20 illustrates using a model trained using bicubic downsampling (EDVR) on a sequence downscaled with the classical kernel convolution + downsampling method.

(a) Ours - FastDVDnet PSNR↑/SSIM↑/LPIPS↓ 29.08/0.87/0.19

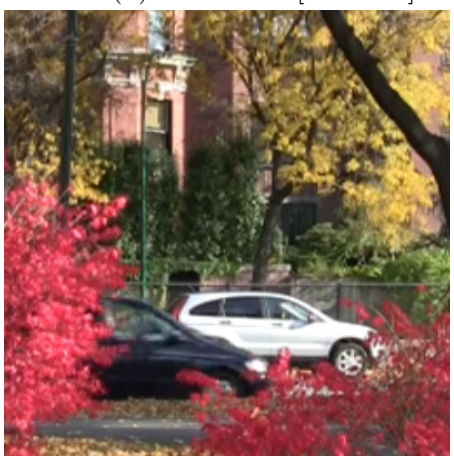
(b) BasicVSR [Cha+21] PSNR↑/SSIM↑/LPIPS↓ 26.07/0.85/0.17

(c) ground truth

Figure 3.16: Super-resolution (×4 / Gauss.($\sigma = 1.6$), $\sigma_{\boldsymbol{n}} = 0$) on the `aerobatics` sequence of DAVIS-2017-test-480p [Pon+17] (256 × 256 center crop of frame 10 shown here). While Ours - FastDVDnet has higher PSNR/SSIM and BasicVSR has higher relative absolute error around edges, BasicVSR produces a result whose perceived sharpness is closer to the ground truth.

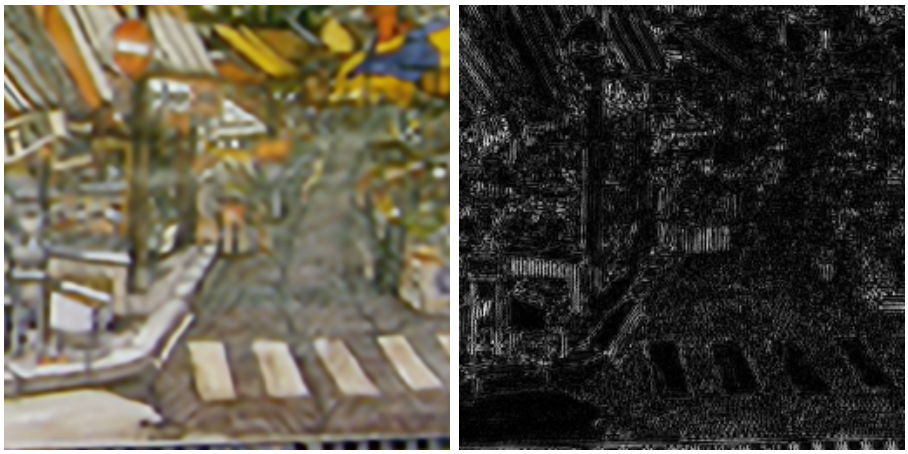
(a) Ours - FastDVDnet PSNR/SSIM 23.88/0.69/0.37

(b) BasicVSR [Cha+21] PSNR/SSIM 23.98/0.74/0.27

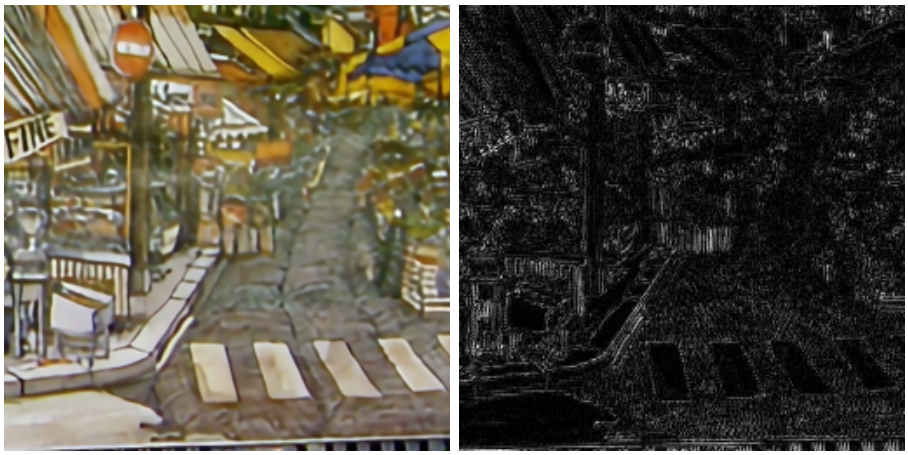


(c) ground truth

Figure 3.17: Super-resolution (×4 / Gauss.($\sigma = 1.6$), $\sigma_{\boldsymbol{n}} = 0$) on the `foliage` sequence of Vid4 [CD14] (256 × 256 center crop of frame 13 shown here). While both methods are close in terms of PSNR, BasicVSR produces a sharper result and is better at recovering structure of the original sequence.



(a) Ours - FastDVDnet PSNR↑/SSIM↑/LPIPS↓ 19.98/0.53/0.57



(b) BasicVSR [Cha+21] PSNR↑/SSIM↑/LPIPS↓ 20.93/0.67/0.34

(c) ground truth

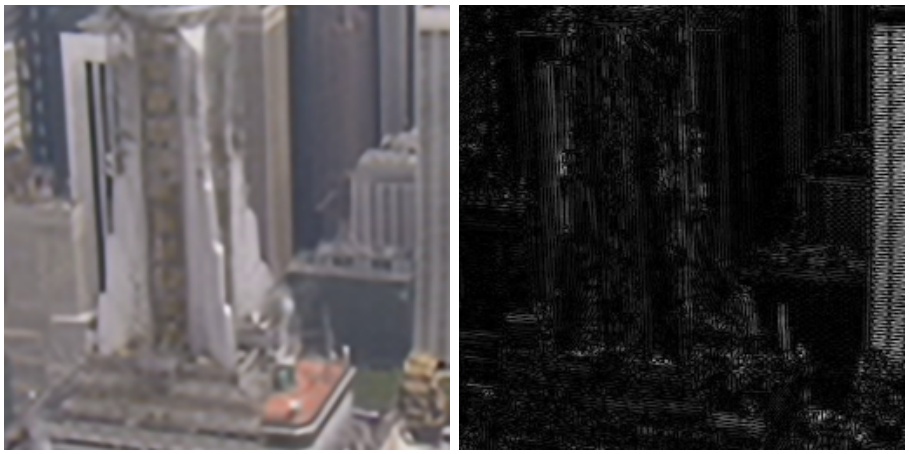
Figure 3.18: Super-resolution (×4 / Gauss.($\sigma = 1.6$), $\sigma_{\boldsymbol{n}} = 0$) on the `calendar` sequence of Vid4 [CD14] (256 × 256 crop of frame 13 shown here). BasicVSR significantly outperforms our method, recovering more high frequency detail and sharper edges.

(a) Ours - FastDVDnet PSNR↑/SSIM↑/LPIPS↓ 25.66/0.73/0.39

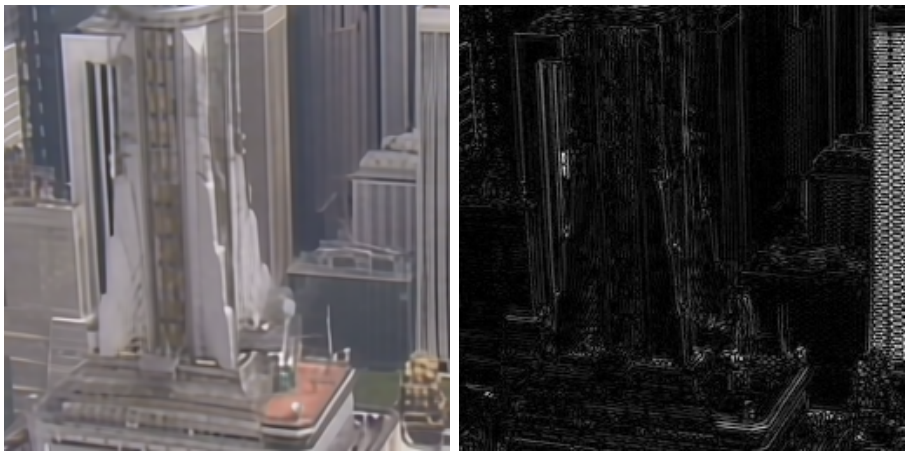
(b) BasicVSR [Cha+21] PSNR↑/SSIM↑/LPIPS↓ 22.44/0.64/0.53

(c) ground truth

Figure 3.19: Super-resolution (×4 / Gauss.($\sigma = 3.2$), $\sigma_{\boldsymbol{n}} = 2.55$) on the `aerobatics` sequence of DAVIS-2017-test-480p [Pon+17] (256 × 256 center crop of frame 10 shown here). Our method can adapt to any blur kernel and various amounts of noise, while BasicVSR simply upscales the blurry and noisy input frames.



(a) Ours - FastDVDnet PSNR↑/SSIM↑/LPIPS↓ 24.60/0.70/0.35



(b) EDVR [Wan+19] PSNR↑/SSIM↑/LPIPS↓ 23.62/0.71/0.21

(c) ground truth

Figure 3.20: Super-resolution (×4 / Gauss.($\sigma = 1.6$), $\sigma_{\boldsymbol{n}} = 0$) on the `city` sequence of Vid4 [CD14] (256 × 256 center crop of frame 26 shown here) EDVR (recall it is trained with bicubic downsampling) produces a sharper result but tends to hallucinate structure that does not match with the ground truth.

3.4 Summary

In this Chapter, we tackled the inverse problem of video restoration with the Deep Plug and Play framework. It combines a model-based data fidelity term and a regularization term parametrized by a deep denoising network in an alternate optimization scheme. We explored two main strategies for restoring degraded videos using Plug & Play algorithms: using *(i)* a state-of-the-art image denoiser (DRUNet) as a regularizer, and *(ii)* a lightweight, competitive video denoiser (FastDVDnet) as a regularizer. While the first approach is equivalent to applying a PnP algorithm separately on each frame, the second approach is explored here for the first time. Our experiments show that the second approach outperforms the first one in the vast majority of problems tested (deblurring, super-resolution, interpolation of missing pixels), both in terms of PSNR and temporal consistency. The performance delta is even larger when the degradation changes from one frame to another, as in our deblurring and pixel interpolation problems. This difference is quite remarkable, as FastDVDnet has 13× less parameters than DRUNet. Promising results for video interpolation of random missing pixels and 3D deconvolution paves the way for other video restoration problems where the degradation operator cannot be separated frame by frame, such as frame interpolation or spatio-temporal deconvolution. Another potential direction for future work is the exploration of other more complex and evolved video denoising architectures, *e.g.* other multi-frame architectures like [Sun+21a; Lia+22b], or even recurrent architectures as in [Mag+21; Lia+22a]. The evaluation of these denoisers could take multiple forms: one could study their impact on final PnP restoration performance, both in terms of per-frame quality and consistency from one frame to another; one could also study them in terms of convergence speed and stability. In future work, we also intend to provide convergence guarantees for video restoration PnP schemes under very mild conditions on the data-fitting term. A promising way to do so would be to generalize the work of [HLP22a; Pes+21; HLP22b] to video denoisers.

Conclusion

Throughout this manuscript, we have explored three computational imaging approaches that leverage the information of multiple frames to improve image and video restoration.

Part I focused on the problem of real raw burst denoising, where multiple raw images corrupted with real noise are combined to produce a single image with higher signal-to-noise ratio. Chapter 1 presented a thorough analysis and reimplementation of HDR+, a burst photography imaging pipeline featured in Google Pixel smartphones. It performs burst denoising in two main steps: multi-scale distance-based alignment of Bayer patches, and Wiener-based temporal fusion of patches in the Fourier space. Since HDR+ is a full photography pipeline, it contains many additional processing steps such as spatial denoising, HDR tone mapping using synthetic exposure fusion, and color correction. Like many other handcrafted methods, it has many tuning hyperparameters that can drastically change is output. The HDR+ pipeline is a realistic target for embedded systems given that it is a complex system made of many simple parts and that it has already been included in smartphones.

Chapter 2 tackled the raw burst denoising problem with a deep learning approach. By designing RBDnet, we successfully transformed a CNN video denoising method designed for Additive White Gaussian Noise removal [TDV20] into a burst denoising algorithm that significantly outperforms handcrafted methods such as HDR+. Its streamlined and elegant cascaded architecture with two U-Net blocks is competitive with state-of-the-art methods with significantly higher architectural or numerical complexity. It is also interesting to note that a model only trained on varied synthetic data can perform as well as, if not better (when looking at real noisy images with no ground truth), than the same model trained on real raw data captured in limited controlled scenarios. With the increasing availabilty of dedicated neural chips in embedded systems, RBDnet (or a variant of it) could be a realistic target for on-device inference in smartphones or action cameras in the first half of the 2020s.

In Part II and its singular Chapter 3, we studied an approach that bridges the gap between model-based and learning-based approaches for video restoration, namely the deep Plug-and-Play framework. We have shown that a single deep neural network trained for Gaussian denoising can be used for multiple video restoration inverse problems such as deconvolution, super-resolution or pixel interpolation. Our experiments show that using a network specifically designed for

video denoising instead of a single image denoiser improves stability and final restoration performance. While PnP methods are versatile and have an elegant mathematical background, their iterative nature and their sensitivity to input data and optimization hyperparameters do not make them an obvious choice for real-time applications and embedded systems but better suited for offline and/or off-device applications.

Circling back to compromises in photography and videography, it is clear that these three approaches each have their strengths, weaknesses and tradeoffs.

Model-based or handcrafted method are fully explainable, and they can easily be modified, augmented or fine-tuned. However, they often require painstaking hyperparameter tuning and extensive engineering and programming that might lead to technical debt in the long term.

In contrast, learning-based methods usually require less software development time and can provide giant performance leaps when compared to model-based techniques. That being said, correctly training neural networks is a skillset of its own: the process usually features many variables and hyperparameters, such as architecture design, or choice of learning rate, training data, number of iterations, optimizer hyperparameter values, and so on. Perfecting neural network training also involves a lot of trial and error, a lot of time, and a lot of computing resources, this requirement seemingly increasing at a frantic pace, which might seem at odds with the growing environmental concerns of society at large in the early 2020s. Just like deep learning models that manage to captivate the attention of the general population thanks to major technological breakthroughs or disruption such as image generation algorithms or large language models, neural networks for image and video restoration also suffer from the infamous "black box" problem. While it is possible to know what data and objective functions were used to train a network, or why its architecture was designed a certain way, it is sometimes difficult to formally explain its behaviour in all use cases, to grasp what specific representations of the data the network actually learns, or to understand what leads to some failure cases (*e.g.* the artifacts of Figure 2.10 present in only some variations of our model).

Unsurprisingly, Plug-and-Play methods carry some of the pros and cons of both model-based and learning-based algorithms. They can yield better performance than classical methods, while being more explainable than neural networks. A single framework can be applied to many restoration tasks with very little adjustments, while most neural networks need to be retrained each time the degradation changes. On the other hand, PnP methods are quite sensitive to the denoiser used and to the hyperparameters of the alternate optimization scheme. Moreover, additional robustness and convergence guarantees of the method actually require additional properties and assumptions on the denoiser than simply being a Gaussian denoising algorithm. In the Deep PnP case, it can mean having to retrain or fine-tune the denoising neural network, which entails all the trial and error and the various costs described above.

To conclude this manuscript, let us discuss opportunities for future research that may build upon the work conducted during this thesis. We could start from experiments that were initiated during it but did not make it into this document.

The architecture and denoising performance of FastDVDnet could be improved, both for a raw burst denoising version like RBDnet or as a Gaussian Plug-and-Play video denoiser. One might take a brute-force approach and simply increase the number of examples. For example, by replacing the U-Net blocks by bigger autoencoders based on DRUNet [Zha+21a] (with deeper residual blocks instead of convolution layers and 3 dowsampling stages instead of 2), it is possible to achieve +0.45dB of PSNR of video denoising on DAVIS-2017-test-480p [Pon+17], but this "FastDVDnet_DRUNet" has 36M parameters instead of FastDVDnet's 2.5M. Another possibility would be to improve communication between the features of both U-Net blocks *e.g.* by using cross-stage feature fusion similarly to [Zam+21] or channel-wise attention as in [Zam+22].

In the case of video PnP restoration, one might want to provide additional theoretical convergence guarantees by training regularized versions of FastDVDnet. As in [Pes+21; HLP22b], it is possible to add a penalty term to the training loss that enforces FastDVDnet (or its residual) to have specral norm < 1 (although the spectral norm approximation using power iterations is a bit more complex given that the network has 5 input frames and 1 output frame). Another possibility inspired by [HLP22a] would be to explicitly train FastDVDnet as a video denoising gradient step, *e.g.* by using sequences of more than 5 consecutive noisy images and automatically computing the non-zero gradients of output frames with respect to input frames in a sliding window fashion.

A last possible direction of research would be multi-image restoration of more complex or combined degradations. In addition to raw denoising, one might one want to concurrently perform Bayer to RGB demosaicking as recently explored in [Dew+23]. Real raw deblurring is another possibility; however, gathering paired blurry and sharp images with realistic, non-uniform blur is difficult, whether using real or synthetic blurry data. Realistic blurry data generation in the raw domain was actually the subject of the 6 months long masters degree intership of Yacine Bouaouni, co-supervised with Charles Laroche, also a GoPro / MAP5 PhD student. This internship has shown promising results, including generating gyroscope-guided non-uniform blurry images, and the work is ongoing in hopes of being published.

Publications

- Antoine Monod, Julie Delon, and Thomas Veit, *An Analysis and Implementation of the HDR+ Burst Denoising Method*, Image Processing On Line, 11 (2021), pp. 142–169.
- Antoine Monod, Julie Delon, Matias Tassano, and Andrés Almansa, *A priori Plug-and-Play profond pour la restauration de vidéos*, GRETSI'22
- Antoine Monod, Julie Delon, Matias Tassano, and Andrés Almansa, *Video Restoration with a Deep Plug-and-Play Prior*, arXiv:2209.02854 (preprint).

Patents

- Apparatus and methods for Plug-and-Play video deblur, patent pending

Bibliography

[ABB19] Abdelrahman Abdelhamed, Marcus Brubaker, and Michael Brown. "Noise Flow: Noise Modeling With Conditional Normalizing Flows." en. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea (South): IEEE, Oct. 2019, pp. 3165–3173. DOI: 10.1109/ICCV.2019.00326.

[AD18] Miika Aittala and Frédo Durand. "Burst Image Deblurring Using Permutation Invariant Convolutional Neural Networks." en. In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Vol. 11212. Cham: Springer International Publishing, 2018, pp. 748–764. DOI: 10.1007/978-3-030-01237-3_45.

[Ahm+20] Rizwan Ahmad et al. "Plug-and-Play Methods for Magnetic Resonance Imaging: Using Denoisers for Image Recovery." In: *IEEE Signal Processing Magazine* 37.1 (2020), pp. 105–116. DOI: 10.1109/MSP.2019.2949470.

[ALB18] Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. "A High-Quality Denoising Dataset for Smartphone Cameras." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[AM17] J. Anger and E. Meinhardt-Llopis. "Implementation of Local Fourier Burst Accumulation for Video Deblurring." In: *Image Processing On Line* 7 (2017), pp. 56–64.

[Ber18] Marcelo Bertalmío, ed. *Denoising of Photographic Images and Video: Fundamentals, Open Challenges and New Trends*. Advances in Computer Vision and Pattern Recognition. Cham: Springer International Publishing, 2018. DOI: 10.1007/978-3-319-96029-6.

[Bha+21] Goutam Bhat et al. "Deep burst super-resolution." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 9209–9218.

[BM11] Mohsen Bayati and Andrea Montanari. "The Dynamics of Message Passing on Dense Graphs, with Applications to Compressed Sensing." In: *IEEE Transactions on Information Theory* 57.2 (2011), pp. 764–785. DOI: 10.1109/TIT.2010.2094817.

[Boy+11] Stephen Boyd et al. "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers." English. In: *Foundations and Trends® in Machine Learning* 3.1 (July 2011), pp. 1–122. DOI: 10.1561/2200000016.

[BR19] Joshua Batson and Loic Royer. *Noise2Self: Blind Denoising by Self-Supervision*. June 2019.

[BRE16] Alon Brifman, Yaniv Romano, and Michael Elad. "Turning a denoiser into a super-resolver using plug and play priors." en. In: *2016 IEEE International Conference on Image Processing (ICIP)*. Phoenix, AZ, USA: IEEE, Sept. 2016, pp. 1404–1408. DOI: 10.1109/ICIP.2016.7532589.

[Bro+19] T. Brooks et al. "Unprocessing images for learned raw denoising." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11036–11045.

[BT09] Amir Beck and Marc Teboulle. "Fast Gradient-Based Algorithms for Constrained Total Variation Image Denoising and Deblurring Problems." In: *IEEE Transactions on Image Processing* 18.11 (Nov. 2009), pp. 2419–2434. DOI: 10.1109/TIP.2009.2028250.

[Bua+09] A. Buades et al. "A note on multi-image denoising." In: *International Workshop on Local and Non-Local Approximation in Image Processing*. IEEE. 2009, pp. 1–15.

[Buz+18] Gregery T. Buzzard et al. "Plug-and-Play Unplugged: Optimization-Free Reconstruction Using Consensus Equilibrium." In: *SIAM Journal on Imaging Sciences* (2018).

[Car+21] Guillermo Carbajal et al. *Non-uniform Blur Kernel Estimation via Adaptive Basis Decomposition*. Apr. 2021. DOI: 10.48550/arXiv.2102.01026.

[CB13] Miguel Colom and Antoni Buades. "Analysis and extension of the percentile method, estimating a noise curve from a single image." In: *Image Processing On Line* 3 (2013), pp. 332–359.

[CD14] Ce Liu and Deqing Sun. "On Bayesian Adaptive Video Super Resolution." en. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.2 (Feb. 2014), pp. 346–360. DOI: 10.1109/TPAMI.2013.127.

[CEM21] Regev Cohen, Michael Elad, and Peyman Milanfar. "Regularization by Denoising via Fixed-Point Projection (RED-PRO)." In: *SIAM Journal on Imaging Sciences* 14.3 (Jan. 2021), pp. 1374–1406. DOI: 10.1137/20M1337168.

[Cha+20] Ke-Chi Chang et al. "Learning camera-aware noise models." In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*. Springer. 2020, pp. 343–358.

[Cha+21] Kelvin CK Chan et al. "BasicVSR: The Search for Essential Components in Video Super-Resolution and Beyond." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2021.

[Cha+22] Kelvin C.K. Chan et al. "BasicVSR++: Improving Video Super-Resolution with Enhanced Propagation and Alignment." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2022.

[Che+18] Chen Chen et al. "Learning to See in the Dark." en. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, June 2018, pp. 3291–3300. DOI: 10.1109/CVPR.2018.00347.

[Che+21] Liangyu Chen et al. "Hinet: Half instance normalization network for image restoration." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 182–192.

[Che+22] Wei-Ting Chen et al. "Learning multiple adverse weather removal via two-stage knowledge learning and multi-contrastive regularization: Toward a unified model." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 17653–17662.

[Coh+21] Regev Cohen et al. "It Has Potential: Gradient-Driven Denoisers for Convergent Solutions to Inverse Problems." In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 18152–18164.

[CP11] Patrick L. Combettes and Jean-Christophe Pesquet. "Proximal Splitting Methods in Signal Processing." In: *Fixed-point algorithms for inverse problems in science and engineering* (2011), pp. 185–212.

[CWE16] Stanley H. Chan, Xiran Wang, and Omar A. Elgendy. "Plug-and-Play ADMM for Image Restoration: Fixed Point Convergence and Applications." In: *IEEE Transactions on Computational Imaging* 3.1 (2016), pp. 84–98.

[Dab+07] Kostadin Dabov et al. "Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering." In: *IEEE Transactions on Image Processing* 16.8 (Aug. 2007), pp. 2080–2095. DOI: 10.1109/TIP.2007.901238.

[Dab+09] K. Dabov et al. “BM3D Image Denoising with Shape-Adaptive Principal Component Analysis.” In: *SPARS'09 - Signal Processing with Adaptive Sparse Structured Representations*. Ed. by Rémi Gribonval. Inria Rennes - Bretagne Atlantique. Saint Malo, France, Apr. 2009.

[Dar+16] Yehuda Dar et al. “Postprocessing of Compressed Images via Sequential Denoising.” In: *IEEE Transactions on Image Processing* (2016).

[Dew+21] Valéry Dewil et al. *Self-Supervised training for blind multi-frame video denoising*. Apr. 2021. DOI: 10.48550/arXiv.2004.06957.

[Dew+23] Valéry Dewil et al. “Video joint denoising and demosaicing with recurrent CNNs.” In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, pp. 5108–5119.

[DH18] Julie Delon and Antoine Houdard. “Gaussian Priors for Image Denoising.” en. In: *Denoising of Photographic Images and Video*. Ed. by Marcelo Bertalmío. Cham: Springer International Publishing, 2018, pp. 125–149. DOI: 10.1007/978-3-319-96029-6_5.

[Dia+18] Steven Diamond et al. *Unrolled Optimization with Deep Priors*. 2018. arXiv: 1705.08041 [cs.CV].

[DJ95] David L. Donoho and Iain M. Johnstone. “Adapting to Unknown Smoothness via Wavelet Shrinkage.” In: *Journal of the American Statistical Association* 90.432 (Dec. 1995), pp. 1200–1224. DOI: 10.1080/01621459.1995.10476626.

[DMM09] David L. Donoho, Arian Maleki, and Andrea Montanari. “Message-passing algorithms for compressed sensing.” en. In: *Proceedings of the National Academy of Sciences* 106.45 (Nov. 2009), pp. 18914–18919. DOI: 10.1073/pnas.0909892106.

[DMP18] Alain Durmus, Éric Moulines, and Marcelo Pereyra. “Efficient Bayesian Computation by Proximal Markov Chain Monte Carlo: When Langevin Meets Moreau.” In: *SIAM Journal on Imaging Sciences* 11.1 (2018), pp. 473–506. DOI: 10.1137/16M1108340.

[Don06] D.L. Donoho. “Compressed sensing.” In: *IEEE Transactions on Information Theory* 52.4 (Apr. 2006), pp. 1289–1306. DOI: 10.1109/TIT.2006.871582.

[Dos+20] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale.” In: *arXiv preprint arXiv:2010.11929* (2020).

[DS15] M. Delbracio and G. Sapiro. “Removing Camera Shake via Weighted Fourier Burst Accumulation.” In: *IEEE Transactions on Image Processing* 24.11 (2015), pp. 3293–3307. DOI: 10.1109/TIP.2015.2442914.

[Dud+22] Akshay Dudhane et al. "Burst image restoration and enhancement." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5759–5768.

[EHN96] Heinz Werner Engl, Martin Hanke, and A. Neubauer. *Regularization of Inverse Problems*. en. Springer Science & Business Media, July 1996.

[Ehr+20] Thibaud Ehret et al. *Model-blind Video Denoising Via Frame-to-frame Training*. Feb. 2020.

[ESP21] Thomas Eboli, Jian Sun, and Jean Ponce. *Learning to Jointly Deblur, Demosaick and Denoise Raw Images*. Apr. 2021. DOI: 10.48550/arXiv.2104.06459.

[EVM21] Sophie Carneiro Esteves, Antoine Vacavant, and Odyssée Merveille. "Learning a reconnecting regularization term for blood vessel variational segmentation." In: *2021 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*. July 2021, pp. 1–4. DOI: 10.1109/BHI50953.2021.9508543.

[Foi+08] A. Foi et al. "Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data." In: *IEEE Transactions on Image Processing* 17.10 (2008), pp. 1737–1754.

[GC20] Ruturaj G. Gavaskar and Kunal N. Chaudhury. "Plug-and-play ISTA converges with kernel denoisers." In: *IEEE Signal Processing Letters* 27 (2020). DOI: 10.1109/LSP.2020.2986643.

[GMU18] Clément Godard, Kevin Matzen, and Matt Uyttendaele. "Deep Burst Denoising." en. In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Vol. 11219. Cham: Springer International Publishing, 2018, pp. 560–577. DOI: 10.1007/978-3-030-01267-0_33.

[GOW19] Davis Gilton, Greg Ongie, and Rebecca Willett. "Neumann Networks for Inverse Problems in Imaging." In: *IEEE Transactions on Computational Imaging* 6 (Jan. 2019), pp. 328–343. DOI: 10.1109/TCI.2019.2948732.

[Gu+14] Shuhang Gu et al. "Weighted nuclear norm minimization with application to image denoising." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 2862–2869.

[Gun+05] B.K. Gunturk et al. "Demosaicking: color filter array interpolation." In: *IEEE Signal processing magazine* 22.1 (2005), pp. 44–54.

[Has+16] S.W. Hasinoff et al. "Burst photography for high dynamic range and low-light imaging on mobile cameras." In: *ACM Transactions on Graphics (Proceedings SIGGRAPH Asia)* 35.6 (2016).

[He+16] Kaiming He et al. "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[Hei+14] Felix Heide et al. "FlexISP: a flexible camera image processing framework." en. In: *ACM Transactions on Graphics* 33.6 (Nov. 2014), pp. 1–13. DOI: 10.1145/2661229.2661260.

[Hes18] C. Hessel. "An Implementation of the Exposure Fusion Algorithm." In: *Image Processing On Line* 8 (2018), pp. 369–387.

[Hes19] C. Hessel. "Simulated Exposure Fusion." In: *Image Processing On Line* 9 (2019), pp. 469–482.

[HK94] G.E. Healey and R. Kondepudy. "Radiometric CCD camera calibration and noise estimation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16.3 (1994), pp. 267–276.

[HLP22a] Samuel Hurault, Arthur Leclaire, and Nicolas Papadakis. *Gradient Step Denoiser for convergent Plug-and-Play*. 2022. arXiv: 2110.03220 [cs.CV].

[HLP22b] Samuel Hurault, Arthur Leclaire, and Nicolas Papadakis. "Proximal denoiser for convergent plug-and-play optimization with nonconvex regularization." In: *arXiv preprint arXiv:2201.13256* (2022).

[HP05] K. Hirakawa and T. W. Parks. "Adaptive homogeneity-directed demosaicing algorithm." In: *IEEE Transactions on Image Processing* 14.3 (2005), pp. 360–369.

[JM13] Adel Javanmard and Andrea Montanari. "State evolution for general approximate message passing algorithms, with applications to spatial coupling." In: *Information and Inference: A Journal of the IMA* 2.2 (Dec. 2013), pp. 115–144. DOI: 10.1093/imaiai/iat004.

[JY22] Bo Ji and Angela Yao. "Multi-Scale Memory-Based Video Deblurring." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 1919–1928.

[KB21] Vahid Khorasani Ghassab and Nizar Bouguila. "Plug-and-Play video reconstruction using sparse 3D transform-domain block matching." en. In: *Machine Vision and Applications* 32.3 (Apr. 2021). DOI: 10.1007/s00138-021-01201-w.

[KBJ19] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. *Noise2Void - Learning Denoising from Single Noisy Images*. Apr. 2019. DOI: 10.48550/arXiv.1811.10980.

[KMW17] Ulugbek S. Kamilov, Hassan Mansour, and Brendt Wohlberg. "A Plug-and-Play Priors Approach for Solving Nonlinear Imaging Inverse Problems." In: *IEEE Signal Processing Letters* (2017).

[KS21] Zahra Kadkhodaie and Eero Simoncelli. "Stochastic Solutions for Linear Inverse Problems using the Prior Implicit in a Denoiser." In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 13242–13254.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012.

[KTF11] Dilip Krishnan, Terence Tay, and Rob Fergus. "Blind deconvolution using a normalized sparsity measure." In: *CVPR 2011*. June 2011, pp. 233–240. DOI: 10.1109/CVPR.2011.5995521.

[LAT22] Charles Laroche, Andrés Almansa, and Matias Tassano. *Deep Model-Based Super-Resolution with Non-uniform Blur*. Oct. 2022.

[Lau+22] Rémi Laumont et al. "Bayesian Imaging Using Plug & Play Priors: When Langevin Meets Tweedie." In: *SIAM Journal on Imaging Sciences* 15.2 (June 2022), pp. 701–737. DOI: 10.1137/21M1406349.

[Lau+23] Rémi Laumont et al. "On Maximum a Posteriori Estimation with Plug & Play Priors and Stochastic Gradient Descent." 2023.

[LBM13] M. Lebrun, A. Buades, and J-M. Morel. "A nonlocal Bayesian image denoising algorithm." In: *SIAM Journal on Imaging Sciences* 6.3 (2013), pp. 1665–1688.

[LeC+89] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition." In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[Leh+18] Jaakko Lehtinen et al. *Noise2Noise: Learning Image Restoration without Clean Data*. Oct. 2018. DOI: 10.48550/arXiv.1803.04189.

[Lev+09] Anat Levin et al. "Understanding and evaluating blind deconvolution algorithms." In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. June 2009, pp. 1964–1971. DOI: 10.1109/CVPR.2009.5206815.

[Li+22a] Boyun Li et al. "All-In-One Image Restoration for Unknown Corruption." In: *IEEE Conference on Computer Vision and Pattern Recognition*. New Orleans, LA, June 2022.

[Li+22b] Yawei Li et al. "NTIRE 2022 challenge on efficient super-resolution: Methods and results." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 1062–1102.

[Lia+21] Jingyun Liang et al. *SwinIR: Image Restoration Using Swin Transformer*. Aug. 2021. DOI: 10.48550/arXiv.2108.10257.

[Lia+22a] Jingyun Liang et al. *Recurrent Video Restoration Transformer with Guided Deformable Attention*. Tech. rep. arXiv:2206.02146. arXiv, June 2022.

[Lia+22b] Jingyun Liang et al. *VRT: A Video Restoration Transformer*. Tech. rep. arXiv:2201.12288. arXiv, June 2022.

[Lib+19] O. Liba et al. “Handheld mobile photography in very low light.” In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–16.

[Lim+17] Bee Lim et al. “Enhanced deep residual networks for single image super-resolution.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017, pp. 136–144.

[Liu+20] Jiaming Liu et al. “RARE: Image Reconstruction Using Deep Priors Learned Without Groundtruth.” In: *IEEE Journal of Selected Topics in Signal Processing* 14.6 (Oct. 2020), pp. 1088–1099. DOI: 10.1109/JSTSP.2020.2998402.

[Liu+21] Jiaming Liu et al. “Recovery Analysis for Plug-and-Play Priors using the Restricted Eigenvalue Condition.” In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 5921–5933.

[Liu+22] Renhao Liu et al. *Recovery of Continuous 3D Refractive Index Maps from Discrete Intensity-Only Measurements using Neural Fields*. Aug. 2022. DOI: 10.48550/arXiv.2112.00002.

[LM13] Cécile Louchet and Lionel Moisan. “Posterior Expectation of the Total Variation Model: Properties and Experiments.” In: *SIAM Journal on Imaging Sciences* 6.4 (Jan. 2013), pp. 2640–2684. DOI: 10.1137/120902276.

[Lug+20] Andreas Lugmayr et al. “Srflow: Learning the super-resolution space with normalizing flow.” In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*. Springer. 2020, pp. 715–732.

[Mag+21] Matteo Maggioni et al. “Efficient Multi-Stage Video Denoising with Recurrent Spatio-Temporal Fusion.” en. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, June 2021, pp. 3465–3474. DOI: 10.1109/CVPR46437.2021.00347.

[Mal+22] Ali Maleky et al. “Noise2NoiseFlow: Realistic Camera Noise Modeling without Clean Images.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 17632–17641.

[MDT22] Antoine Monod, Julie Delon, and Matias Tassano. “A priori Plug-and-Play profond pour la restauration de vidéos.” In: *28° Colloque sur le traitement du signal et des images*. GRETSI - Groupe de Recherche en Traitement du Signal et des Images, Sept. 2022, pp. 365–368.

[MDV21] Antoine Monod, Julie Delon, and Thomas Veit. “An Analysis and Implementation of the HDR+ Burst Denoising Method.” In: *Image Processing On Line* 11 (2021), pp. 142–169.

[Mei+17] Tim Meinhardt et al. "Learning proximal operators: Using denoising networks for regularizing inverse imaging problems." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1781–1790.

[Men+22] Lingchen Meng et al. "Adavit: Adaptive vision transformers for efficient image recognition." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12309–12318.

[MF12] Markku Makitalo and Alessandro Foi. "Optimal inversion of the generalized Anscombe transformation for Poisson-Gaussian noise." In: *IEEE transactions on image processing* 22.1 (2012), pp. 91–103.

[Mil+18] B. Mildenhall et al. "Burst denoising with kernel prediction networks." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 2502–2510.

[Mil+20] Ben Mildenhall et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. Aug. 2020. DOI: 10.48550/arXiv.2003.08934.

[Miy+18] Takeru Miyato et al. *Spectral Normalization for Generative Adversarial Networks*. en. Feb. 2018.

[Miy+61] Koichi Miyasawa et al. "An empirical Bayes estimator of the mean of a normal population." In: *Bull. Inst. Internat. Statist* 38.181-188 (1961), pp. 1–2.

[MKV09] Tom Mertens, Jan Kautz, and Frank Van Reeth. "Exposure fusion: A simple and practical alternative to high dynamic range photography." In: *Computer graphics forum*. Vol. 28. 1. Wiley Online Library. 2009, pp. 161–171.

[MLE21] Vishal Monga, Yuelong Li, and Yonina C. Eldar. "Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing." In: *IEEE Signal Processing Magazine* 38.2 (Mar. 2021), pp. 18–44. DOI: 10.1109/MSP.2020.3016905.

[Moh+20] Sreyas Mohan et al. "Robust And Interpretable Blind Image Denoising Via Bias-Free Convolutional Neural Networks." In: *International Conference on Learning Representations*. 2020.

[Mon+22a] Kristina Monakhova et al. "Dancing Under the Stars: Video Denoising in Starlight." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 16241–16251.

[Mon+22b] Antoine Monod et al. "Video Restoration with a Deep Plug-and-Play Prior." In: *arXiv preprint arXiv:2209.02854* (2022).

[Nah+19] Seungjun Nah et al. “NTIRE 2019 Challenge on Video Deblurring and Super-Resolution: Dataset and Study.” In: *CVPR Workshops*. June 2019.

[NKL17] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. “Deep Multi-Scale Convolutional Neural Network for Dynamic Scene Deblurring.” In: *CVPR*. July 2017.

[Pes+21] Jean-Christophe Pesquet et al. *Learning Maximally Monotone Operators for Image Recovery*. Apr. 2021. DOI: 10.48550/arXiv.2012.13247.

[Pon+17] Jordi Pont-Tuset et al. “The 2017 DAVIS Challenge on Video Object Segmentation.” In: *arXiv:1704.00675* (2017).

[PR17] Tobias Plotz and Stefan Roth. “Benchmarking Denoising Algorithms with Real Photographs.” en. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, July 2017, pp. 2750–2759. DOI: 10.1109/CVPR.2017.294.

[PTK22] Naama Pearl, Tali Treibitz, and Simon Korman. *NAN: Noise-Aware NeRFs for Burst-Denoising*. Apr. 2022. DOI: 10.48550/arXiv.2204.04668.

[RA16] M. Rakhshanfar and M. A. Amer. “Estimation of Gaussian, Poissonian-Gaussian, and Processed Visual Noise and Its Level Function.” In: *IEEE Transactions on Image Processing* 25.9 (2016), pp. 4172–4185.

[REM17] Yaniv Romano, Michael Elad, and Peyman Milanfar. *The Little Engine that Could: Regularization by Denoising (RED)*. Sept. 2017. DOI: 10.48550/arXiv.1611.02862.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation.” In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[RGE16] Arie Rond, Raja Giryes, and Michael Elad. “Poisson inverse problems by the Plug-and-Play scheme.” en. In: *Journal of Visual Communication and Image Representation* 41 (Nov. 2016), pp. 96–108. DOI: 10.1016/j.jvcir.2016.09.009.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors.” In: *nature* 323.6088 (1986), pp. 533–536.

[Rim+20] Jaesung Rim et al. “Real-World Blur Dataset for Learning and Benchmarking Deblurring Algorithms.” In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.

[ROF92] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms." en. In: *Physica D: Nonlinear Phenomena* 60.1 (Nov. 1992), pp. 259–268. DOI: 10.1016/0167-2789(92)90242-F.

[Ryu+19] Ernest Ryu et al. "Plug-and-Play Methods Provably Converge with Properly Trained Denoisers." In: *PMLR*. 2019.

[SC20] Wanjie Sun and Zhenzhong Chen. "Learned image downscaling for upscaling using content adaptive resampler." In: *IEEE Transactions on Image Processing* 29 (2020), pp. 4027–4040.

[Sha+22] Wentao Shangguan et al. *Learning Cross-Video Neural Representations for High-Quality Frame Interpolation*. Feb. 2022. DOI: 10.48550/arXiv.2203.00137.

[She+21] Dev Yashpal Sheth et al. "Unsupervised Deep Video Denoising." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 1759–1768.

[Shi+22] Zhihao Shi et al. "Video frame interpolation transformer." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 17482–17491.

[Son+21] Sanghyun Son et al. "NTIRE 2021 challenge on video super-resolution." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 166–181.

[Sre+16] Suhas Sreehari et al. "Plug-and-Play Priors for Bright Field Electron Tomography and Sparse Interpolation." In: *IEEE Transactions on Computational Imaging* 2.4 (Dec. 2016), pp. 408–423. DOI: 10.1109/TCI.2016.2599778.

[Sun+21a] Lu Sun et al. "Deep Maximum a Posterior Estimator for Video Denoising." In: *International Journal of Computer Vision* 129.10 (2021), pp. 2827–2845. DOI: 10.1007/s11263-021-01510-7.

[Sun+21b] Yu Sun et al. *CoIL: Coordinate-based Internal Learning for Imaging Inverse Problems*. Feb. 2021. DOI: 10.48550/arXiv.2102.05181.

[SWK19] Yu Sun, Brendt Wohlberg, and Ulugbek S. Kamilov. "An Online Plug-and-Play Algorithm for Regularized Image Reconstruction." In: *IEEE Transactions on Computational Imaging* 5.3 (Sept. 2019), pp. 395–408. DOI: 10.1109/TCI.2019.2893568.

[SZ14] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." In: *arXiv preprint arXiv:1409.1556* (2014).

[Tan+22] Yehui Tang et al. "Patch slimming for efficient vision transformers." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12165–12174.

[Tao+18] Xin Tao et al. "Scale-Recurrent Network for Deep Image Deblurring." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[TDV19a] Matias Tassano, Julie Delon, and Thomas Veit. "An Analysis and Implementation of the FFDNet Image Denoising Method." en. In: *Image Processing On Line* 9 (Jan. 2019), pp. 1–25. DOI: 10.5201/ipol.2019.231.

[TDV19b] Matias Tassano, Julie Delon, and Thomas Veit. "Dvdnet: A fast network for deep video denoising." In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 1805–1809.

[TDV20] Matias Tassano, Julie Delon, and Thomas Veit. "FastDVDnet: Towards Real-Time Deep Video Denoising Without Flow Estimation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[UVL18] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Deep image prior." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9446–9454.

[Vas+17] Ashish Vaswani et al. "Attention is All you Need." In: *Advances in neural information processing systems* 30 (2017).

[VBW13] Singanallur V. Venkatakrishnan, Charles A. Bouman, and Brendt Wohlberg. "Plug-and-Play priors for model based reconstruction." In: *GlobalSIP*. 2013. DOI: 10.1109/GlobalSIP.2013.6737048.

[VYP22] Jeya Maria Jose Valanarasu, Rajeev Yasarla, and Vishal M Patel. "Transweather: Transformer-based restoration of images degraded by adverse weather conditions." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 2353–2363.

[Wan+18] Xintao Wang et al. "ESRGAN: Enhanced super-resolution generative adversarial networks." In: *The European Conference on Computer Vision Workshops (ECCVW)*. Sept. 2018.

[Wan+19] Xintao Wang et al. "EDVR: Video restoration with enhanced deformable convolutional networks." In: *The IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. June 2019.

[Wan+20] Yuzhi Wang et al. "Practical deep raw image denoising on mobile devices." In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI*. Springer. 2020, pp. 1–16.

[WC17] Xiran Wang and Stanley H. Chan. "Parameter-free Plug-and-Play ADMM for image restoration." In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[Wei+13] Philippe Weinzaepfel et al. "DeepFlow: Large displacement optical flow with deep matching." In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 1385–1392.

[Wei+20] Kaixuan Wei et al. "A Physics-Based Noise Formation Model for Extreme Low-Light Raw Denoising." en. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 2755–2764. DOI: 10.1109/CVPR42600.2020.00283.

[WT22] Jue Wang and Lorenzo Torresani. "Deformable video transformer." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 14053–14062.

[XQC21] Yazhou Xing, Zian Qian, and Qifeng Chen. "Invertible Image Signal Processing." en. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, June 2021, pp. 6283–6292. DOI: 10.1109/CVPR46437.2021.00622.

[Xu+20] Xiaojian Xu et al. "Provable Convergence of Plug-and-Play Priors With MMSE Denoisers." In: *IEEE Signal Processing Letters* 27 (2020), pp. 1280–1284. DOI: 10.1109/LSP.2020.3006390.

[Xu+22] Xiaogang Xu et al. "PVDD: A Practical Video Denoising Dataset with Real-World Dynamic Scenes." In: *arXiv preprint arXiv:2207.01356* (2022).

[Xue+19] Tianfan Xue et al. "Video Enhancement with Task-Oriented Flow." In: *International Journal of Computer Vision (IJCV)* 127.8 (2019), pp. 1106–1125.

[Yan+20] Fuzhi Yang et al. "Learning Texture Transformer Network for Image Super-Resolution." en. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 5790–5799. DOI: 10.1109/CVPR42600.2020.00583.

[Yin+22] Hongxu Yin et al. "A-vit: Adaptive tokens for efficient vision transformer." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10809–10818.

[Yoo+21] Youngho Yoon et al. *SphereSR: 360{\deg} Image Super-Resolution with Arbitrary Projection via Continuous Spherical Image Representation*. Dec. 2021. DOI: 10.48550/arXiv.2112.06536.

[Yua+21] Xin Yuan et al. "Plug-and-play algorithms for video snapshot compressive imaging." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.10 (2021), pp. 7093–7111.

[Yue+20] Huanjing Yue et al. "Supervised Raw Video Denoising with a Benchmark Dataset on Dynamic Scenes." In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020.

[Zam+20] Syed Waqas Zamir et al. “Cycleisp: Real image restoration via improved data synthesis.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2696–2705.

[Zam+21] Syed Waqas Zamir et al. “Multi-Stage Progressive Image Restoration.” en. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, June 2021, pp. 14816–14826. DOI: 10.1109/CVPR46437.2021.01458.

[Zam+22] Syed Waqas Zamir et al. *Restormer: Efficient Transformer for High-Resolution Image Restoration*. Mar. 2022. DOI: 10.48550/arXiv.2111.09881.

[Zha+16] Ningning Zhao et al. “Fast Single Image Super-Resolution Using a New Analytical Solution for $_2 - _2$ Problems.” In: *IEEE Transactions on Image Processing* 25.8 (Aug. 2016), pp. 3683–3697. DOI: 10.1109/TIP.2016.2567075.

[Zha+17a] Kai Zhang et al. “Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising.” en. In: *TIP* 26.7 (July 2017). DOI: 10.1109/TIP.2017.2662206.

[Zha+17b] Hang Zhao et al. “Loss Functions for Image Restoration With Neural Networks.” en. In: *IEEE Transactions on Computational Imaging* 3.1 (Mar. 2017), pp. 47–57. DOI: 10.1109/TCI.2016.2644865.

[Zha+18] Yulun Zhang et al. “Image Super-Resolution Using Very Deep Residual Channel Attention Networks.” en. In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Vol. 11211. Cham: Springer International Publishing, 2018, pp. 294–310. DOI: 10.1007/978-3-030-01234-2_18.

[Zha+21a] Kai Zhang et al. “Plug-and-Play Image Restoration with Deep Denoiser Prior.” In: *TPAMI* (2021). DOI: 10.1109/TPAMI.2021.3088914.

[Zha+21b] Yi Zhang et al. “Rethinking Noise Synthesis and Modeling in Raw Denoising.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 4593–4601.

[Zha+22] Jinnian Zhang et al. “Minivit: Compressing vision transformers with weight multiplexing.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12145–12154.

[ZVT20] Kai Zhang, Luc Van Gool, and Radu Timofte. “Deep Unfolding Network for Image Super-Resolution.” en. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, June 2020, pp. 3214–3223. DOI: 10.1109/CVPR42600.2020.00328.

[ZW11] Daniel Zoran and Yair Weiss. "From learning models of natural image patches to whole image restoration." en. In: *2011 International Conference on Computer Vision*. Barcelona, Spain: IEEE, Nov. 2011, pp. 479–486. DOI: 10.1109/ICCV.2011.6126278.

[ZZZ18] K. Zhang, W. Zuo, and L. Zhang. "FFDNet: Toward a fast and flexible solution for CNN-based image denoising." In: *IEEE Transactions on Image Processing* 27.9 (2018), pp. 4608–4622.