Détection de falsification d'images via l'analyse du démosaïquage : dévoilement d'une signature

Quentin Bammey

université
PARIS-SACLAY

Image Forgery Detection through Demosaicing Analysis:
Unconcealment of a signature

Détection de falsification d'images via l'analyse du démosaïquage : dévoilement d'une signature

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 574, École Doctorale de Mathématiques Hadamard (EDMH)
Spécialité de doctorat : Mathématiques appliquées et applications des mathématiques
Unité de recherche : Université Paris-Saclay, CNRS, ENS Paris-Saclay, Centre Borelli, 91190, Gif-sur-Yvette, France
Référent : ENS Paris-Saclay

Thèse présentée et soutenue à Paris-Saclay, le 6 décembre 2021, par

Quentin Bammey

Composition du jury:

Mathilde Mougeot Mathilde Mougeot, Professeur des université, ENSIIE, ENS Paris-Saclay	Présidente
Luisa Verdoliva Professeur adjoint, Università degli Studi di Napoli Federico II (Italie)	Rapporteur
Florent Retraint Professeur des universités, Université de Technologie de Troyes	Rapporteur
Denis Perraud Docteur ingénieur chef de police technique et scientifique, Police Scientifique de Lyon	Examinateur
Rafael Grompone von Gioi Directeur de recherche associé, Centre Borelli, ENS Paris-Saclay	Directeur
Jean-Michel Morel Professeur des universités, Centre Borelli, ENS Paris-Saclay	Codirecteur

Thèse de doctorat

NNT: 2021UPASM040

A man was journeying in the wilderness and he found Truth standing there all alone. He said to her, 'Ancient lady, why do you dwell here in the wilderness, leaving the city behind?' From the great depths of her wisdom, Truth replied, 'Among the people of old, lies were found among only a few, but now they have spread throughout all of human society!'

Aesop's Fables (Babrius 126 = Perry 355), translation Laura Gibbs.

Abstract

Once considered reliable evidence, photographic images can no longer be assumed to depict the naked truth. With the advent of digital photography and the progress of photo editing tools, altering a picture has never been easier. While most of these modifications solely seek to enhance the image, they can potentially alter its very semantics. Concealing, modifying or adding a foreign object, all those can give an image a new and false meaning. Although these forgeries can easily be made visually realistic, they still distort the very fabric of the image. The formation of a digital image, from the camera sensors to storage, leaves traces, which act like a signature for the image. Modifying an image distorts these traces, creating detectable inconsistencies.

Raw images are initially a mosaic of red, blue and green pixels. Missing colour values must be interpolated in a process known as demosaicing. In this thesis, we study the traces left by this process. The 2-periodic nature of the mosaic pattern leaves its imprint onto the image. Forgeries may dephase these traces, or even remove them entirely; mosaic pattern identification is consequently helpful in localizing tampered regions.

Non-specific forgery detection methods can already analyse many traces in an image; nevertheless they remain blind to shifts in the mosaic, due to the translation-invariance of the convolutional neural networks on which most are based. Demosaicing-specific methods can thus provide complementary results for forgery detections. However, these have historically received little attention. Analysis of demosaicing artefacts is made harder by the vast array of often-undisclosed demosaicing algorithms, and above all by JPEG compression. Those artefacts, created early in the image formation pipeline and lying at the highest frequencies of the image, are quick to wane during compression.

Yet, those artefacts can still be detected under mild compression. To channel the representative power of convolutional neural networks into the analysis of demosaicing artefacts, we introduce the notion of positional training. This self-supervised scheme trains the network to detect the modulo-2 position of each pixel, leveraging the translation invariance of convolution to make the network implicitly analyse demosaicing artefacts, its only clue to the modulo-2 position of a pixel. On top of that, internal training on a single potentially forged image can bolster the method's robustness to JPEG compression on said image. Errors in the output of the neural network are then clues of mosaic inconsistencies. An a contrario paradigm then enables us to make automatic decisions on the authenticity of an image. Using only demosaicing artefacts, the proposed method beats the state of the art on several uncompressed datasets. On compressed images, it still provides decent results that are fully complementary with methods that are not mosaic-specific.

Finally, we explore the very evaluation of forgery detection methods. We propose a methodology and dataset to study the sensitivity of forensic tools to specific traces, as well as their ability to make detections without semantic cues on the image. More than a simple evaluation tool, this methodology can be used to assess the strength and weaknesses of each method, as well as their complementarities.

Remerciements

Ces années de thèse ont été pour moi une période privilégiée, durant lesquelles j'ai bénéficié d'excellentes conditions pour me consacrer à la recherche. Je suis sincèrement reconnaissant envers tous ceux qui ont rendu cela possible, en premier lieu Rafael Grompone et Jean-Michel Morel, mes directeurs de thèse, dont le soutien a été indéfectible et qui tout en me laissant une grande liberté ont su me guider et m'aider à ordonner mes idées chaque fois que cela était nécessaire. Je souhaite aussi remercier les membres du jury de ma thèse, Mathilde Mougeot, Luisa Verdoliva, Florent Retraint et Denis Perraud, d'avoir accepté cette tâche et consacré leur temps à ma thèse. Un grand merci en particulier à Luisa Verdoliva et Florent Retraint, rapporteurs de ma thèse. Je souhaiterais ensuite remercier la Direction Générale de l'Armement (DGA) pour avoir financé ma thèse. Ma thèse n'aurait pu se faire dans de telle conditions sans l'aide constante de notre équipe administrative, c'est donc tout naturellement que je tiens à remercier Véronique Almadovar, Alina Müller, Gwladys Stouvenel et Soraya Zarroug. Je remercie également Agnès Desolneux, qui en tant que codirectrice de l'École Doctorale de Mathématiques Hadamard donne beaucoup de son temps à guider les doctorants dans la réalisation de leur thèse. Ma gratitude va également à Hans Henrik Rugh, grâce à qui j'ai découvert l'enseignement durant ma thèse.

Bien évidemment, un grand merci à ma famille, pour m'avoir poussé et soutenu dans mon éducation toutes ces années, tout en me laissant libre de mes choix.

Un grand remerciement aussi à tous les enseignants et professeurs qui m'ont accompagné, de mon enfance jusqu'à maintenant, qui auront su captiver mon intérêt pour des sujets scientifiques ou non, et auront *in fine* forgé mon esprit et mon caractère. Je pense en particulier à Laurence et Laurent Maginelle, Alexandra Morales, Elisabeth Depoers, Marie-Bernadette Gilliers, Pascal Ferroni, Michel Cognet, Jean-Michel Ferrard, Thomas Caniard, Denis Trarieux, Daniel Hirschkoff, Yves Robert, Ines Funke, Véronique Rancurel, Beate Langenbruch, Éric Thierry ; et bien sûr à Jean-Michel Morel. J'en oublie certainement – veuillez par avance m'en excuser –, ils sont bien entendu tous remerciés.

Au cours de mes études, j'ai eu l'occasion de faire plusieurs stages, tous très enrichissants. Je remercie de tout cœur mes encadrants pour le temps passé à mes côtés, et grâce à qui j'ai pu découvrir la recherche bien avant de commencer mon doctorat. Merci donc à Márton Karsai et Éric Fleury à l'ENS Lyon ; Markus Rempfler et

Björn Menze à la Technische Universität München ; et enfin Éric Chassande-Mottin (Université de Paris), Aurelia Fraysse (CentraleSupélec), Stéphane Jaffard (Université Paris-Est) et Yves Meyer (ENS Paris-Saclay).

Revenons maintenant au laboratoire. Du CMLA au Centre Borelli, de Cachan à Gif-sur-Yvette, j'ai eu la chance de préparer ma thèse dans une ambiance unique et inoubliable, que le contexte sanitaire n'aura pu ébranler. Merci infiniment à tous ceux qui ont fait ou continuent de faire de ce laboratoire un véritable lieu de vie. La vie du laboratoire est intrinsèquement liée à ses fréquents séminaires, pour l'organisation desquels je souhaite remercier Axel, Tina, Marina, Roger, Anthea, Argyris, Alejandro, Sylvain et Laurent. Tina, pour ton amitié constante depuis notre arrivée. Merci aussi pour tes photos de Pashmina, qui parsèment ma thèse et nos articles, et nos fréquentes collaborations, pour lesquelles je souhaite également remercier Marina et Miguel. Yanhao, pour ta compagnie quasi-quotidienne. Anthea, Élodie, Khoa, Valéry, pour votre compagnie et nos – trop courtes ! – pauses café. Enric, Gabriele, Axel, Jeremy, Charles, Martina, Thibaud, Adrien, pour nos discussions enrichissantes – Jeremy, merci aussi pour vpv ! Anne, enfin, c'est grâce à ton contagieux enthousiasme que j'ai découvert le taekwondo ; tu as pour cela toute ma gratitude. Plus généralement, je voudrais remercier tous les membres et anciens membres du Centre Borelli que je n'ai pas déjà cité, et qui contribuent au quotidien à faire du laboratoire un espace stimulant pour la recherche.

Sortant du laboratoire, je souhaite remercier Samantha, pour ton amitié continue, malgré la distance ; et le Taekwondo Moudok Kwan pour leur accueil et l'ambiance qui y règne, en particulier Maître Yoon-Soo Choi, Yiyi, Louise, Lely, Thomas, Anne, Julien, Iliana, et Élise.

Pour tous ces bons moments passés et à venir, merci !
Quentin Bammey

Contents

Résumé

Internet, les médias numériques, les nouveaux moyens de communication et les réseaux sociaux ont favorisé l'émergence d'un monde connecté où la maîtrise parfaite de l'information devient impossible. Les images sont omniprésentes et sont donc devenues un élément essentiel de l'actualité. Malheureusement, elles sont aussi devenues un outil de désinformation visant à détourner le public de la réalité.

La manipulation d'images est partout. Le simple fait d'enlever les yeux rouges des photos de famille pourrait déjà être qualifié de manipulation d'images, alors qu'il s'agit simplement de rendre une photographie plus naturelle. Même les amateurs peuvent facilement effacer les câbles électriques d'un panorama de vacances, corriger les imperfections physiques telles que les rides d'un visage, sans parler des retouches effectuées sur les mannequins dans les magazines.

Au-delà de ces exemples plutôt bénins, la manipulation d'images peut conduire à des résultats falsifiés dans des publications scientifiques, des rapports ou des articles journalistiques. Les images modifiées peuvent changer de sens et donc être utilisées comme fausses preuves, par exemple en diffamation ou pour prétendre à un phénomène paranormal. Plus fréquemment, des images falsifiées sont publiées et relayées sur les médias sociaux, afin de créer et de contribuer à la diffusion de *fake news*.

Fabriquer une contrefaçon visuellement convaincante est désormais à la portée de tous ; ces falsifications peuvent ensuite être diffusées sur des médias en ligne ou des réseaux sociaux [1], falsifier les résultats d'études scientifiques ou être présentées comme de fausses preuves dans un procès.

Récemment, les réseaux de neurones ont permis de générer des images manipulées de manière presque automatique, comme le site This Person Does Not Exist[1] qui génère de manière aléatoire des visages réalistes de personnes qui n'existent pas. Les méthodes de *deepfake* permettent entre autres de remplacer un visage dans une vidéo par celui d'une autre personne (*face swapping*).

Ces nouvelles possibilités de manipulation d'images sont exploitées depuis longtemps par les gouvernements, les organisations criminelles et les délinquants. On peut penser aux images de propagande stalinienne, dans lesquelles certains personnages devenus indésirables étaient retirés des photographies officielles. (Figure 0.1).

Aujourd'hui, la manipulation d'images peut servir les intérêts d'organisations criminelles ou terroristes dans le cadre de leur propagande (fausses revendications, faux événements, masquage d'éléments d'identification, ajout d'objets). Les techniques de face swapping et de deepfake sont également un moyen simple de porter atteinte à l'image et à la vie privée de personnalités publiques en les plaçant sur des photos compromettantes. La manipulation d'images est également un moyen

[1] www.thispersondoesnotexist.com.

FIGURE 0.1 : Un exemple montrant comment une image a été modifiée plusieurs fois de suite, chaque personne perdant les faveurs du régime voyant son image supprimée de la photo. Seul Joseph Staline apparaît sur les quatre photos.

d'exercer une coercition, une pression ou un chantage à l'encontre d'un tiers. Les images manipulées peuvent également être utilisées pour nuire aux entreprises par le biais de campagnes de désinformation. Les documents administratifs peuvent être falsifiés afin d'obtenir des papiers officiels, un document de location ou un prêt auprès d'organismes spécialisés. Le morphing de visage, dont l'objectif est d'obtenir la photo d'un visage visuellement "compatible" à partir de deux visages, permet à deux utilisateurs de partager la même identité afin de tromper un contrôle d'identité. Ces manipulations posent également des problèmes aux forces de l'ordre. Par le passé, les aveux, les témoignages ou les photographies suffisaient à prouver la culpabilité. Les technologies de falsification n'étaient pas suffisamment développées pour tromper les enquêteurs. Aujourd'hui, ces méthodes ne suffisent plus et les forces de l'ordre ont besoin d'outils scientifiques innovants pour pouvoir présenter des preuves fiables devant les tribunaux.

L'image numérique est un moyen de communication essentiel dans le monde d'aujourd'hui. Les gens doivent pouvoir faire confiance à cette méthode de communication. Il est donc essentiel de pouvoir détecter les images qui ont été manipulées.

Pourtant, même si les images sont faciles à modifier de manière visuellement réaliste, ces modifications peuvent être difficiles à détecter automatiquement.

Dans la mythologie grecque, Dolus, l'esprit de la ruse, a tenté de reproduire une statue d'Aletheia, déesse de la vérité ; ce faisant, il a manqué d'argile et a laissé ses pieds inachevés. Lorsque Prométhée, le maître de Dolus, a donné vie aux deux statues, la fausse statue s'est révélée incapable de marcher aussi bien que l'originale ; la contrefaçon a alors été révélée. Dans les mots d'Ésope (Traduction Henri Tournier) :

> Ce modeleur d'un nouveau siècle, Prométhée,
> Façonna d'une fine argile Vérité,
> Pour que, chez les humains, elle rendît justice.
> Soudain mandé de Zeus par messager spécial,

Il confie l'atelier à la Ruse trompeuse
Qu'il avait récemment prise comme apprentie.
Elle, d'un zèle ardent, façonne une statue
– Même visage et proportions, en tout semblable –
De ses habiles mains et pendant son absence.
Elle avait presque terminé cette merveille
Quand pour finir les pieds, lui manqua de l'argile.
Mais le maître revient ; la Ruse, en grande hâte,
Tremblant de peur, retourne à sa place s'asseoir.
Prométhée, étonné de tant de ressemblance,
Veut démontrer alors la grandeur de son œuvre.
Aussi met-il ensemble au four les deux statues ;
Après cuisson, il donne vie à l'une et l'autre ;
D'un pas modeste va la sainte Vérité ;
Mais l'effigie tronquée reste clouée au sol.
Lors cette fausse image, ouvrage clandestin,
Prit le nom de Mensonge ; à cette affirmation
Qu'il est privé de pieds, je souscris volontiers.

De la même manière que la fausse statue avait des traces imparfaites qui ont rendue son identification possible, les falsifications d'images laissent généralement des traces. Depuis la scène réelle dont la photographie est prise jusqu'au stockage de l'image capturée sur un support numérique, de nombreuses opérations ont lieu pour créer l'image finale, chacune imprimant ses traces sur l'image. L'ensemble de ces traces constitue une véritable signature de l'image, à la manière d'un filigrane naturel. Bien qu'habituellement imperceptible à l'œil nu, cette signature peut généralement être détectée et analysée, ce qui permet de reconstituer l'historique d'une image, de modéliser les différentes opérations qui ont eu lieu lors de la création de l'image, ainsi que leur ordre et leurs paramètres. Les informations sur la chaîne de traitement spécifique à l'appareil photo sont pertinentes en soi, non seulement parce qu'elles peuvent guider la restauration de l'image, mais surtout car elles constituent une signature identifiant l'image.

En effet, lorsqu'une image est manipulée, sa signature est perturbée. Un modèle de chaîne de traitement localement incohérent sur l'ensemble de l'image est donc souvent un indice que l'image a été trafiquée.

0.1 Chaîne de formation de l'image

Les principales étapes du processus d'acquisition d'images numériques, illustrées dans la Figure 0.2, seront brièvement décrites dans cette section. D'autres étapes importantes, comme le débruitage, dépassent le cadre de cette thèse et ne seront donc pas abordées ici.

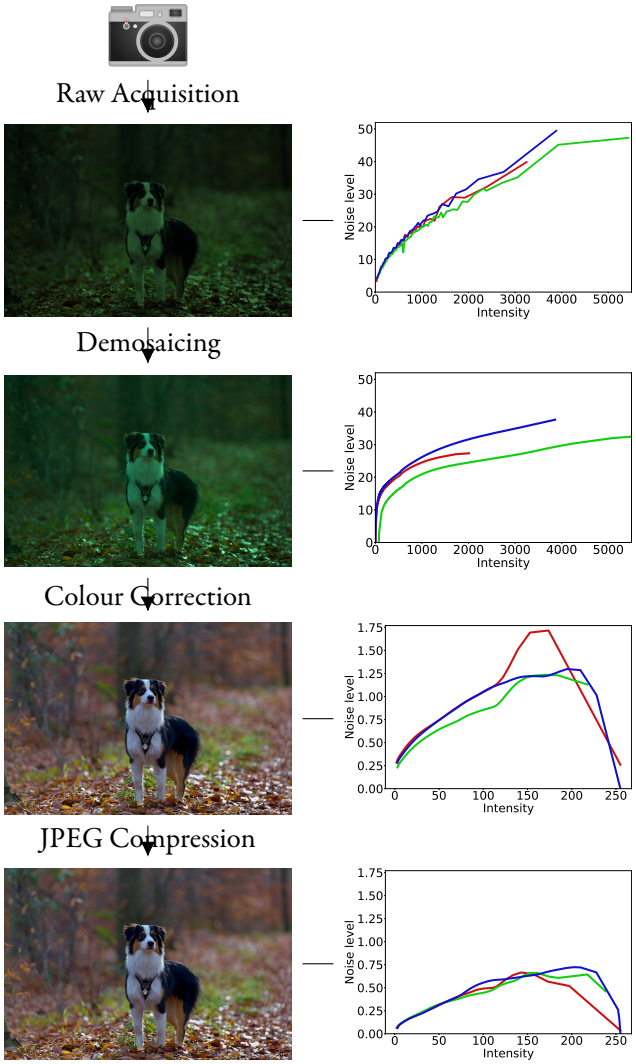


FIGURE 0.2 : Chaîne de traitement simplifiée d'une image, de son acquisition par le capteur de l'appareil jusqu'à son stockage sous forme d'image compressée en JPEG. La colonne de gauche représente l'image à chaque étape. La colonne de droite représente le bruit de l'image en fonction de l'intensité dans les trois canaux (rouge, vert, bleu). Comme chaque étape laisse une empreinte spécifique sur le modèle de bruit de l'image, l'analyse de ce bruit nous permet d'effectuer une rétro-ingénierie de la chaîne de traitement d'une image, ce afin de détecter les régions d'une image qui ont été traitées différemment, et qui sont donc susceptibles d'avoir été falsifiées.

Acquisition de l'image brute

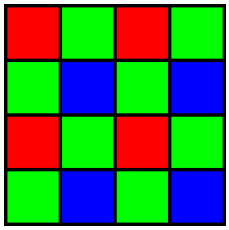
La première étape pour acquérir une image brute consiste à compter le nombre de photons incidents sur le capteur sur la durée de l'exposition. Les capteurs utilisent des dispositifs à couplage de charge (CCD) ou des semi-conducteurs à oxyde métallique complémentaire (CMOS). Bien que les deux technologies reposent sur des principes de fonctionnement différents, elles peuvent être modélisés de manière très similaire [2]. Les capteurs transforment les photons lumineux entrants en charge

électronique qui interagit avec les dispositifs de détection pour produire des électrons stockés dans un puits de lumière potentiel. Lorsque ce dernier est plein, les pixels sont saturés. L'étape finale consiste à convertir les mesures de tension analogiques en valeurs numériques quantifiées.

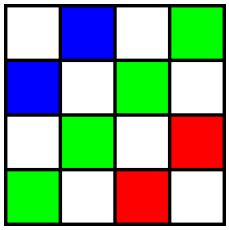
Démosaïquage

La plupart des appareils photo ne peuvent pas voir la couleur directement, car chaque pixel est obtenu par un seul capteur qui ne peut compter que le nombre de photons qui l'atteignent dans une certaine gamme de longueurs d'onde. Afin d'obtenir une image en couleur, une matrice de filtres colorés (CFA, de l'anglais *Colour Filer Array*) est placé devant les capteurs. Chacun d'eux ne compte que les photons d'une certaine longueur d'onde. Par conséquent, chaque pixel a une valeur relative à une couleur. En utilisant des filtres de couleurs différentes sur les pixels voisins, les couleurs manquantes peuvent alors être interpolées.

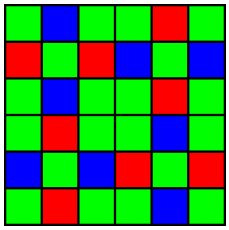
Bien qu'il en existe d'autres, presque tous les appareils utilisent la même CFA : la matrice de Bayer. Voir la figure 0.3 pour des exemples de motifs. Cette matrice échantillonne la moitié des pixels en vert, un quart en rouge et le dernier quart en bleu. L'échantillonnage d'un plus grand nombre de pixels en vert est justifié par le système visuel humain, qui est plus sensible à la couleur verte.



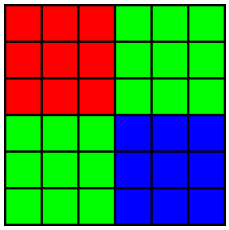
(a) Bayer



(b) RGBW, utilisé dans certains appareils Kodak.



(c) Fujifilm X-Trans, utilisé dans des appareils Fujifilm pour réduire les artefacts de couleur **xtrans**.



(d) Nonacell, utilisé dans le Samsung Galaxy S20 Ultra. Il est similaire au motif de Bayer, mais utilise le *pixel binning* pour améliorer la sensitivité en cas de mauvaise luminosité [3].

FIGURE 0.3 : Different colour filter arrays.

Contrairement aux autres étapes de la création d'une image, une grande variété d'algorithmes est utilisée pour démosaïquer une image.

Aucune méthode de démosaïquage n'est parfaite - après tout, il s'agit de reconstruire des informations manquantes - et produit un certain niveau d'artefacts, bien

que certaines en produisent beaucoup moins que d'autres[2]. Il est donc possible de détecter ces artefacts pour obtenir des informations sur la méthode de démosaïquage appliquée à l'image, ce qui sera l'objet de cette thèse.

Correction couleur

La balance des blancs vise à ajuster les valeurs obtenues par les capteurs afin qu'elles correspondent aux couleurs perçues par l'observateur en ajustant les valeurs de gain de chaque canal. La manière dont la balance des blancs ajuste la sortie dépend des caractéristiques des sources lumineuses, et est effectuée de manière à ce que les objets achromatiques de la scène réelle soient rendus comme tels [4].

Par exemple, la balance des blancs peut être obtenue en multipliant la valeur de chaque canal, de sorte qu'un pixel qui a une valeur maximale dans chaque canal se retrouve avec la même valeur maximale 255 dans tous les canaux.

Ensuite, l'image subit une correction gamma. La charge accumulée par le capteur est proportionnelle au nombre de photons incidents sur le dispositif pendant le temps d'exposition. Cependant, la perception humaine n'est pas linéaire par rapport à l'intensité du signal [5]. Par conséquent, l'image est traitée pour représenter précisément la vision humaine en appliquant une fonction concave de la forme $f_{k,\gamma}(u) = ku^{\frac{1}{\gamma}}$, où γ varie généralement entre 1,8 et 2,2. L'idée derrière cette procédure est non seulement d'améliorer le contraste de l'image, mais aussi de coder plus précisément les informations dans les zones sombres, qui sont trop sombres dans l'image brute.

Néanmoins, les appareils modernes n'appliquent généralement pas cette simple fonction, mais plutôt une courbe de tonalité, afin de mettre en correspondance les intensités des images selon des tables précalculées qui simulent la non-linéarité présente dans la vision humaine.

Compression JPEG

Les étapes de l'algorithme de compression JPEG sont détaillées ci-dessous. La première étape du processus de codage JPEG consiste à effectuer une transformation de l'espace couleur de RGB en YC_BC_R où Y est la composante de luminance et C_B et C_R sont les composantes de chrominance de la différence bleue et de la différence rouge. L'œil étant moins sensible aux changements de couleur qu'aux changements de luminance, les composantes de couleur peuvent être sous-échantillonnées sans trop affecter la perception visuelle. Le taux de sous-échantillonnage généralement appliqué est de 4 :2 :0, ce qui signifie que les résolutions horizontale et verticale sont réduites d'un facteur 2. Après le sous-échantillonnage des couleurs, chaque canal est divisé en blocs de 8×8 et chaque bloc est traité indépendamment. La transformée en cosinus discrète (DCT) est appliquée à chaque bloc et les coefficients sont quantifiés.

[2] De manière surprenante, nous avons constaté que les méthodes plus avancées, qui produisent moins d'artefacts, ne sont pas toujours plus difficiles à analyser que les méthodes plus simples. Ceci est particulièrement évident dans le chapitre 5 (Demosaicing to Detect Demosaicing).

Le facteur de qualité JPEG, compris entre 1 et 100, correspond au taux de compression de l'image. Plus ce taux est faible, plus le fichier résultant est léger, mais plus l'image est détériorée. Une matrice de quantification liée à ce facteur fournit un coefficient pour chaque composante des blocs DCT. C'est lors de cette étape de quantification que se produit la plus grande perte d'information, mais c'est aussi cette étape qui permet le mieux de réduire le poids de l'image. Les coefficients des hautes fréquences, dont l'œil peine à distinguer les variations, sont les plus quantifiés, allant parfois jusqu'à être entièrement mis à zéro, entraînant alors une perte totale des informations de haute fréquence.

Enfin, comme dans l'exemple de la figure 0.4, les blocs quantifiés sont encodés sans perte pour obtenir un fichier JPEG.

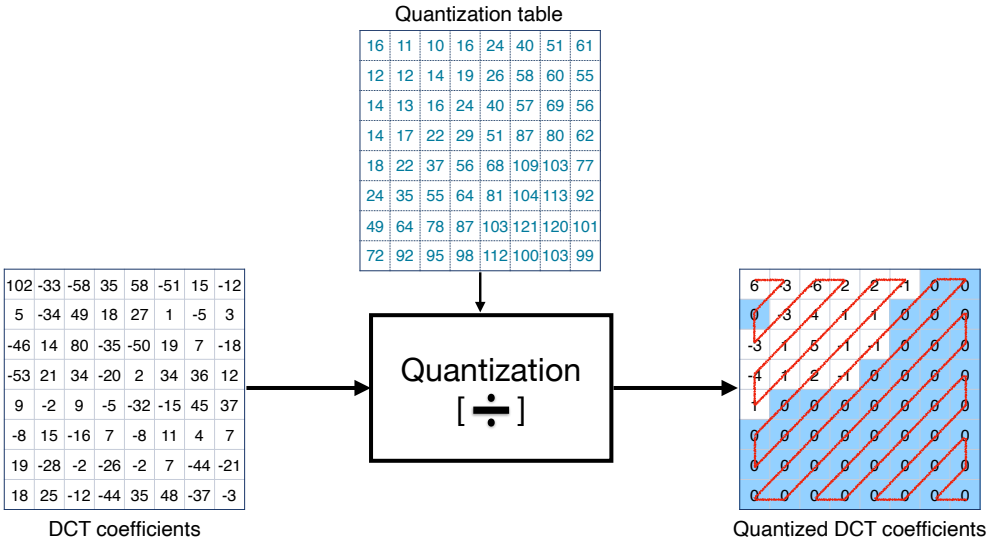


FIGURE 0.4 : Un exemple de l'impact de la quantification sur un bloc DCT. Chaque coefficient DCT est quantifié par une valeur trouvée dans une matrice de quantification. L'arrondi à l'entier le plus proche entraîne la mise à zéro d'un grand nombre de coefficients de haute fréquence. Chaque bloc est mis en zigzag pour être codé comme un vecteur.

Reconfiguration de la chaîne de traitement pour reconstruire l'historique d'une image

Chaque étape laisse des traces spécifiques sur l'image. Ces traces peuvent être détectées et analysées pour révéler comment une image spécifique a été traitée. Cette connaissance est de la plus haute importance pour l'authentification des images. Savoir quelle chaîne de traitement a été utilisée pour créer une image permet de la relier à un appareil photo ou au moins à un modèle d'appareil. Il est alors possible de dire si une image peut provenir d'un appareil donné.

La rétro-ingénierie de la chaîne de formation de l'image, ou d'une partie de celle-ci, est également au cœur de nombreuses méthodes de détection des falsifications. En effet, l'altération d'une image modifie souvent aussi ses traces ; la région contrefaite apparaît alors incohérente avec le modèle de rétro-ingénierie.

0.2 Comment détecter les falsifications d'images ?

Les différentes traces laissées lors de la création de l'image peuvent être analysées séparément. Cette thèse se concentre sur l'une de ces traces, à savoir le démosaïquage.

Analyse du démosaïquage

Comme nous l'avons expliqué précédemment, l'image brute n'est pas une image couleur à 3 canaux, mais chaque pixel est échantillonné dans une couleur, selon un réseau de filtres colorés (CFA). Le CFA de Bayer, illustré dans la Figure 0.3, est de loin le plus courant ; les méthodes d'analyse de démosaïquage sont donc généralement conçues en supposant que l'image a été traitée avec le CFA de Bayer, et nous ferons de même. Bien que d'autres CFA existent,leur utilisation reste limitée. Notez que si les méthodes d'analyse du démosaïquage ont été conçues en tenant compte de la matrice de Bayer, la plupart pourraient facilement être adaptées à d'autres CFA.

Pour détecter les falsifications d'images via l'analyse du démosaïquage, deux pistes peuvent être suivies.

On peut essayer directement de détecter les régions où aucune trace de démosaïquage n'est présente. Cela peut être dû à une manipulation directe sur l'image, comme le floutage qui supprime les traces de démosaïquage. Même en présence de traces de démosaïquage, si ces traces sont différentes, certaines méthodes peuvent détecter une absence de traces si elles diffèrent fortement du reste de l'image. Cela peut être le cas, par exemple, dans les collages d'objets externes. Une telle analyse doit être effectuée avec précaution, car il est assez fréquent que les images naturelles ne présentent aucune trace de démosaïquage, par exemple dans les régions plates où un démosaïquage parfait est possible.

D'autre part, il est également possible de rechercher des décalages dans le modèle CFA. Comme le montre la figure 0.6, en cas de copier-coller, qu'il soit interne ou externe, il y a une chance de $\frac{3}{4}$ que le motif CFA de la région collée ne soit pas aligné avec celui de l'image originale. Ce décalage de périodicité peut être détecté comme une incohérence dans la mosaïque de l'image, et donc la preuve d'une falsification potentielle. De manière presque équivalente, il est également possible de détecter localement la mosaïque utilisée dans l'image. Avec la matrice de Bayer, seuls quatre motifs sont possibles, chacun étant un décalage des autres motifs. Les motifs possibles peuvent être regroupés en deux par leur diagonale, c'est-à-dire les pixels échantillonnés en vert qu'ils partagent. C'est ce que montre la figure 0.5.

De nombreuses méthodes peuvent implicitement faire les deux, bien qu'elles se concentrent généralement sur un seul cas. Une différence essentielle entre les deux classes est en effet les canaux sur lesquels on se penche. Lorsqu'elles recherchent l'absence de traces de démosaïquage, de nombreuses méthodes ne considèrent que le canal vert. Comme la moitié des pixels sont échantillonnés en vert, le démosaïquage du canal vert est plus simple que ses homologues rouge et bleu, et est donc plus facile à analyser. En revanche, lorsqu'on recherche des décalages dans la mosaïque, le canal vert est beaucoup moins informatif ; en effet, comme on peut le voir dans la Figure 0.5, bien qu'il y ait quatre motifs au total, ils sont regroupés en deux paires partageant la même diagonale, c'est-à-dire qu'ils partagent leurs pixels échantillon-

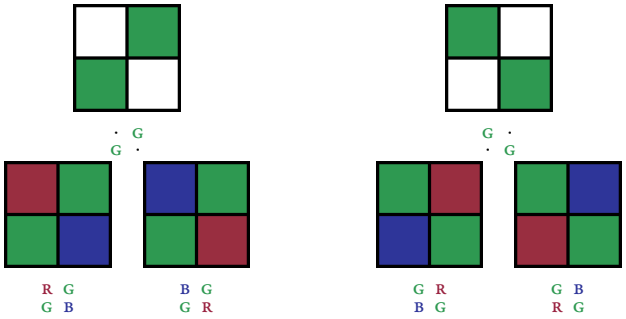
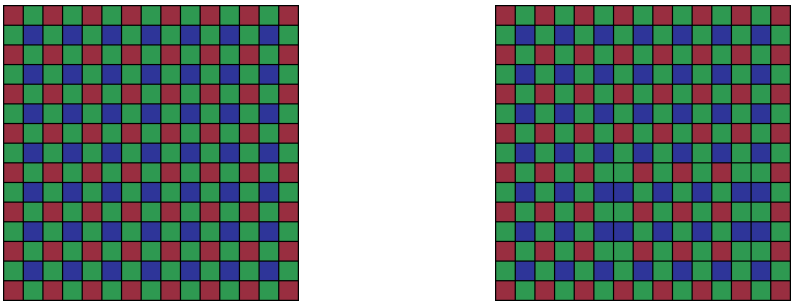


FIGURE 0.5 : Les quatre schémas d'échantillonnage possibles peuvent être regroupés selon la diagonale sur laquelle le canal vert a été échantillonné : $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ et $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ partagent la diagonale $\begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}$, tandis que $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$ et $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$ partagent $\begin{smallmatrix}G&\cdot\\\cdot&G\end{smallmatrix}$.

nés en vert. Si l'on ne regarde que les canaux verts, les déplacements entre deux motifs partageant leur diagonale de verts seront manqués.

Dans la suite de cette section, nous passons en revue les travaux connexes sur l'analyse du démosaïquage dans le cadre de l'analyse des images.



(a) Authentic image (b) Forged image

FIGURE 0.6 : Couleurs dans lesquelles les pixels sont échantillonnés dans une image authentique et une image contrefaite. Dans la zone contrefaite de la deuxième image, il existe une probabilité de $\frac{3}{4}$ que les motifs de la zone authentique et de la zone contrefaite soient mal alignés, ce qui entraîne un décalage de l'CFA, par ailleurs périodique.

Dans un article pionnier sur l'analyse du démosaïquage, POPESCU et FARID [6] propose d'estimer conjointement un modèle linéaire pour l'algorithme de démosaïquage et de détecter quels pixels ont été échantillonnés dans un canal donné avec un algorithme d'*expectation-maximization* (EM). L'algorithme de démosaïquage est estimé sur les pixels détectés comme interpolés (c'est-à-dire non échantillonnés), comme une combinaison linéaire des pixels voisins dans ce canal. Les pixels échantillonnés sont détectés comme des pixels pour lesquels la combinaison linéaire donne un résultat éloigné de la valeur correcte du pixel. Une carte de pseudo-probabilité de chaque pixel échantillonné est alors calculée. En supposant que le modèle linéaire soit correct, les pixels échantillonnés seront correctement détectés et la carte présentera une forte composante 2-périodique, facilement visible sous la forme d'un pic dans la transformée de Fourier de l'image. Cependant, dans une région qui a été altérée, le modèle linéaire estimé ne sera plus correct, soit parce que

l'estimation du démosaïquage apparaît différemment, soit parce qu'il ne reste aucune trace de démosaïquage. Le pic de 2-périodicité va donc localement disparaître, cette disparition peut être détectée comme une preuve potentielle de falsification. Comme beaucoup d'autres méthodes qui recherchent des traces incohérentes sans examiner le motif spécifique, l'algorithme est utilisé uniquement sur le canal vert, où le démosaïquage est généralement considéré comme plus facile à détecter.

Le même concept est utilisé par Liu, Zhao, Ni et al. [7], où la carte de pseudo-probabilité calculée est analysée par un réseau de neurones pour distinguer les vrais faux des artefacts de post-traitement tels que la compression JPEG. La méthode originale propose de trouver des régions dans lesquelles la périodicité disparaît ; toutefois, si la carte de pseudo-probabilité est correcte, les changements de périodicité de la mosaïque devraient également s'y refléter. González Fernández, Sandoval Orozco, Garcia Villalba et al. [8] étend cette méthode en visualisant la carte de pseudo-probabilité avec une transformée en cosinus discrète (DCT) au lieu d'une transformée de Fourier. Les variations de la périodicité de la mosaïque apparaissent alors comme un changement de signe dans le domaine de la DCT, et peuvent donc être détectées. Les principales limites de ces méthodes proviennent de l'estimation linéaire elle-même. Bien que cette hypothèse ait pu être parfaitement raisonnable dans le passé, les algorithmes de démosaïquage les plus couramment utilisés de nos jours sont fortement non linéaires. De plus, ils se comportent différemment selon les régions de l'image, et ce qui peut apparaître comme une absence ou une incohérence du démosaïquage peut au contraire être simplement une région de nature différente, que le même algorithme de démosaïquage a traitée différemment.

Ferrara, Bianchi, De Rosa et al. [9] proposent également de rechercher les régions où les traces de démosaïquage sont localement absentes. En travaillant sur le canal vert, ils partent du principe qu'il faut connaître le motif global de l'image[3]. Ils appliquent un prédicteur fixe, généralement un démosaïquage bilinéaire, et examinent la différence de variance entre les deux treillis des pixels supposés échantillonnés et interpolés. En présence d'artefacts de démosaïquage, la variance est nettement plus élevée sur le treillis des pixels échantillonnés, qui ne peut être estimé avec précision à partir des pixels interpolés. En revanche, en l'absence de telles traces, la variance est égale sur les deux treillis. L'analyse bayésienne à l'aide de modèles de mélange gaussien à plusieurs échelles de blocs met alors en évidence les régions où les traces de démosaïquage sont absentes dans une image autrement démosaïquée.

Kirchner [10] proposent d'estimer le motif d'une image en effectuant un démosaïquage inverse avec un algorithme de démosaïquage simple tel que l'interpolation bilinéaire. Ils recréent artificiellement la mosaïque de l'image et estiment les canaux de pixels masqués avec l'algorithme fixe. Même lorsque l'image n'a pas été démosaïquée à l'origine avec le même algorithme, le second démosaïquage donne généralement de meilleurs résultats sur le motif original, ce qui permet son identification.

Dirik et Memon [11] calcule deux caractéristiques liées au démosaïquage. La première tente de classifier le motif d'une image. Pour ce faire, ils effectuent un démosaïquage bilinéaire inverse, comme dans Kirchner [10], localement et en se li-

[3] Connaître le motif CFA d'une image complète n'est généralement pas trop difficile, surtout lorsqu'on ne s'intéresse qu'au motif diagonal comme ici.

mitant aux blocs non lisses, dans lesquels les artefacts sont plus visibles. La deuxième caractéristique est une analyse du bruit. Ils estiment le bruit d'une image avec un algorithme de débruitage standard, et comparent la variance du bruit estimé sur les différents motifs. En effet, la variance du bruit devrait être plus élevée sur les pixels échantillonnés, à condition qu'ils proviennent de l'image réelle et n'aient pas été ajoutés manuellement après le démosaïquage.

En partant du fait que la plupart des algorithmes de démosaïquage évitent d'interpoler contre un gradient abrupt, Swaminathan, Wu et Liu [12] proposent de réaliser trois modèles linéaires différents des algorithmes de démosaïquage, pour un gradient lisse, horizontal ou vertical.

Le et Retraint [13] étendent leur méthode à la détection de falsification : en travaillant sur le canal vert, ils estiment les trois modèles linéaires dans les deux schémas diagonaux possibles pour sélectionner le schéma donnant l'erreur la plus proche globalement. Pour filtrer les régions susceptibles d'apporter de l'instabilité à la détection, ils ne travaillent ensuite que sur les régions lisses. Des caractéristiques basées sur le rapport des variances entre les treillis échantillonnés et interpolés sont construites. Cette caractéristique est normalement distribuée dans les images authentiques démosaïquées ; un test statistique est alors utilisé pour localiser les régions qui s'écartent significativement de la distribution normale estimée par EM.

Choi, Choi et Lee [14] travaillent sur une caractéristique plus subtile du démosaïquage. Ils remarquent que les pixels interpolés sont plus susceptibles d'être des valeurs intermédiaires par rapport à leurs voisins, alors que les pixels échantillonnés sont plus susceptibles d'être des extrema locaux. En comptant le nombre de valeurs intermédiaires dans chaque modèle possible, on obtient une estimation du modèle correct. Si la méthode originale se limite à l'identification du motif, sans chercher à détecter les falsifications, dans le chapitre 3 (Intermediate Values Counting for CFA Pattern identification), nous analyserons, implémenterons et étendrons cette méthode pour détecter les falsifications.

[15] remarque que la plupart des méthodes de détection de démosaïquage travaillent séparément sur chaque canal, alors que la plupart des algorithmes de démosaïquage utilisent largement les informations des autres canaux. Afin de mimer ce comportement, ils proposent de travailler sur les différences de canaux, autrement dit sur les cartes $R - G$ et $B - G$, où R, G, B représentent les canaux rouge, vert et bleu de l'image. Ils étudient ensuite la différence de variance entre les motifs possibles afin d'identifier le motif correct comme étant celui dont la variance est la plus élevée.

Plus récemment, Park, Moon et Eom [16] tentent d'éviter de baser leur détection sur la réinterpolation. En effet, les résultats de la réinterpolation sont soumis à l'adéquation du noyau, qui est difficile à estimer correctement. En particulier, lorsque des transferts inter-canaux ont lieu pendant le démosaïquage, la variance peut devenir plus faible dans le motif correct, provoquant ainsi des erreurs d'identification du motif, comme nous le remarquerons également dans les chapitres 2 (CFA Identification with Differential Operators) et 3 (Intermediate Values Counting for CFA Pattern identification). Ils remplacent la réinterpolation par une décomposition en valeurs singulières dans chaque canal, de manière à supprimer les informations de fond et à obtenir un résidu plus fiable.

Au final, les méthodes actuelles d'analyse de démosaïquage souffrent de deux défauts :

- La **grande variété de méthodes de démosaïquage**, rend la détection difficile avec des prédicteurs fixes, tandis que la non-linéarité et les nombreux transferts inter-canaux du démosaïquage rendent l'estimation d'un algorithme de démosaïquage complexe, surtout sur une seule image.

- Le démosaïquage intervient au tout début de la chaîne de traitement de l'appareil. Par conséquent, les traces de démosaïquage sont fortement influencées par tout ce qui se passe après le démosaïquage. En particulier, la compression JPEG supprime rapidement les fréquences les plus élevées d'une image, où se trouvent les traces de démosaïquage. Les méthodes d'analyse du démosaïquage sont donc peu ou pas du tout résistantes à la compression JPEG et ne peuvent analyser que des images non compressées. Le sous-échantillonnage d'image supprime également toutes les traces de démosaïquage si le facteur de sous-échantillonnage est d'au moins 2, et altère fortement ces traces même avant cela.

Le deuxième défaut limite l'applicabilité de l'analyse du démosaïquage aux images de haute qualité. S'il est peu probable qu'elle soit utile pour détecter des faux dans des images de faible qualité, comme celles que l'on trouve sur les médias sociaux, qui sont généralement sous-échantillonnées et compressées, cette analyse reste pertinente dans d'autres domaines où l'on peut trouver des images de haute qualité, comme les enquêtes criminelles ou les concours photographiques. Néanmoins, même dans ces domaines, un certain degré de robustesse à la compression JPEG est souhaitable. En effet, même les images de haute qualité sont souvent stockées avec un facteur de qualité JPEG de 95 ou même de 90, ce qui permet de réduire le poids de l'image pour une perte à peine perceptible visuellement.

Autres méthodes de détection de falsification

Bien que cette thèse se concentre sur les traces de démosaïquage, d'autres traces peuvent également être exploitées pour détecter des falsifications d'images.

Les méthodes basées sur le **niveau de bruit** analysent le modèle de bruit des images (voir la section 0.1) afin de trouver des régions présentant une quantité de bruit différente, qui pourrait résulter d'une altération. Mahdian et Saic [17] effectuent une estimation du niveau de bruit basée sur les ondelettes locales en utilisant un estimateur de déviation absolue médiane. Lyu et al. [18] s'appuie sur le phénomène de concentration du kurtosis. Plus récemment, Noisesniffer [19] définit un modèle stochastique de fond permettant la détection d'anomalies de bruit locales et statistiquement significatives. Ces méthodes peuvent potentiellement détecter une variété relativement large de contrefaçons, car chacune d'entre elles peut modifier le niveau de bruit.

La **compression JPEG** laisse les effets de blocage et la quantification du coefficient DCT de chaque bloc. Les outils de forensique JPEG peuvent donc être divisés en deux catégories. BAG [20] et CAGI [21] analysent les artefacts de blocage, tandis que les autres méthodes analysent les coefficients DCT. Plus précisément,

CDA [22] et I-CDA [23] sont basés sur les distributions des coefficients AC, tandis que FDF-A [24] est basé sur la distribution du premier chiffre des coefficients AC. Zero [25] compte le nombre de coefficients DCT nuls dans tous les blocs et en déduit l'origine de la grille. Ces méthodes ne peuvent fonctionner que lorsque la falsification a été faite après une première compression JPEG, mais donnent généralement de très bons résultats lorsque c'est le cas.

Méthodes génériques. Toutes les méthodes présentées ci-dessus traitent séparément les différents types de traces qui peuvent être analysées. La variété des configurations avant et après la falsification rend l'exhaustivité difficile, mais les résultats obtenus par ces méthodes spécifiques sont explicites. Une autre possibilité est de développer des outils polyvalents pour classer et/ou localiser les faux indépendamment de la configuration et du type de faux.

Splicebuster [26] calcule le bruit résiduel d'une image après un filtre passe-haut, et utilise les cooccurences de ces résidus comme caractéristiques locales de la signature d'une image. Un modèle de mélange gaussien-uniforme est ensuite utilisé pour détecter et localiser les régions où la signature est différente du reste de l'image.

Noiseprint [27] s'étend sur Splicebuster en utilisant des réseaux siamois pour extraire un autre résidu de bruit d'une image. Le réseau est entraîné sur des paires de patchs pour extraire le même résidu si les patchs proviennent du même appareil photo, en utilisant la sortie d'un patch comme cible pour l'autre patch. Sur des patchs provenant d'appareils différents, le modèle est entraîné à produire des résidus différents.

L'analyse d'autoconsistance [28] utilise également un réseau siamois dans le but de détecter si deux patchs sont susceptibles de partager les mêmes métadonnées EXIF, et donc d'avoir été traités avec la même chaîne de traitement. Ils utilisent la segmentation N-Cuts [29] pour regrouper et détecter automatiquement les traces pertinentes de falsifications.

On peut également tenter de détecter les faux directement ; par exemple, ManTraNet [30] est un réseau bipartite de bout en bout, entraîné à détecter les manipulations au niveau de l'image avec une partie, tandis que la seconde partie est entraînée sur des ensembles de données synthétiques de faux pour détecter et localiser les faux dans l'image. Avec de telles méthodes, l'exhaustivité est théoriquement possible. Cependant, les résultats ne sont pas explicites et leurs décisions sont plus difficiles à justifier. En effet, il est difficile de comprendre pour quelle raison un modèle détecte une incohérence. Ces méthodes peuvent également être aveugles à certains types de traces, par exemple, tous les outils polyvalents testés se révèlent dans le chapitre charefcha :trace être aveugles aux changements dans la mosaïque CFA ou dans la grille JPEG, bien que quelques-uns puissent, dans une certaine mesure, détecter les changements dans l'algorithme CFA ou dans la qualité de compression JPEG. Cette cécité aux changements de la mosaïque CFA ou de la grille JPEG n'est pas surprenante, car la plupart de ces méthodes utilisent des réseaux de neurones convolutifs (CNN). En effet, comme les CNN sont invariants à la translation, ils ne peuvent pas détecter les changements dans un motif périodique, sauf peut-être à la limite du changement.

En outre, les méthodes basées sur l'apprentissage peuvent être limitées par les données d'apprentissage et ne pas bien généraliser dans des scénarios non contrôlés.

C'est notamment le cas de méthodes telles que ManTraNet, qui tentent de détecter directement les contrefaçons. Il existe de nombreuses façons de créer des faux, et un modèle formé sur des images falsifiées d'une certaine façon peut être difficile à généraliser à d'autres falsifications.

Jeux de données pour la forensique

Les jeux de données d'images authentiques sont nécessaires à de nombreux algorithmes de détection de falsification. Ils peuvent être utilisés pour entraîner des modèles sur des images authentiques, comme Noiseprint [27] et Self-consistency [28]. Le jeu de données Raise [31] contient 8156 images brutes de divers scénarios. La base de données d'images de Dresde [32] fournit 16 961 images authentiques prises avec 27 appareils différents. Parmi elles, 1 491 images prises avec trois appareils différents, les Nikon D200, D70 et D70s, sont fournies non traitées au format RAW. Des images spécifiques peuvent également être utiles pour des expériences. Par exemple, le jeu de données d'images de test sans bruit [33] contient 16 images presque sans bruit. Ces images ont été soigneusement sous-échantillonnées pour éliminer la plupart du bruit, ainsi que les traces de démosaïquage. En l'absence de démosaïquage préalable, nous pouvons utiliser ces images pour simuler nous-mêmes divers algorithmes de démosaïquage dans différents motifs. De même, le jeu de données Pixelshift200 [34] contient 210 images de haute qualité sans traces de démosaïquage, qui ont été obtenues en fusionnant 4 images prises quasi-simultanément lors du décalage du CFA.

Il existe également une littérature considérable proposant des ensembles de données de falsifications d'images, qui sont nécessaires pour l'évaluation des outils de forensique. Un des premiers exemples est le Columbia Dataset [35], qui ne contient que des blocs copiés de 128 × 128 sur des images en niveaux de gris, pour lesquels aucun masque n'est fourni. Deux ans plus tard, le Columbia Color Dataset a ajouté des images couleur de meilleure résolution et des masques de falsification. De nouveaux benchmarks ont été proposés en 2009 avec CASIA V1.0 et V2.0 [36]. Ces jeux de données comprenaient des attaques par copier-coller externe et interne, avec au total 8000 images vierges et 6000 images falsifiées. Le post-traitement a été présenté comme une technique de contre-analyse. Les jeux de données MICC F220 et F2000 [37] ainsi que IMD [38] fournissent d'autres points de référence pour la détection des mouvements de copie. Ces jeux de données ont été construits de manière automatique. Alors que les deux premiers sélectionnent de manière aléatoire la région de l'image à copier-coller, le jeu de données IMD effectue l'extraction de fragments. Les deux approches incluent la possibilité d'ajouter plusieurs artefacts à la région contrefaite. D'autres jeux de données traitant des contrefaçons par copiage avec des contre-attaques post-traitement sont également disponibles [39], [40].

Les défis liés à la détection des falsifications d'images constituent une autre source de données de référence. Le National Institute of Standards and Technology (NIST) organise, depuis 2017, un défi annuel pour lequel différents jeux de données sont publiés [41]. Les falsifications incluses dans ces jeux de données sont de type varié et elles sont générées à la fois automatiquement et manuellement. Il comprend des faux générés automatiquement et manuellement Il peut donc être utile pour évaluer

la détection de falsifications d'images dans des scénarios non contrôlés.

Certains jeux de données visent à réaliser des falsifications imperceptibles à l'œil nu. Un bon exemple est le jeu de données Korus [42], [43] qui contient 220 images vierges et 220 images falsifiées à la main visant à supprimer ou insérer des objets.

Le récent jeu de données DEFACTO [44] est construit à partir du jeu de données MSCOCO [45] et comprend un large éventail de falsifications telles que le copier-coller interne comme externe, l'inpainting et le morphing. Les falsifications sémantiquement significatives sont générées automatiquement mais avec plusieurs biais tels que le copier-coller d'objets dans le même axe ou uniquement avec des objets simples.

Détection automatique de falsifications

De nombreuses méthodes de détection de falsification n'effectuent pas de détection automatique. Elles produisent plutôt une carte thermique des régions qui semblent être falsifiées et laissent l'utilisateur décider si l'image est effectivement falsifiée. Cependant, l'utilisateur qui cherche à savoir si une image est fausse n'a pas nécessairement les connaissances nécessaires pour interpréter les résultats. En outre, l'analyse visuelle de toutes les images n'est pas possible si de nombreuses images doivent être inspectées. Pour que la détection soit réellement automatique, une méthode de détection de falsification devrait idéalement fournir une sortie binaire de la détection.

Pour y parvenir, Self-consistency [28] fait appel à la segmentation en n-cuts pour sélectionner les régions de la carte thermique sur lesquelles la méthode réagit fortement. Le et Retraint [13] utilisent des tests de normalité pour décider de l'authenticité de l'image.

Dans ce contexte, l'analyse *a contrario* peut s'avérer utile. Introduite par Desolneux, Moisan et Morel [46], elle a déjà été appliquée avec succès à d'autres méthodes de détection de falsification [19], [25]. Basée sur la théorie de la Gestalt, cette théorie de détection effectue un seuillage automatique des données en contrôlant une borne supérieure du nombre de fausses alarmes (NFA) auxquelles on peut s'attendre. Étant donné une hypothèse de base H_0 et une fonction d'importance S qui représente l'importance attribuée à une observation, nous pouvons calculer la valeur p d'une observation x, c'est-à-dire la probabilité qu'une observation aléatoire soit au moins aussi importante que x :

$$p(x) = \mathbb{P}_{y\sim\mathcal{H}_0}(S(y) \geq S(x)). \tag{1}$$

Si la valeur p donne une idée de l'importance d'une détection, elle ne tient pas compte du nombre d'observations effectuées ; en présence d'une grande quantité d'observations, on peut s'attendre à des détections parasites même avec un petit seuil de la valeur p. Le cadre *a contrario* propose plutôt de fixer un seuil sur le nombre de fausses alarmes toléré (NFA), qui est obtenu en multipliant la valeur p par le nombre de tests, réels ou potentiels. Par exemple, lors du calcul de la NFA de rectangles sur une image, le nombre de tests est le nombre total de rectangles possibles dans l'image – même si tous les rectangles ne sont pas testés. Associer un NFA à chaque détection et ne garder que les détections dont le score est inférieur au seuil ε revient à fixer une borne supérieure ε sur le nombre attendu de fausses alarmes dans l'image entière :

$$\mathbb{E}_{x\sim\mathcal{H}_0}\left(|\{x|\mathrm{NFA}(x) < \varepsilon\}|\right) \leq \varepsilon. \tag{2}$$

Le NFA d'une détection appartient à $]0, +\infty[$, les scores plus proches de 0 correspondant à des détections plus significatives. Un NFA de 10^{-3}, par exemple, signifie que sous l'hypothèse de fond $\mathcal{H}_0$, le nombre attendu de détections au moins aussi significatives est au plus 10^{-3}. On s'attend donc à une fausse détection au maximum toutes les 1000 images.

0.3 Résumé de la thèse

Dans cette thèse, nous nous concentrons sur l'analyse des artefacts de démosaïquage pour la détection des falsifications. Cependant, avant d'entrer dans le vif du sujet, nous étudions le problème de l'évaluation des outils de forensique dans le chapitre 1 (Non-Semantic Evaluation of Image Forensics Tools).

How to evaluate forgery detection methods ?

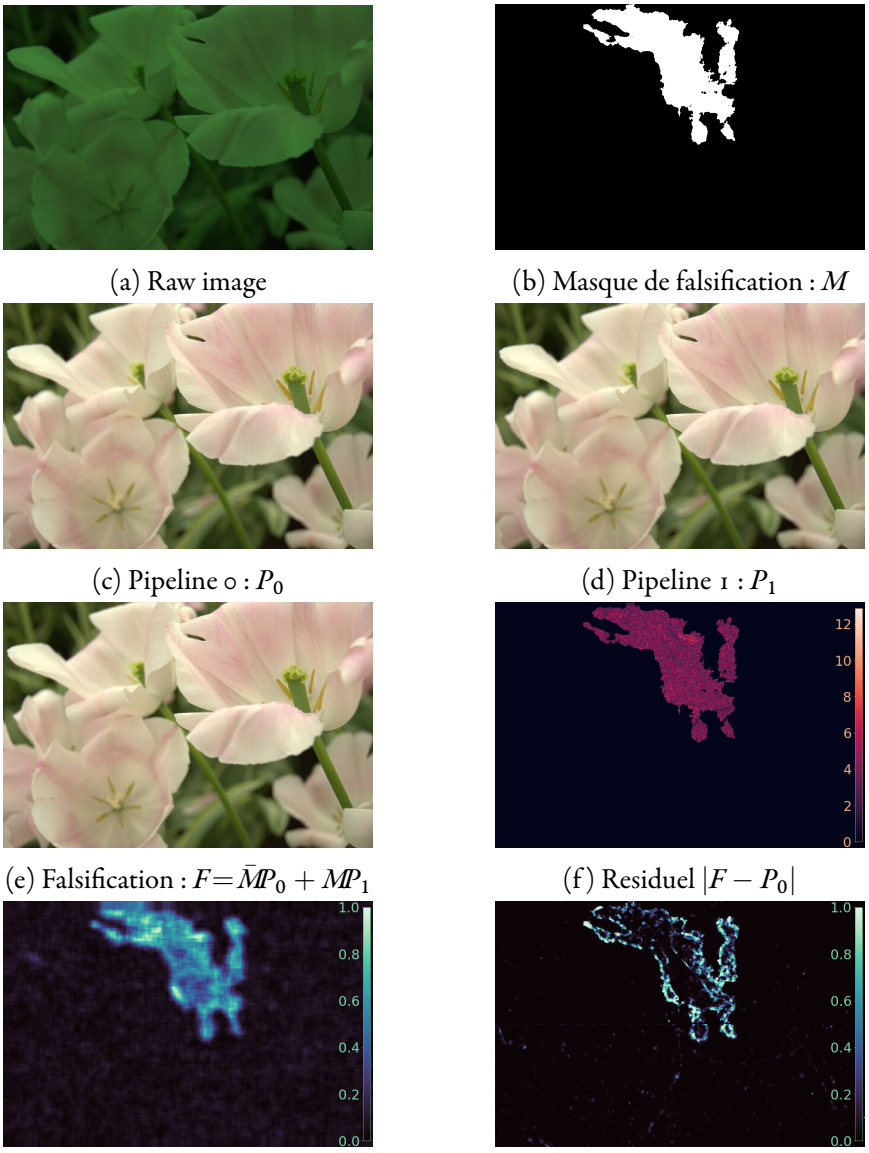
Les algorithmes d'analyse d'images sont principalement évalués en fonction de leurs performances lors de tests de référence. Cette pratique présente plusieurs limites : dans de nombreux cas, la même base de données est divisée en données d'entraînement et d'évaluation. Par conséquent, les algorithmes sont formés et évalués sur des images qui ont subi un de traitement, des algorithmes de falsification et des outils anti-falsification similaires. Il n'y a donc aucune garantie que ces méthodes d'apprentissage fonctionneront dans la nature, où ces paramètres varient beaucoup plus. Indépendamment de la variété de l'ensemble d'apprentissage, la question se pose de savoir si les faux sont détectés par des détecteurs entraînés pour des raisons sémantiques ou en raison d'incohérences locales dans l'image.

En effet, si l'analyse sémantique d'une image peut fournir des indices, la preuve rigoureuse d'un faux ne doit pas être basée uniquement sur des arguments sémantiques. La situation est similaire au dilemme soulevé par les observations de Galilée, qui contredisaient les connaissances acceptées à son époque. connaissances de son époque. Selon Bertolt Brecht [47] :

> Galilée : Et que dirait Son Excellence de regarder ses étoiles impossibles et inutiles à travers ce télescope ?
> Le Mathématicien : L'on pourrait dire, pourtant, que si votre télescope montre quelque chose qui ne peut exister, peut-être n'est-il pas vraiment fiable, n'est-ce pas ?

Le télescope aurait pu être peu fiable, en effet, et une enquête scientifique sur l'instrument aurait pu être justifiée. Cependant, il n'est pas prudent de conclure, comme le fait le mathématicien, que le télescope n'était pas fiable *juste* sur la base du contenu des observations. De même, la preuve d'un faux doit être fondée sur des traces d'images, et non sur des arguments sémantiques, car la sémantique d'une image est généralement le but et non le moyen d'un faux.

Avec ces considérations en tête, nous proposons une méthodologie et une base de données pour évaluer les outils d'analyse d'images sur des images où les régions authentiques et falsifiées ne diffèrent que par les traces laissées par le traitement de



(a) Raw image (b) Masque de falsification : M

(c) Pipeline 0 : P_0 (d) Pipeline 1 : P_1

(e) Falsification : $F = \bar{M}P_0 + MP_1$ (f) Residuel $|F - P_0|$

(g) Noiseprint [27] result (h) ManTraNet [30] result

FIGURE 0.7 : Dans le chapitre 1 (Non-Semantic Evaluation of Image Forensics Tools), différentes chaînes de formation d'image sont appliqués à la même image RAW pour obtenir deux images, qui sont combinées pour obtenir une image falsifiée. Les régions authentique et falsifiée présentent des traces différentes, mais sont par ailleurs parfaitement cohérentes. La seule différence entre les régions authentiques et contrefaites sont les traces de l'appareil. La RMSE dans la région contrefaite est de 4,46. La dernière ligne montre le résultat de deux outils d'analyse sur cette image.

l'images. À l'aide de cette méthodologie, nous créons la base de données *Trace* en ajoutant diverses traces de falsification aux images brutes du jeu d'images Raise [31], comme le montre la figure 0.7. Cette procédure permet d'éviter les difficultés liées à la production de faux sémantiques convaincants et impartiaux, qui nécessitent souvent un travail manuel. Nous créons plusieurs ensembles de données, chacun d'entre

eux correspondant à une incohérence spécifique, comme un niveau de bruit ou un alignement de compression différent. Cela nous donne un aperçu de la sensibilité des outils de forensique à des traces spécifiques, et met ainsi en évidence la complémentarité des différentes méthodes.

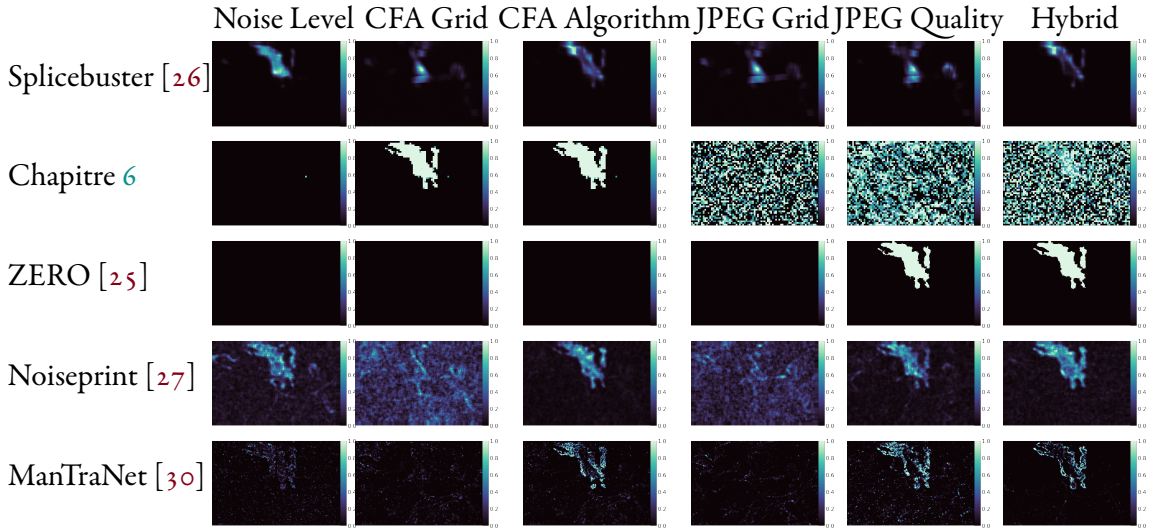


FIGURE 0.8 : Visualisation des résultats de plusieurs méthodes pour une image sur tous les jeux de données du Chapitre 1 (Non-Semantic Evaluation of Image Forensics Tools). Certaines méthodes, comme Noiseprint ou celle proposée dans le Chapitre 6 (Positional Learning for Demosaicing Analysis), détectent correctement les faux dans les images pertinentes, mais ont tendance à faire des fausses détections de type bruit dans les images pour lesquelles elles ne peuvent pas voir le faux. La sélection automatique des détections pertinentes d'un algorithme faciliterait son utilisation sans nécessiter d'interprétation, c'est pourquoi le chapitre 7 (Internal Learning to Improve Adaptability) le fera en étendant la méthode du chapitre 6 (Positional Learning for Demosaicing Analysis). L'image et le masque de falsification sont visibles sur la Fig. 0.7.

À l'aide de la base de données nouvellement créée, nous procédons ensuite à une évaluation des outils de forensique existants, voir Fig. 0.8 pour un exemple.

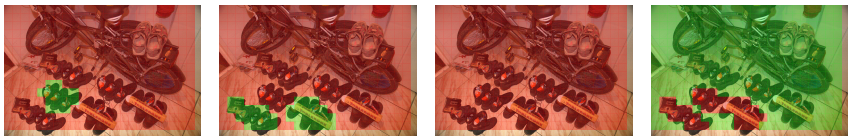
Le reste de la thèse se concentre uniquement sur l'analyse des traces de démosaïquage.

Analyse directe des traces de démosaïquage

Après avoir analysé le problème de l'évaluation des méthodes forensiques, nous commençons notre recherche d'une méthode d'analyse de démosaïquage. Dans le chapitre 2 (CFA Identification with Differential Operators), nous essayons de faire ressortir les pixels échantillonnés des pixels interpolés à l'aide de simples indices numériques provenant d'un opérateur différentiel. Nous comparons la réponse des quatre motifs paire par paire avec un test statistique, afin de détecter les motifs significativement impossibles. Les images sont alors déclarées fausses lorsqu'aucun motif unique n'est possible partout, comme le montre la figure 0.9.

(a) Les régions forgées trouvées sont en rouge, et les régions où la grille CFA exacte a été identifiée sont en vert.

(b) Original image



(c) Régions où $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$, $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$, $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$ et $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ sont significativement impossibles (rouge) ou possible (vert)

FIGURE 0.9 : La méthode du chapitre 2 (CFA Identification with Differential Operators) détecte les motifs CFA significativement impossibles. Images de [38].

La méthode proposée est très robuste aux fausses détections. Cependant, la linéarité et l'indépendance vis-à-vis des canaux rendent les détections rares lorsque l'algorithme de démosaïquage est trop avancé, ou lorsque plusieurs motifs sont présents. Accepter les résultats du canal le plus fort, au lieu de refuser de faire une détection lorsque deux canaux sont incohérents, permet d'obtenir des détections légèrement meilleures, mais le contrôle total du nombre de faux positifs ne peut plus être atteint, comme le montre la figure 0.10.

Le chapitre 2 (CFA Identification with Differential Operators) met ainsi en évidence la difficulté d'une analyse cohérente des traces de démosaïquage.

Dans le chapitre 3 (Intermediate Values Counting for CFA Pattern identification), nous conservons l'idée d'utiliser un indice numérique simple pour mettre en évidence les pixels interpolés par rapport aux pixels échantillonnés. Cependant, nous essayons de suivre une approche plus subtile. Au lieu de mettre directement en évidence les pixels interpolés, nous utilisons plutôt une de leurs propriétés : comme l'a remarqué CHOI, CHOI et LEE [14] et mis en évidence dans la Figure 0.11, les pixels interpolés sont plus enclins à être des valeurs intermédiaires pour détecter dans quel schéma une image a été échantillonnée. Nous analysons, implémentons et étendons leur méthode pour détecter le motif CFA. Nous utilisons ensuite cette information pour trouver les régions qui sont incohérentes avec l'image globale. Nous attribuons un score de confiance à chaque détection, qui peut ensuite être seuillé pour fournir une carte binaire des falsifications détectées. Bien que cette méthode ne donne pas de résultats cohérents sur quelques algorithmes de démosaïquage, elle est globalement bonne pour détecter la mosaïque, du moins sur les images non compressées, comme le montre la figure 0.12.

Une démonstration en ligne de ce chapitre est disponible à l'adresse suivante : https ://www.ipol.im/pub/art/2021/355/. Dans le cadre du projet Envisu4, la mé-

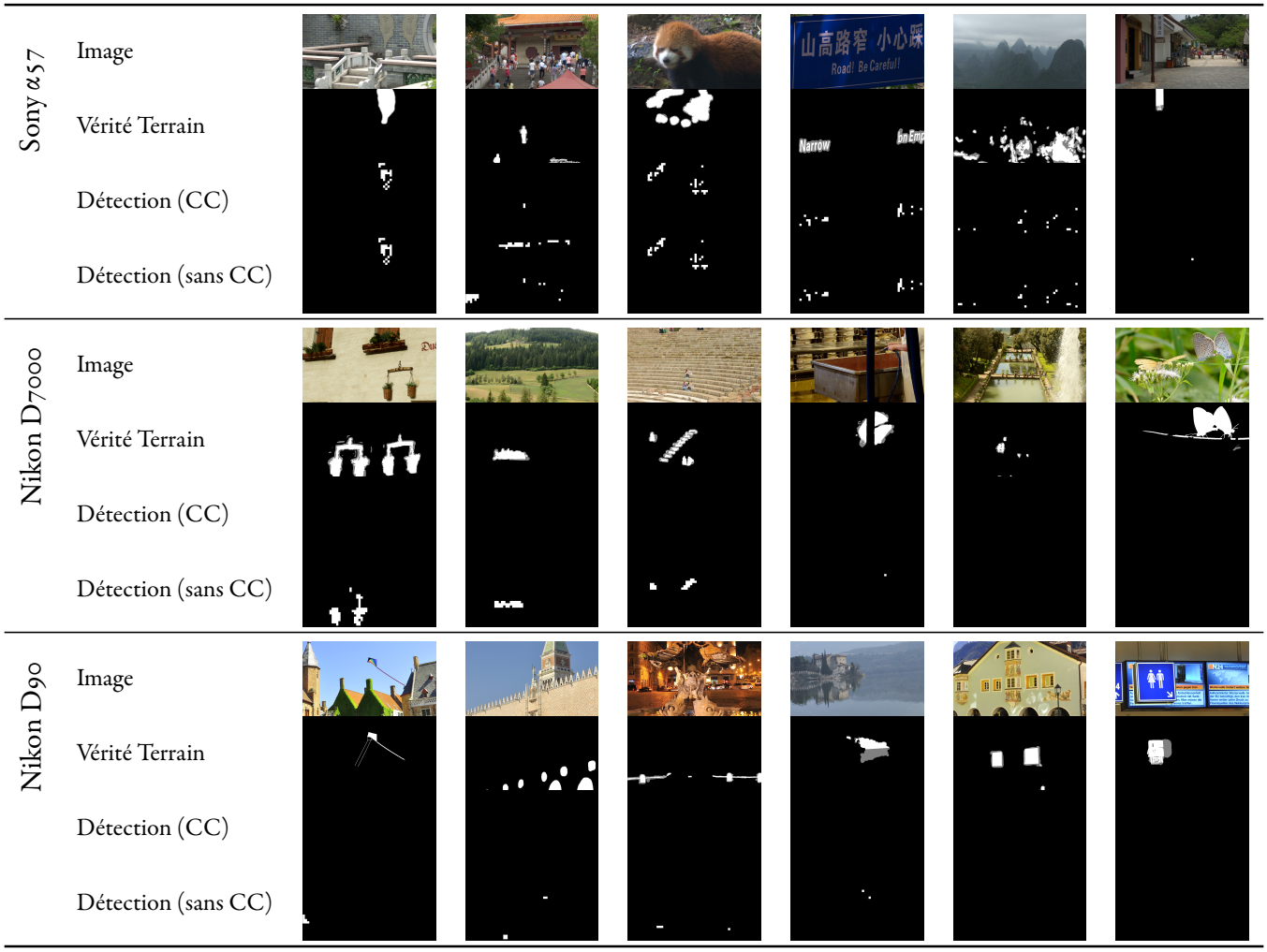


FIGURE 0.10 : Résultats visuels sur les images du jeu de données Korus selon si la cohérence des canaux (CC) est appliquée ou non, avec la méthode du chapitre 2 (CFA Identification with Differential Operators). Aucune détection n'a été faite sur les images de l'appareil photo Canon 60D, qui ne sont donc pas présentées ici. Dans l'ensemble, le fait de ne pas imposer la cohérence des canaux permet de faire beaucoup plus de détections, au prix de quelques faux positifs comme l'image la plus à gauche de l'appareil Nikon D90, ou les deuxième et sixième images de l'appareil Sony α57.

thode présentée ici a également été intégrée au plugin de navigateur InVID & WeVerify, un outil en ligne et un plugin destinés aux journalistes et aux vérificateurs de faits pour vérifier l'authenticité des images et rechercher des traces de falsification.

Bien que la méthode soit utilisable, ces problèmes la rendent encore insatisfaisante. Il est possible de détecter des incohérences locales même au sein de détections erronées, cependant les détections de motifs erronés ne peuvent être évitées contre tous les algorithmes de démosaïquage sans exploiter les transferts inter-canaux. En outre, nous souhaitons améliorer la robustesse à la compression JPEG. Cela nous amène à abandonner l'idée de la détection directe des pixels interpolés et échantillonnés. Au lieu de cela, nous essayons de faire de la rétro-ingénierie de l'algorithme

18	56	94	85	76	96	116	104
49	56	63	52	41	64	88	87
80	56	32	19	6	33	60	70
59	62	66	49	32	59	87	88
38	69	100	79	58	86	114	106
40	51	63	74	85	75	66	63
42	34	26	69	112	65	18	21
38	47	57	75	94	72	50	35

(a) Rouges

139	240	154	16	94	56	72	20
92	131	168	76	72	94	24	43
85	24	100	48	102	224	130	72
60	107	160	68	64	122	200	153
92	184	125	0	50	0	133	108
52	155	156	76	136	117	224	127
146	228	111	12	110	108	107	44
56	114	48	90	184	141	52	90

(b) Verts

FIGURE 0.11 : Canaux rouge et vert d'une image jouet démosaïquée par interpolation bilinéaire dans le modèle $\begin{smallmatrix} R & G \\ G & B \end{smallmatrix}$. Les valeurs rouges correspondent aux positions où la valeur a été interpolée. Les cellules surlignées correspondent aux pixels qui prennent une valeur intermédiaire, c'est-à-dire qui ne sont pas un extremum local parmi leurs voisins directs. Si les pixels échantillonnés peuvent avoir des valeurs intermédiaires, on en trouve beaucoup plus parmi les pixels interpolés dans les canaux rouge et vert. Le canal bleu, non représenté ici, se comporte de manière similaire au canal rouge. Cette observation est au cœur de CHOI, CHOI et LEE [14], dont la méthode est analysée, implémentée et étendue dans le chapitre 3 (Intermediate Values Counting for CFA Pattern identification).

de démosaïquage lui-même, afin d'estimer la mosaïque potentielle dans laquelle un algorithme de démosaïquage a été effectivement utilisé.

Ingéniérie inverse de l'algorithme de démosaïquage

Les chapitres 2 (CFA Identification with Differential Operators) et 3 (Intermediate Values Counting for CFA Pattern identification) ont exploré des approches directes pour distinguer les pixels échantillonnés, et *in fine* pour révéler le motif CFA correct. Cependant, ces approches n'étaient pas satisfaisantes. Si le chapitre 3 (Intermediate Values Counting for CFA Pattern identification) a fourni de bons résultats, la simplicité de ces approches directes ne pouvait pas prendre en compte les propriétés de chaque algorithme de démosaïquage spécifique, ce qui conduisait à des résultats erronés sur les images démosaïquées par ces algorithmes. Dans le chapitre 4 (Linear Estimation of the Demosaicing Algorithm), nous proposons plutôt de suivre une approche de rétro-ingénierie afin de refléter plus précisément les spécificités de chaque image et algorithme de démosaïquage. Nous créons une estimation linéaire du démosaïquage associé à chacun des quatre modèles CFA possibles. Les résidus de ces modèles donnent une estimation locale du motif CFA dans l'image. Une approche *a contrario* est ensuite appliquée pour trouver les régions dont le motif détecté s'écarte significativement du reste de l'image. Nous montrons que si une estimation linéaire peut être suffisante pour trouver le motif CFA de l'image, les nombreuses non-linéarités du démosaïquage, ainsi que la texture naturelle des images, rendent ce modèle global et linéaire localement peu fiable.

Néanmoins, comme on peut le voir sur la figure 0.13, la méthode *a contrario* déployée ici semble satisfaisante, bien qu'elle ne puisse naturellement pas filtrer les

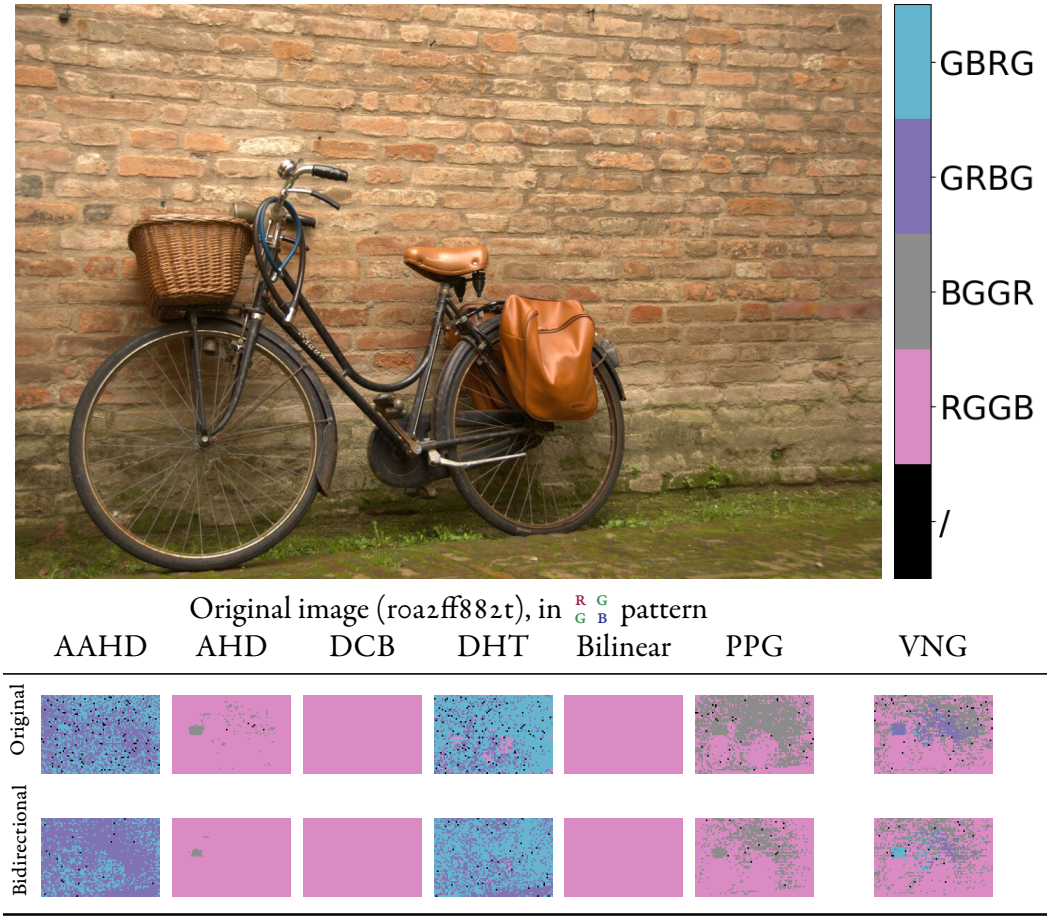


FIGURE 0.12 : Résultats de la méthode du chapitre 3 (Intermediate Values Counting for CFA Pattern identification) avec le filtre original et le filtre proposé, sur une image authentique avec les 7 différents algorithmes de démosaïquage. Les deux méthodes fonctionnent parfaitement sur les images démosaïquées par DCB et Bilinéaire. Avec les méthodes AHD, PPG et VNG, les filtres isotrope original et bidirectionnel ont du mal à discerner les deux motifs partageant la même diagonale, mais la détection bidirectionnelle fait moins d'erreurs. Les régions texturées périodiquement, comme le panier, peuvent créer un décalage localisé dans la mosaïque détectée, ce qui pourrait être confondu avec une contrefaçon. Avec l'algorithme AAHD et DHT, la méthode détecte systématiquement la mauvaise diagonale.

incohérences qui sont significatives dans les votes de blocs eux-mêmes. Les masques de localisation ne sont pas parfaits et peuvent nécessiter une analyse visuelle des votes par blocs pour localiser précisément les algorithmes. Bien que cela puisse avoir un impact négatif sur le score de la méthode sous-jacente, cela n'a que peu d'importance pratique tant que l'approche proposée permet la détection automatique des images falsifiées tout en limitant le nombre de tests à effectuer, et donc le coût de calcul. Cette approche étant suffisamment bonne, nous ne prolongerons pas davantage les travaux sur l'analyse *a contrario*. L'approche que nous avons suivie ici sera réutilisée avec notre méthode finale dans le chapitre 7 (Internal Learning to Improve Adaptability). Cela nous permet de nous concentrer sur l'identification de la mosaïque CFA elle-même dans les chapitres 5 (Demosaicing to Detect Demosaicing) et 6 (Po-

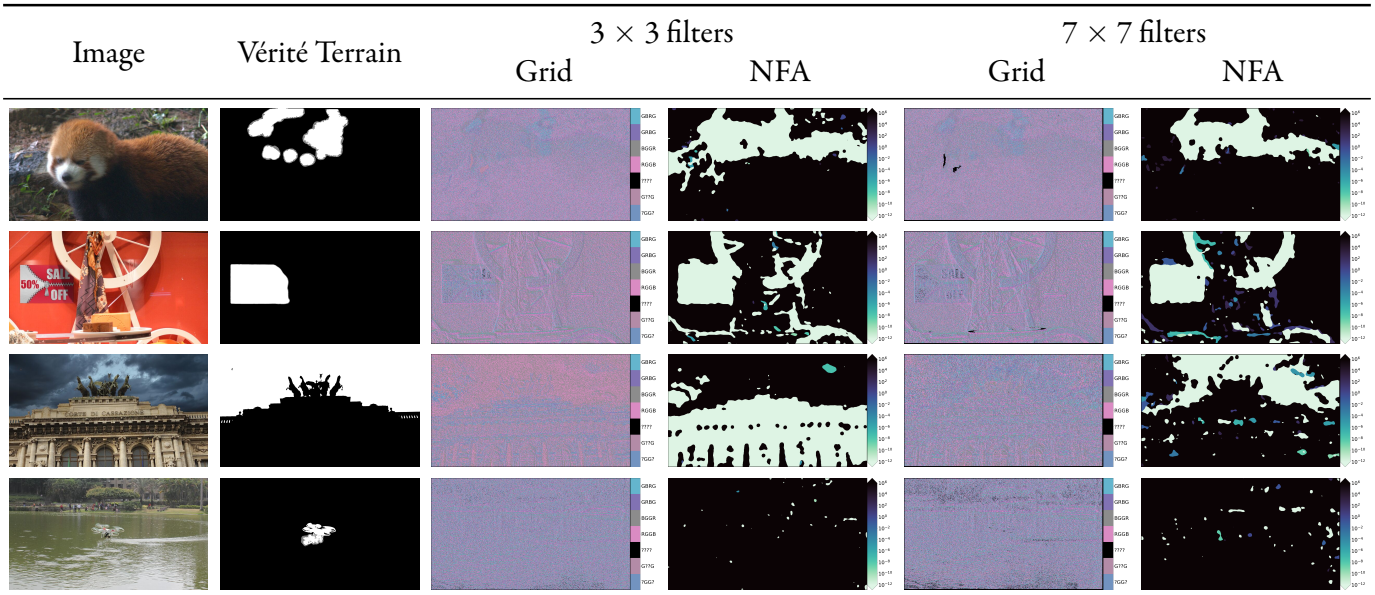
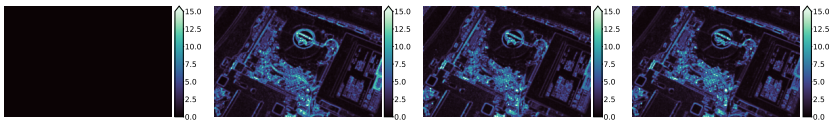


FIGURE 0.13 : Résultats de la méthode du chapitre 4 (Linear Estimation of the Demosaicing Algorithm) sur des images Korus, de haut en bas, provenant des appareils photo Sony α57, Nikon D7000, Nikon D90, Canon 60D. Sur la première image, la contrefaçon est détectée avec précision. Même si le masque détecté est trop grand, une analyse visuelle de la grille détectée permet de localiser précisément le faux. Sur la deuxième image, le faux est détecté, mais une deuxième région de l'image est détectée par erreur. Sur la troisième image, le faux est inversé en utilisant le filtre le plus petit. Bien que cela affecte négativement le score, ce n'est pas vraiment un problème pour la détection, car la méthode montre toujours que les deux régions ne sont pas cohérentes entre elles. Enfin, sur la quatrième image, aucune trace de démosaïquage n'est présente. Certaines régions sont incorrectement marquées comme incohérentes. Ces incohérences sont déjà présentes dans les grilles détectées : ces défauts proviennent de l'estimation linéaire elle-même, et non du seuillage NFA.

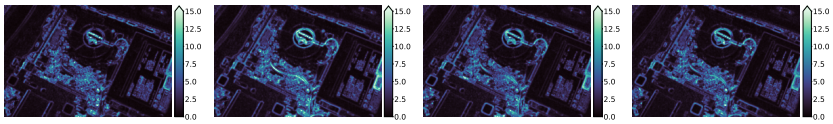
sitional Learning for Demosaicing Analysis), et moins sur l'analyse ultérieure pour la détection des falsifications.

Les méthodes des chapitres 2 (CFA Identification with Differential Operators), 3 (Intermediate Values Counting for CFA Pattern identification) et 4 (Linear Estimation of the Demosaicing Algorithm) présentent deux problèmes : leur incapacité à prendre correctement en compte les transferts inter-canaux du démosaïquage dans les chapitres 2 (CFA Identification with Differential Operators) et 3 (Intermediate Values Counting for CFA Pattern identification) et la nécessité d'aller au-delà des hypothèses linéaires trop simplistes dans les chapitres 2 (CFA Identification with Differential Operators) et 4 (Linear Estimation of the Demosaicing Algorithm). Dans le chapitre 5 (Demosaicing to Detect Demosaicing), nous essayons d'éviter naturellement ces deux problèmes en utilisant une collection d'algorithmes de démosaïquage existants, et acceptons plutôt les hypothèses plus naturelles qui en découlent. En effet, nous pouvons effectuer le démosaïquage en utilisant chacune des quatre mosaïques possibles. La mosaïque correcte a plus de chances de produire une image plus proche de l'image originale, comme le montre la figure 0.14. Nous explo-

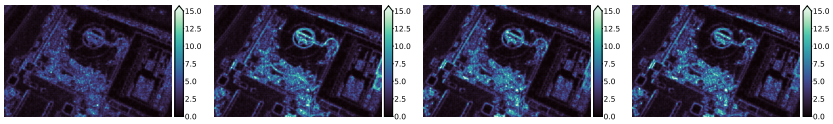
(a) Entrée, démosaïquée avec HA en $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$.



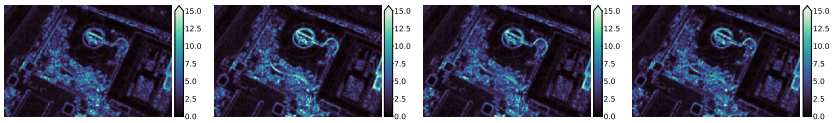
(b) Résidus lorsque l'image d'entrée est à nouveau démosaïquée avec le même algorithme (HA) dans les quatre positions, de gauche à droite : $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ (motif correct), $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$, $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$, $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$. Le résidu est nul lorsque le modèle correct est utilisé, ce qui facilite l'identification du modèle.



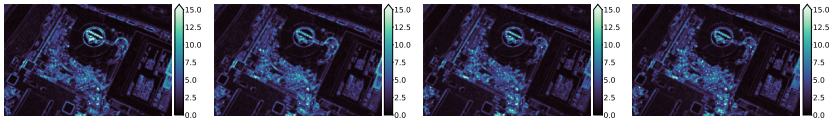
(c) Ici, l'image d'entrée est analysée avec un algorithme différent (LMMSE). Le résidu sur le motif correct (à gauche) n'est plus nul, mais reste plus faible que sur les motifs incorrects.



(d) L'image d'entrée est compressée en JPEG ($Q = 90$ avant le second démosaïquage, avec le même algorithme (HA). Encore une fois, bien que le résidu du motif correct ne soit pas nul, il est toujours plus faible que dans les autres positions.



(e) L'image d'entrée est compressée en JPEG ($Q = 90$ avant le second démosaïquage, avec un algorithme différent (LMMSE). Le résidu est toujours plus faible avec le motif correct.



(f) Ici, l'image d'entrée (non compressée) est analysée par un autre algorithme (ARI).

FIGURE 0.14 : **Double Demosaicing :** Dans le chapitre 5 (Demosaicing to Detect Demosaicing), une image a été démosaïquée avec HA et analysée avec plusieurs algorithmes, à la fois non compressée et après compression JPEG. Lorsque l'image n'est pas compressée entre les deux opérations de démosaïquage, et si l'algorithme utilisé est le même, le résidu est nul dans le motif de mosaïque correct, ce qui permet d'identifier facilement ce motif. Lorsque l'image est compressée, le résidu n'est plus nul dans le motif correct, mais il reste plus faible que dans les autres motifs pour autant que la compression soit modérée. Si un algorithme différent est utilisé pour l'analyse, les résultats varient : L'analyse de l'image avec LMMSE donne un résidu plus faible dans le motif correct, mais ce n'est plus le cas lorsqu'on utilise ARI, qui n'est pas adapté à l'analyse des images démosaïquées HA. Les résiduels sont floutés ($\sigma = 1.7$) pour une meilleure visualisation.

rons les possibilités de ce que l'on appelle le **double démosaïquage** pour identifier le motif CFA. La sélection locale de l'algorithme de démosaïquage le plus approprié permet de détecter le motif CFA même lorsque l'algorithme de démosaïquage original est inconnu. Cette méthode évite naturellement les deux principaux problèmes des chapitres précédents, à savoir leur incapacité à prendre en compte les transferts inter-canaux du démosaïquage et la nécessité d'aller au-delà des hypothèses linéaires trop simplistes. Nous simulons les deux étapes du démosaïquage pour analyser dans quelles conditions le double démosaïquage peut être utilisé pour détecter le motif. De plus, l'introduction de la compression JPEG entre les deux démosaïquages nous permet d'étudier les limites de la détection du motif CFA sur des images compressées.

Lorsque l'algorithme initial n'est pas connu, le motif peut être sélectionné par une stratégie simple. Une liste d'algorithmes peut être utilisée, et l'algorithme qui donne le résidu le plus faible est choisi localement. En principe, cette stratégie donne d'excellents résultats lorsque l'algorithme de démosaïquage initial figure dans la liste des algorithmes testés. Cependant, même lorsque ce n'est pas le cas, cette méthode permet généralement d'obtenir de meilleurs résultats qu'une comparaison utilisant un seul algorithme.

Souvent, les images sont trouvées dans un état comprimé. Toujours dans le chapitre 5 (Demosaicing to Detect Demosaicing), nous étudions la robustesse du double démosaïquage lorsque la compression JPEG est appliquée après le démosaïquage initial. Bien que la détection soit rendue plus difficile par la compression, le motif correct peut toujours être trouvé au niveau de l'image même si l'algorithme de démosaïquage original n'est pas connu.

Le principal inconvénient de cette méthode provient de la difficulté à effectuer des détections fiables à une échelle locale. S'il est possible, avec un degré de confiance très élevé, de détecter si une image entière a été démosaïquée et dans quel schéma, cette décision devient beaucoup plus difficile à prendre localement. Dans les petits blocs de 32×32, le contraste du résidu à travers les motifs est toujours biaisé vers la détection d'un résidu plus faible dans la bonne position ; cependant ce contraste n'est pas significativement plus élevé que le contraste sur les images sans mosaïque, dans lesquelles aucune détection ne devrait être faite. C'est particulièrement le cas sur les images fortement compressées.

Globalement, cette méthode peut être utilisée pour analyser l'image à une échelle globale, aidée par sa surprenante bonne robustesse à la compression JPEG. Il n'est cependant pas encore possible de l'utiliser localement. Une analyse locale serait nécessaire pour détecter les incohérences de la mosaïque et donc les falsifications potentielles.

Ce quatrième chapitre de la quête de preuves fiables de traces de démosaïquage n'apporte pas de réponse universelle. Même l'hypothèse la plus naturelle de proximité entre différents algorithmes de démosaïquage n'est pas toujours valable localement, surtout en cas de forte compression. Cela est compréhensible ; même si différents algorithmes peuvent se comporter de manière proche, ce n'est plus nécessairement le cas si l'un des deux algorithmes est suivi d'un post-traitement lourd.

En somme, aucun des modèles testés jusqu'à présent n'est parfaitement satisfaisant et fiable. La détection directe, bien qu'elle permette d'identifier la grille dans

FIGURE 0.15 : Dans le chapitre 6 (Positional Learning for Demosaicing Analysis), nous entraînons un CNN à détecter les positions horizontales et verticales modulo-2 de chaque pixel, telles que vues dans ces cibles. Les CNN sont invariants en translation, ils ne connaissent pas directement la position des pixels ; ils doivent donc se fier à des indices externes tels que les traces de démosaïquage. Cet entraînement positionnel va donc entraîner implicitement le réseau à reproduire la mosaïque d'une image et ses incohérences.

de nombreux cas, est imparfaite face à plusieurs algorithmes de démosaïquage, et se heurte à la difficulté de concevoir des caractéristiques qui reflètent celles des algorithmes de démosaïquage. La rétro-ingénierie de l'algorithme de démosaïquage d'une image spécifique n'est pas non plus sans faille ; même l'hypothèse somme toute très naturelle selon laquelle l'algorithme de démosaïquage se comportera localement de manière proche d'au moins un algorithme connu n'est pas toujours vraie, et ne fait pas de miracles sur les images fortement compressées.

Apprentissage positionnelle pour répliquer la mosaïque et ses incohérences

Face à la difficulté de notre problème, nous décidons de revenir à l'hypothèse la plus simple et la plus naturelle concernant les traces de démosaïquage : elles présentent une forte composante 2-périodique. Sans autre hypothèse, nous proposons de détecter la phase de cette composante. Pour ce faire, nous introduisons l'apprentissage positionnel. En tirant parti de l'héritage de la traduction et du pouvoir représentatif élevé des réseaux de neurones convolutifs (CNN), nous en entraînons un à détecter la position modulo $(2, 2)$ de chaque pixel, comme le montre la figure 0.15. Implicitement, le CNN s'appuiera sur les traces de démosaïquage pour fournir sa sortie ; cette sortie imitera donc la phase de la composante bi-périodique.

Lorsqu'une falsification perturbe la mosaïque de l'image, la sortie du réseau reflète cette perturbation, permettant la détection de la falsification sous forme d'erreurs dans la sortie. Cette méthode est entièrement auto-supervisée et ne nécessite que des images authentiques pour l'apprentissage. En outre, étant donné plusieurs images similaires dont l'authenticité n'est pas claire, il est possible d'affiner le modèle sur les images pour augmenter la robustesse à la compression JPEG. En pratique, cependant, il est rare de disposer d'une grande quantité d'images similaires à analyser.

C'est pourquoi, dans le chapitre 7 (Internal Learning to Improve Adaptability), nous montrons qu'un tel réseau peut être ajusté avec précision sur une seule image, potentiellement falsifiée, pour s'y adapter. Ce faisant, nous augmentons considérablement la robustesse à la compression JPEG et aux autres post-traitements. Nous

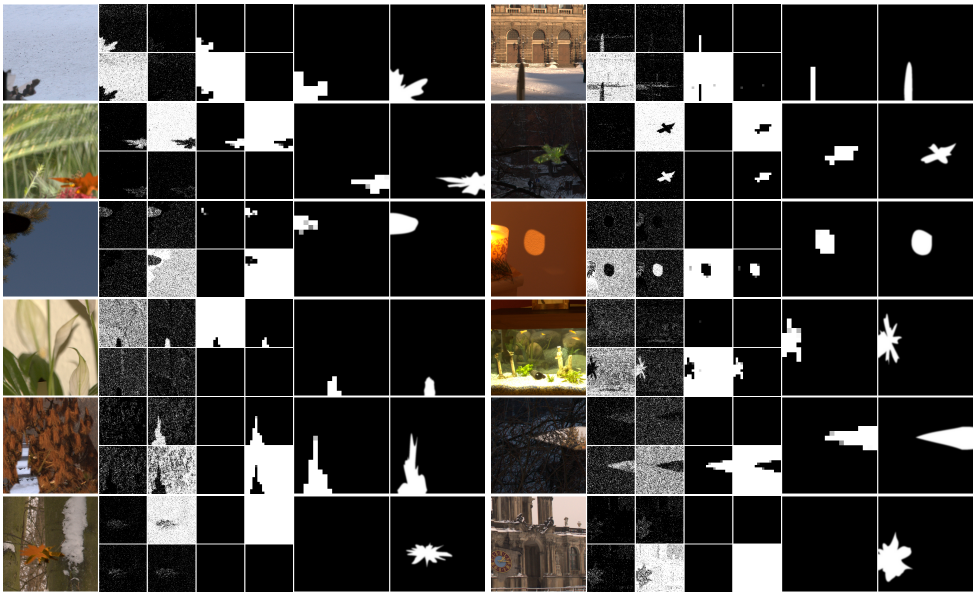
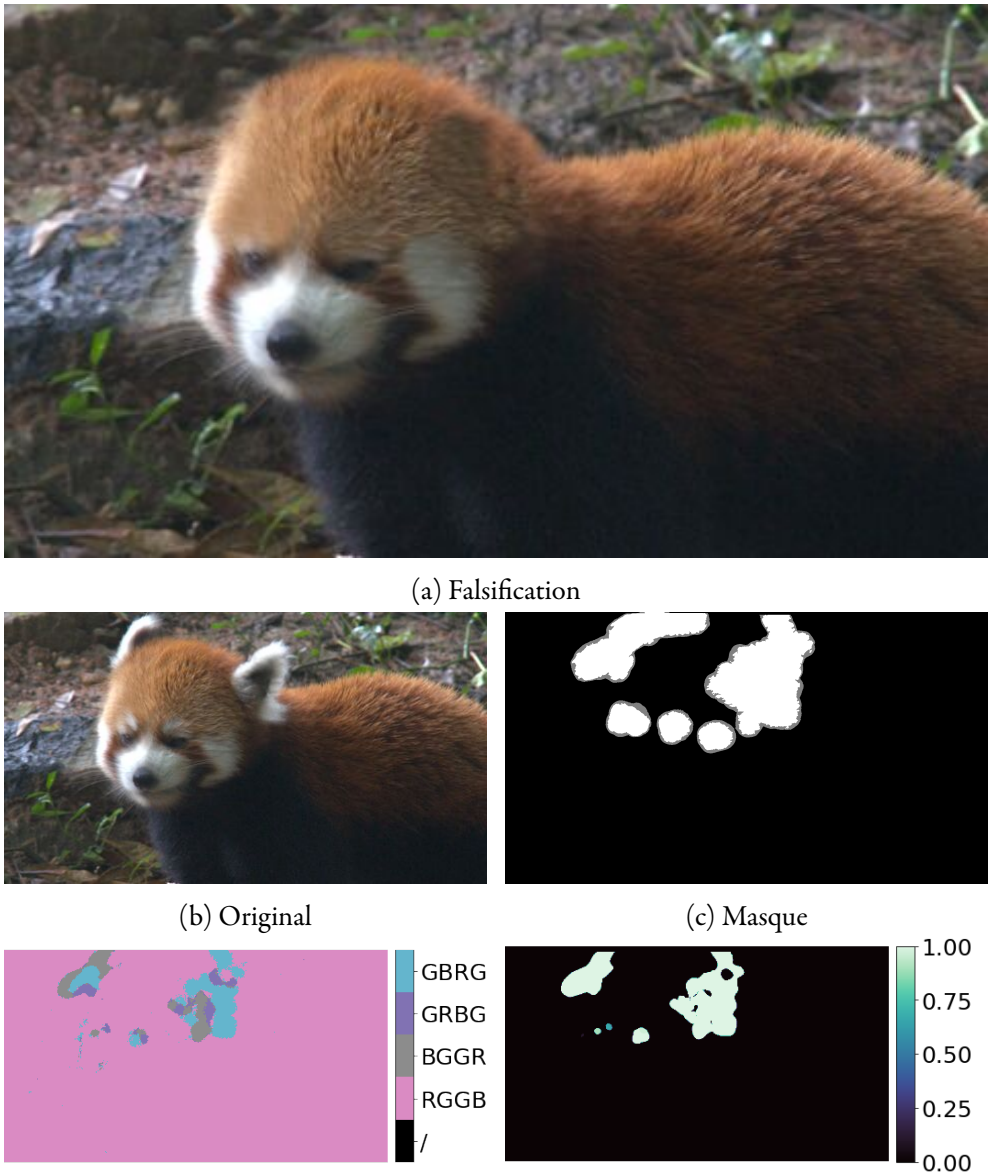


FIGURE 0.16 : Résultats du chapitre 6 (Positional Learning for Demosaicing Analysis). Détections sur le jeu de données CFA Forgeries, présenté dans le chapitre 1 (Non-Semantic Evaluation of Image Forensics Tools). Pour chaque image, dans l'ordre : Image falsifiée, prédictions pixel par pixel pour chacune des 4 grilles, prédictions par bloc pour chacune des 4 grilles, blocs falsifiés détectés, vérité terrain. La mosaïque de l'image et du faux est alignée pour les deux images de la dernière ligne, c'est pourquoi aucune détection ne peut être faite avec notre méthode.

améliorons encore la méthode en ajoutant une couche *a contrario* pour détecter et localiser automatiquement les incohérences importantes dans la mosaïque d'une image. La méthode proposée dans ce dernier chapitre surpasse les méthodes de détection de mosaïque les plus récentes et les méthodes génériques de médecine légale sur des ensembles de données non comprimés. La méthode proposée dans ce dernier chapitre surpasse les méthodes de détection de mosaïque de pointe et les méthodes génériques de forensique sur les images non compressés. Elle reste pertinente sur des images légèrement compressés et apporte un éclairage complémentaire à d'autres méthodes, y compris les modèles génériques basés sur l'apprentissage, qui sont aveugles aux traces de mosaïque que nous analysons.

L'apprentissage interne sur une seule image présente plusieurs défis. Le nombre réduit d'échantillons augmente le risque de surapprentissage ; si le réseau s'adapte trop, il détectera correctement la position du pixel de la contrefaçon, et la contrefaçon ne sera pas détectée. Plus important encore dans notre cas, la compression JPEG pose des difficultés à l'ajustement interne d'une seule image. Le réseau proposé utilise les traces de démosaïquage car elles constituent la principale source d'information sur la position modulo 2 des images. Cependant, l'encodage JPEG comprime l'image en blocs de 8×8, les traces qu'il laisse derrière lui ont donc une forte composante 8-périodique. L'ajustement sur une seule image compressée peut donc détourner le réseau de l'analyse des traces de démosaïquage pour lui faire analyser les traces de compression JPEG. Lors d'un ajustement sur plusieurs images, ce problème serait atténué par les différents alignements entre le CFA et la grille JPEG sur chaque



(a) Falsification

(b) Original

(c) Masque

(d) Grilles détectées

(e) Falsifications détectées

FIGURE 0.17 : Résultats de la méthode du chapitre 7 (Internal Learning to Improve Adaptability) sur une image inpaintée de Korus [42], [43]. La détection locale du motif de démosaïquage permet non seulement de détecter la falsification, mais aussi de montrer les patchs utilisés lors de l'inpainting.

image : un réseau entraîné à détecter des traces JPEG sur un alignement échouerait sur une image alignée autrement. Avec une seule image, l'analyse des traces JPEG peut directement conduire à la position modulo 2 correcte.

Pour éviter ce problème, nous proposons de pré-entraîner le réseau sur des images compressées manuellement à différents alignements JPEG-CFA, comme le montre la figure 0.18. Le réseau apprend ainsi à détecter les artefacts CFA sur la compression JPEG. Bien que cela ne soit pas suffisant pour produire de bons résultats sur les images compressées, cela conduit le modèle sur une meilleure voie avant l'apprentissage interne : Auparavant empêché d'utiliser la position de la grille JPEG, com-

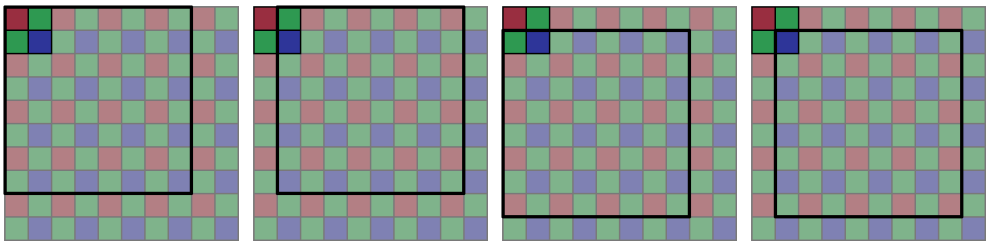


FIGURE 0.18 : La grille JPEG et le motif de Bayer peuvent être alignés de quatre manières différentes. En entraînant simultanément le réseau sur les quatre alignements possibles, nous le forçons à analyser les traces de démosaïquage sur JPEG, au lieu d'utiliser directement les traces JPEG.

mencer à utiliser cette grille aurait un coût immédiat à court terme pour le réseau. L'optimisation locale incite donc efficacement le réseau à ne pas détecter la position de la grille JPEG.

Même avec cette astuce de pré-entraînement JPEG, l'apprentissage interne sur une seule image peut sembler enclin au surapprentissage. Cependant, même si le réseau s'adapte trop à l'image, la formation est effectuée en supposant que l'image complète est authentique. En d'autres termes, l'ajustement fin incite le réseau à conclure que tout est authentique. Par conséquent, si l'image est réellement authentique, les risques de fausse détection sont moindres qu'avec le réseau pré-entraîné ; par conception, le surapprentissage n'induira pas de nouvelles fausses détections. Dans le cas plus intéressant où l'image est effectivement falsifiée, le réseau s'adaptera également au post-traitement pour apprendre les traces de démosaïquage et détecter la position des pixels. Les régions falsifiées dans les images sont généralement petites par rapport à la taille totale. Par conséquent, même si les régions falsifiées orientent le réseau vers une détection correcte de la position de leurs pixels, elles produisent des preuves qui contredisent celles de régions authentiques beaucoup plus grandes. La petite taille et la localité du réseau sont particulièrement importantes ici pour éviter qu'il ne puisse s'adapter à la fois aux régions authentiques et aux régions falsifiées. Bien qu'un certain degré de surapprentissage soit probablement inévitable, son impact est en fait très limité.

À la méthode développée, nous ajoutons la couche *a contrario* déjà introduite dans le chapitre 4 (Linear Estimation of the Demosaicing Algorithm). Ceci permet à notre méthode de détecter des régions incohérentes dans l'image, même si une mosaïque unique ne peut être détectée localement. Les résultats statistiquement non significatifs sont ensuite filtrés sur la base d'un seuil NFA.

Nos expériences montrent que nous pouvons détecter les décalages de motifs de démosaïquage mieux que d'autres méthodes de détection de démosaïquage, et plus généralement que l'état de l'art sur les images falsifiées non compressées. Le seuillage *a contrario* nous permet de seuiller automatiquement les sorties, ce qui limite le nombre de fausses détections et simplifie la combinaison de cette méthode avec d'autres méthodes.

La principale limite de notre méthode, et de la détection du démosaïquage en général, est que les artefacts de démosaïquage sont subtils et situés sur les hautes fréquences. Par conséquent, une forte compression JPEG, ou un sous-échantillonnage,

Méthode	chapitre 2	chapitre 3	chapitre 4	chapitre 5	chapitre 6	chapitre 7
détection *A contrario*	yes	no	yes	no	no	yes
Hypothèse de linéarité	yes	no	yes	no	no	no
Hypothèse d'indépendance des canaux couleurs	yes	yes	no	no	no	no
Ingéniérie inverse	no	no	yes	yes	no	yes
Par apprentissage	non	non	non	non	oui (auto-supervisé)	oui (auto-supervisé, interne)
Autres	–	–	–	Algorithmes similaires	Le démosaïquage laisse des traces 2-périodiques	

TABLE 0.1 : Résumé des propriétés de nos méthodes.

supprimera les artefacts et rendra la détection du démosaïquage impossible. Ceci étant dit, la méthode proposée est suffisamment robuste pour donner des résultats décents à un niveau de qualité de compression de 95, et est encore capable de trouver quelques faux avec un facteur de compression de 90. Cela n'est pas suffisant pour effectuer une détection sur des images de faible qualité telles que celles que l'on trouve sur les médias sociaux. Néanmoins, la méthode fonctionne sur la qualité JPEG généralement élevée fournie par l'appareil. Cela la rend pertinente dans des domaines tels que les concours photographiques, les enquêtes criminelles, les enquêtes sur les inconduites scientifiques ou le journalisme, des tâches où l'authentification des images est souvent nécessaire.

Enfin, nous notons que la méthode proposée – et plus généralement l'analyse de démosaïquage – est totalement complémentaire avec des méthodes forensiques plus génériques telles que Noiseprint [27]. En effet, le chapitre 7 (Internal Learning to Improve Adaptability) montre que ces méthodes sont totalement aveugles aux changements dans le motif de démosaïquage, alors que nous nous concentrons exclusivement sur ceux-ci.

Les propriétés de nos méthodes sont résumées dans le tableau 0.1.

0.4 Publications et présentations

- Q. **Bammey**, R. Grompone von Gioi et J.-M. Morel, 'Automatic Detection of Demosaicing Image Artifacts and Its Use in Tampering Detection', in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, See Chapter 4, 2018, p. 424-429. DOI : 10.1109/MIPR.2018.00091
- Q. **Bammey**, R. Grompone von Gioi et J.-M. Morel, 'Reliable Demosaicing Detection for Image Forensics', in *2019 27th European Signal Processing Conference (EUSIPCO)*, See Chapter 2, 2019, p. 1-5. DOI : 10.23919/EUSIPCO.2019.8903152
- Q. **Bammey**, R. G. v. Gioi et J.-M. Morel, 'An Adaptive Neural Network for Unsupervised Mosaic Consistency Analysis in Image Forensics', in *Pro-*

ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), See Chapter 6, juin 2020

- Q. **Bammey**, R. Grompone von Gioi et J.-M. Morel, 'Image Forgeries Detection through Mosaic Analysis : the Intermediate Values Algorithm', *Image Processing On Line*, t. 11, p. 317-343, 2021. DOI : 10.5201/ipol.2021.355

- Q. **Bammey**, R. G. von Gioi et J.-M. Morel, 'Forgery Detection by Internal Positional Learning of Demosaicing Traces', in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, jan. 2022, p. 328-338

- Q. **Bammey**, T. Nikoukhah, M. Gardella et al., 'Non-Semantic Evaluation of Image Forensics Tools : Methodology and Database', in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, jan. 2022, p. 3751-3760

- **my_tto**

- Q. **Bammey**, M. Colom, M. Gardella et al., 'Sécurité multimédia', in ISTE, juill. 2021, t. 1, chap. Comment reconstruire l'histoire d'une image digitale, et de ses altérations ?, p. 9-50, Directed by William Puech. English version in preparation. DOI : 10.51926/iste.9026.ch1

- Presentation au RESSI 2019 (*Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information*

Introduction

The Internet, digital media, new means of communication and social networks have boosted the emergence of a connected world where perfect mastery over information becomes impossible. Images are ubiquitous and therefore have become an essential part of the news. Unfortunately, they have also become a tool of disinformation aimed at distracting the public from reality.

Manipulation of images is everywhere. Simply removing red eyes from family photos could already be called an image manipulation, whereas it is simply aimed at making a flash image look more natural. Even amateur photographers can easily erase the electric cables of a vacation panorama, correct physical imperfections such as wrinkles on a face, not to mention touch-ups done on models in magazines.

Beyond these mostly-benign examples, image manipulation can lead to falsified results in scientific publications, reports or journalistic articles. Altered images can imply an altered meaning, and can thus be used as fake evidence, for instance to use defamation against someone or report a paranormal phenomenon. More frequently, falsified images are published and relayed on social media, in order to create and to contribute to the spread of fake news.

Making a visually convincing forgery is now within anyone's reach; should one desire so, those realistic falsifications can then be disseminated on online media or social networks [1], fake the outcome of scientific studies, or be presented as forged evidence in a trial.

Recently, deep neural networks have made it possible to generate manipulated images almost automatically, such as the website This Person Does Not Exist[4], which randomly generates faces of people who do not exist while being unexpectedly realistic. Deepfake methods allow, among other things, to replace a face in a video with the one of another person (face swapping).

These new possibilities of image manipulation have been exploited for a long time by governments, criminal organisations and offenders. Stalinist propaganda images can come to mind, in which certain characters who had become undesirable were removed from official photographs. (Figure 0.1).

Today, image manipulation can serve the interests of criminal or terrorist organisations as part of their propaganda (false claims, false events, masking of identification elements, addition of objects). Face swapping and deepfake techniques are also a simple way to undermine the image and privacy of public figures by placing them in compromising photos. The manipulation of images is also a means of exerting coercion, pressure or blackmail against a third party. Manipulated images can also be used to harm companies through disinformation campaigns. Administrative

[4] www.thispersondoesnotexist.com.

Figure 0.1: An example showing how an image has been modified several times in a row, each person disfavoured seeing their image removed from the photo. Only Joseph Staline appears in the four photos.

documents can be falsified in order to obtain official papers, a rental document or a loan from specialised organisations. Face morphing, whose objective is to obtain the photo of a visually "compatible" face from two faces, enables two users to share the same ID in order to deceive an identity check. These manipulations also cause problems to law enforcement. In the past, confessions, testimonies or photographs were enough to prove guilt. Forgery technologies were not sufficiently developed to mislead investigators. Today, these methods are no longer sufficient and law enforcement authorities need innovative scientific tools to be able to present reliable evidence in court.

The digital image is an essential medium of communication in today's world. People need to be able to trust this method of communication. Therefore, it is essential to be able to detect images that have been manipulated.

Yet, even though images are easy to modify in a visually realistic way, those modifications can be difficult to detect automatically.

In Greek mythology, Dolus, the spirit of trickery, attempted to replicate a statue of Aletheia, goddess of truth; in doing so he ran out of clay and left her feet unfinished. When Dolus' master Prometheus gave life to both statues, the fake statue stood out as unable to walk as well as the original; thus the forgery was unconcealed. In the words of Aesop:

> Prometheus, that potter who gave shape to our new generation, decided one day to sculpt a statue of Truth [Aletheia], using all his skill so that she would be able to regulate people's behaviour. As he was working, an unexpected summons from mighty Jupiter called him away. Prometheus left cunning Trickery [Dolus] in charge of his workshop (Trickery had recently become one of the god's apprentices). Fired by ambition, Trickery used the time at his disposal to fashion with his sly fingers a figure of the same size and appearance as Truth with identical features. When he had almost completed

> the piece, which was truly remarkable, he ran out of clay to use for her feet. The master returned, so Trickery quickly sat down in his seat, quaking with fear. Prometheus was amazed at the similarity of the two statues and wanted it to seem as if all the credit were due to his own skill. Therefore, he put both statues in the kiln and when they had been thoroughly baked, he infused them both with life: sacred Truth walked with measured steps, while her unfinished twin stood stuck in her tracks. That forgery, that product of subterfuge, thus acquired the name of Falsehood, and I readily agree with people who say that she has no feet: every once in a while something that is false can start off successfully, but with time the Truth is sure to prevail.

In the same way the fake statue left imperfect footsteps, image forgeries usually leave traces. From the real scene whose photograph is taken to the storage of the captured image on a digital support, many processes take place to create the final image. Each of these operations, imprints its traces onto the image. The set of all those traces form a true signature of the image, akin to a natural watermark. Although usually imperceptible to the naked eye, this signature can usually be detected and analysed, enabling reconstruction of the history of an image, to model the different operations that took place during the creation of the image, as well as their order and parameters. Information about the specific camera pipeline of an image is relevant by itself, in particular because it can guide the restoration of the image. More importantly, it provides an identifying signature of the image.

Indeed, when an image is manipulated, its signature is disrupted. A model of the pipeline that is inconsistent across the whole image is thus often a clue that the image was tampered.

0.5 The image formation pipeline

The main steps in the digital image acquisition process, illustrated in Figure 0.2, will be briefly described in this section. Other important steps, such as denoising, are beyond the scope of this thesis and will therefore not be covered here.

Raw image acquisition

The first step to acquire a raw image consists in counting the number of incident photons over the sensor along the exposure time. There are two different technologies used in camera sensors: Charge Coupled Devices (CCDs) and Complementary Metal-Oxide-Semiconductors (CMOS). Although their operating principles differ, both can be modelled in a very similar way [2]. Both sensors transform incoming light photons into electronic charge which interacts with detection devices to produce electrons stored in a potential light well. When the latter is full, the pixels become saturated.The final step is to convert the analog voltage measurements into digital quantized values.

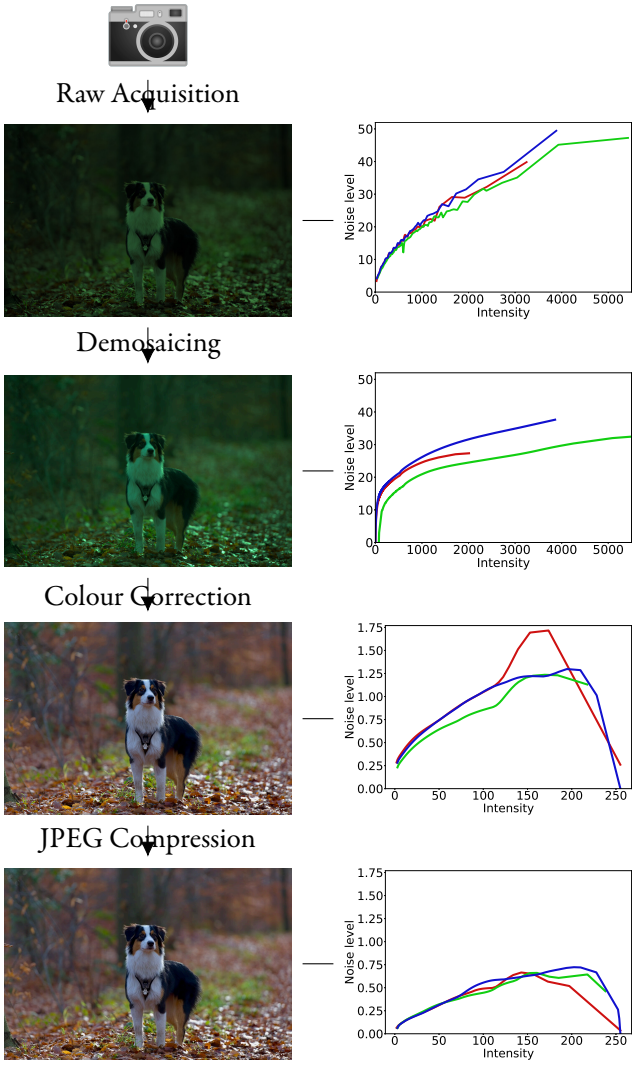


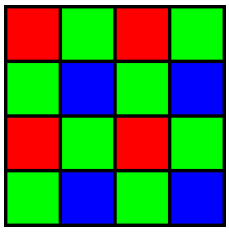
Figure 0.2: Simplified processing pipeline of an image, from its acquisition by the camera sensor to its storage as a JPEG-compressed image. The left column represents the image as it goes through each step. The right column plots the noise of the image as a function of intensity in all three channels (red, green blue). Because each step leaves a specific footprint on the noise pattern of the image, analysing this noise enables us to reverse-engineer the pipeline of an image. This in turn enables us to detect regions of an image which were processed differently, and are thus likely to be falsified.

Demosaicing

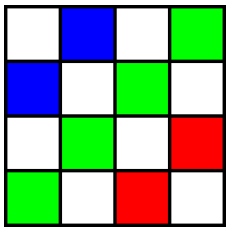
Most cameras cannot see colour directly, because each pixel is obtained through a single sensor which can only count the number of photons reaching it in a certain wavelength range. In order to obtain a colour image, a colour filter array (CFA) is placed in front of the sensors. Each of them only counts the photons of a certain wavelength. As a result, each pixel has a value relative to one colour. By using filters of different colours on neighbouring pixels, the missing colours can then be

interpolated.

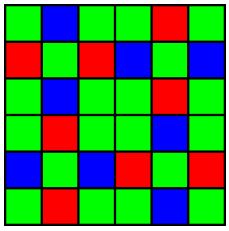
Although other exist, almost all cameras use the Bayer array, see Figure 0.3 for a non-exhaustive list of colour filter arrays.: the Bayer array, which is illustrated in Figure 0.3. This matrix samples half the pixels in green, a quarter in red, and the last quarter in blue. Sampling more pixels in green is justified by the human visual system, which is more sensitive to the green colour.



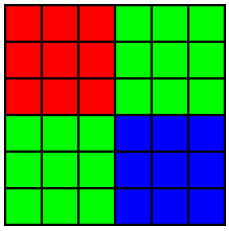
(a) Bayer pattern, the most-commonly used.



(b) RGBW pattern, used in a few Kodak cameras.



(c) Fujifilm X-Trans pattern, used in Fujifilm cameras to reduce colour artefacts **xtrans**.



(d) Nonacell pattern, used in the Samsung Galaxy S20 Ultra. It is similar to the Bayer pattern but uses pixel binning to increase light sensitivity [3].

Figure 0.3: Different colour filter arrays.

Unlike other steps in creating an image, a wide variety of algorithms are used to demosaic an image.

Bilinear interpolation The most simple demosaicing algorithm is bilinear interpolation: Missing values are interpolated by averaging the most direct neighbours sampled in that channel. As the averaging is done regardless of the image gradient, this can cause visible artefacts when interpolated against a strong gradient, such as on image edges.

Hamilton-Adams (HA) demosaicing [55] HA first computes the horizontal and vertical gradients of the image. It then interpolates the green channel – which is easier to interpolate since more values are already known –, using the average of the neighbouring green sampled values and the Laplacian of the red sampled values in the direction with the smoothest gradient. Indeed, a core goal of most demosaicing algorithm is to avoid interpolated across strong edges, as artefacts are very likely in such a scenario. Once the green channel is interpolated, the red (resp. blue) channel is interpolated by using bilinear interpolation to the pointwise difference between the green and the red (resp. blue) channels. While still relatively simple, this algorithm features ideas that are core to many demosaicing algorithms:

- The green channel, which is easier to interpolate, is interpolated first.
- Rather than directly interpolate the red and blue channels, it is often easier to interpolate the difference of these channels with the already-interpolated green channels. Indeed, the difference of channels is smoother. This fact is used for demosaicing detection by Shin, Jeon and Eom [15], which also analyses the colour difference channels rather than the channels themselves.
- High-frequency information is transferred between channels, here with use of a Laplacian operator. This causes difficulties when analysing demosaicing, as analysis performed independently in each channels can be confused by these foreign high frequencies.
- Interpolation should not be done against a strong gradient. When the gradient is strong, it is better to perform the estimation along it than across it.

Alternating Projections (AP) demosaicing [56] AP starts with an initial estimation of the demosaiced image, usually obtained with HA demosaicing. The algorithm alternates between two steps, one in the space-frequency domain and the other in the spatial domain:

1. The image is decomposed into its low and high-frequencies components with a redundant wavelet transform. These high-frequencies components are copied from the green channel to the red and blue channels, as they can be assumed to be similar in all channels (to the contrary of the low frequencies, which can be vastly different). The image is then reconstructed into the spatial domain.
2. The previous step can modify sampled pixels, whose values are already known. The second step enforces the known values by resetting those pixels back to their known values, so the resulting image does not contradict the original mosaic.

This method avoids most colour artefacts, but its working in the frequency domain makes it prone to zipper artefacts. Analysis of this method is thus made more difficult at first, since its artefacts are inherently different from those other methods can yield.

Self-Similarity Driven Demosaicing (SSD) [57] SSD starts with a first estimation of the interpolated image, obtained with HA demosaicing. The estimation is improved by exploiting the image's self-similarity, aggregating the information similar patches. Then, the image is separated into the YUV colour space: the luminance Y is a weighted average of the red, green and blue channels, and the chrominances U and V are the difference of the luminance with the red and blue channels. The chrominance is usually even smoother in natural images than the difference of the green channel with the red and blue channels; SSD thus regularizes it with a simple median filtering.

Directional Linear Minimum Mean-Square-Error Estimation (LMMSE) [58] LMMSE estimates the $R - G$ and $B - G$ colour differences vertically and horizontally using Laplacian interpolation. Then, the two directional estimations are denoised using linear minimum mean square error. A linear combination of the vertical and horizontal estimations is then done in the optimal direction.

Gradient-based threshold-free (GBTF) demosaicing [59] GBTF extends on HA demosaicing. Instead of interpolating either vertically or horizontally, however, it estimates the interpolation in four directions with HA: north, south, west and east. These four interpolations are then weighted and combined. This give the method more flexibility, especially in the cases where no single direction is optimal. Residual Interpolation (RI) [60] improves GBTF by subsituting guided filters to the HA interpolation in all directions. Adaptive Residual Interpolation (ARI) [61] further improves this method, by iterating RI steps on the green and red/blue channels, each time using the improved results of the estimation of the other channels at the previous steps.

CNN-based demosaicing More recently, CNN have been proposed to demosaic an image. In their simplest form, methods like CDM-CNN [62] use an initial estimation of the demosaicing, and train the network to output a better estimation. Demosaicnet [63] even proposes to jointly denoise and demosaic an image, putting an end to the debate of which of the two should be done first for optimal results.

This represents only a small sample of the vast array of existing demosaicing methods. Note that the best methods, such as ARI or CNN-based demosaicing, see little practical use due to their intensive resource requirements – although this may change in the future.

No demosaicing method is perfect – after all, it is a matter of reconstructing missing information – and produce some level of artefacts, although some produce much fewer artefacts than others[5]. Therefore, it is possible to detect these artefacts to obtain information on the demosaicing method applied to the image, which will be the focus of this thesis.

Colour correction

White balance aims to adjust values obtained by the sensors so that they match the colours perceived by the observer by adjusting the gain values of each channel. The way in which white balance adjusts the output depends on the characteristics of the light sources, and is done so that achromatic objects from the real scene are rendered as such [4].

For example, white balance can be achieved by multiplying the value of each channel, so that a pixel that has a maximum value in each channel is found to have the same maximum value 255 in all channels.

[5] Surprisingly, we found that more advanced methods, which produce fewer artefacts, are not always harder to analyse than simpler methods. This is particularly evidenced in Chapter 5 (Demosaicing to Detect Demosaicing).

Then, the image goes through what is known as gamma correction. The charge accumulated by the sensor is proportional to the number of photons incident on the device during the exposure time. However, human perception is not linear with the signal intensity [5]. Therefore, the image is processed to accurately represent human vision by applying a concave function of the form $f_{k,\gamma}(u) = ku^{\frac{1}{\gamma}}$, where γ typically varies between 1.8 and 2.2. The idea behind this procedure is not only to enhance the contrast of the image, but also to encode more precisely the information in the dark areas, which are too dark in the raw image.

Nevertheless, commercial cameras generally do not apply this simple function, but rather a tone curve. Tone curves allow image intensities to be mapped according to precomputed tables that simulate the non-linearity present in human vision.

JPEG compression

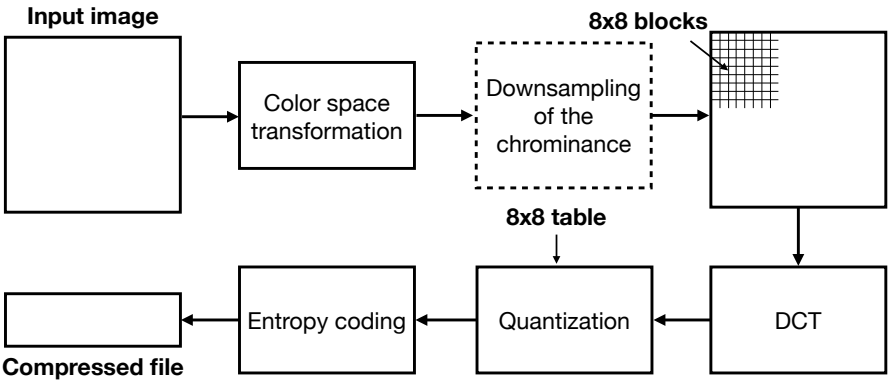


Figure 0.4: JPEG compression pipeline

The stages of the JPEG compression algorithm, illustrated in Figure 0.4, are detailed below. The first stage of the JPEG encoding process consists of performing a colour space transformation from RGB to YC_BC_R where Y is the luminance component and C_B and C_R are the chrominance components of the blue difference and the red difference. Since HVS is less sensitive to colour changes than to changes in luminance, colour components can be subsampled without affecting visual perception too much. The subsampling ratio generally applied is 4:2:0, which means that the horizontal and vertical resolution is reduced by a factor of 2. After the colour subsampling, each channel is divided in blocks of 8×8 and each block is processed independently. The Discrete Cosine Transform (DCT) is applied to each block and the coefficients are quantized.

The JPEG quality factor Q, ranging between 1 and 100, corresponds to the rate of image compression. The lower this rate, the lighter the resulting file, but the more deteriorated the image. A quantization matrix linked to Q provides a factor for each component of the DCT blocks. It is during this quantization step that the greatest loss of information occurs, but it is also this step that allows the most space in memory to be saved. The coefficients corresponding to the high frequencies, of which the HVS struggles to distinguish the variations, are the most quantized, sometimes going so far as to be entirely cancelled.

Finally, as in the example in Figure 0.5, the quantized blocks are encoded without loss to obtain a JPEG file. Each 8×8 block is zig-zagged and the coefficients are arranged in the form of a vector in which the first components represent the low frequencies and the last ones represent the high frequencies.

Lossless compression by RLE coding (range coding) then exploits the long series of zeros at the end of each vector due to the strong quantization of the high frequencies, and then a Huffman code allows for a final lossless compression of the data, to which a header is finally added to form the file.

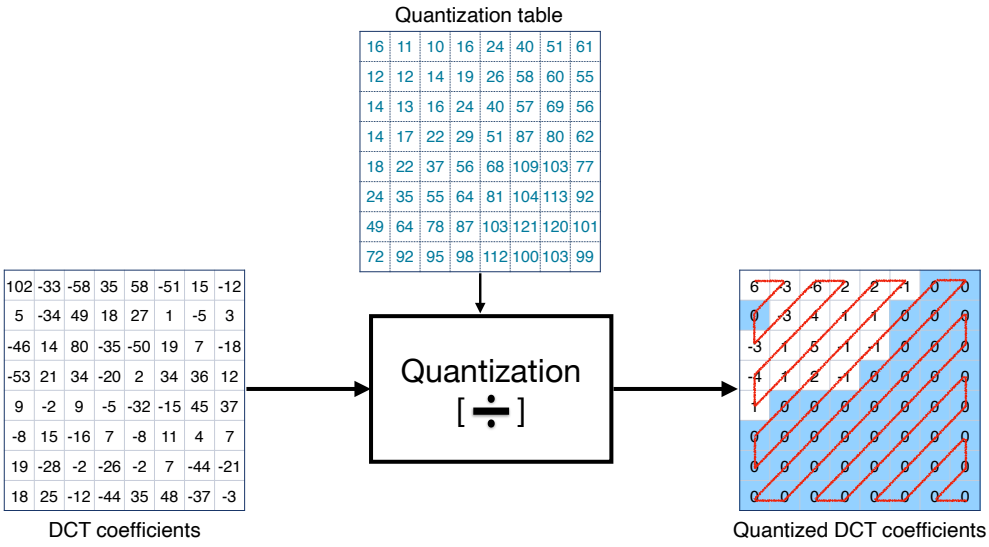


Figure 0.5: An example of the impact of quantization on a DCT block. Each DCT coefficient is quantized by a value found in a quantization matrix. Rounding to the nearest integer results in many of the high frequency coefficients being set to zero. Each block is zig-zagged to be encoded as a vector with a sequence of zeros.

Reverse-engineering the Pipeline to reconstruct the history of an image

Each step of the camera pipeline leaves specific traces on the image. Those traces can be detected and analysed to reveal how a specific image was processed. This knowledge is of utmost importance for image authentication. Knowing what pipeline was used to create an image enables one to link it to a camera or camera model. It is then possible to say whether an image comes from a specific camera.

Reverse-engineering of the image formation pipeline, or part of it, is also at the core of many forgery detection methods. Indeed, tampering with an image often alters its traces as well; the forged region then appears inconsistent with the reverse-engineered model.

0.6 How to detect image forgeries?

The different traces left during the creation of the image can be analysed separately. This thesis will focus on one of such traces; namely demosaicing.

Demosaicing analysis

As explained earlier, the raw image is not a 3-channels colour image, instead, each pixel is sampled in one colour, according to a Colour Filter Array (CFA). The Bayer CFA, shown in Figure 0.3, is by far the most common; demosaicing analysis methods are thus usually designed assuming the image was processed with the Bayer CFA, and we will do the same. Although other CFA exist,their use remains limited. Note that while demosaicing analysis methods are made with the Bayer CFA in mind, most could easily be adapted to other CFA.

To detect image forgeries via demosaicing analysis, two trails can be followed.

One can directly try to detect regions where no traces of demosaicing are present. This can be due to direct manipulation on the image, such as blurring which removes demosaicing traces. Even in the presence of demosaicing traces, if those traces are different, some methods may detect an absence of traces if they strongly differ from the rest of the image. This may be the case, for instance, in splicing forgeries. Such analysis must be done carefully, as it is quite common for natural images to feature no demosaicing traces, for instance in flat regions where perfect demosaicing is possible.

On the other hand, it is also possible to look for shifts in the CFA pattern. As seen in Figure 0.7, in case of a copy-move, either internal or external (splicing), there is a $\frac{3}{4}$ chance that the pasted region's CFA pattern will not be aligned with the original image's. This shift of periodicity can be detected as an inconsistency in the mosaic of the image, and thus evidence of a potential forgery. Almost equivalently, it is also possible to locally detect the mosaic used in the image. With the Bayer CFA, only four patterns are possible, each being a shift of the other patterns. The possible patterns can be further grouped in two by their diagonal, i.e. the green-sampled pixels they share. This can be seen in Figure 0.6.

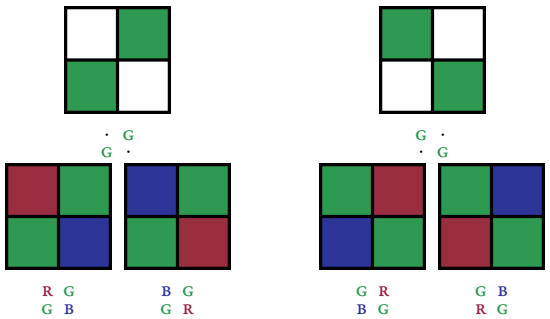
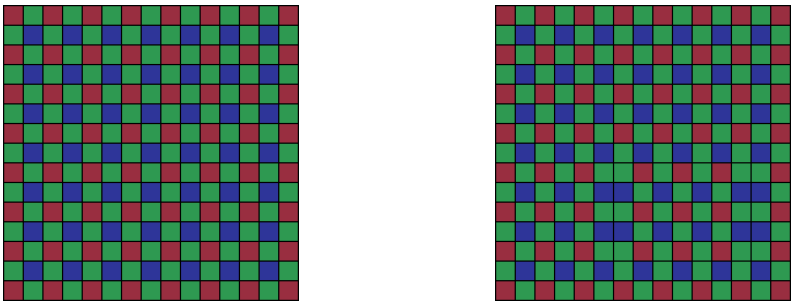


Figure 0.6: The four possible sampling patterns can be grouped by the diagonal on which the green channel was sampled: $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ and $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ share the $\begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}$ diagonal, whereas $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$ and $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$ share the $\begin{smallmatrix}G&\cdot\\\cdot&G\end{smallmatrix}$ one.

Many methods can implicitly do both, although they usually focus on one case. A key difference between both classes is indeed the channels at which one looks. When looking for the absence of demosaicing traces, many methods only consider the green channel. As half the pixels are sampled in green, demosaicing of the green channel is more simple than its red and blue counterparts, and is thus easier to analyse. On the other hand, when looking for shifts in the mosaic, the green channel

is much less informative. Indeed, as seen in Figure 0.6, while there are four patterns in total, they are grouped in two pairs sharing the same diagonal, i.e. sharing their green-sampled pixels. If one only looks at the green channels, shifts between two patterns sharing their diagonal will be missed. In case of a copy-move, the $\frac{3}{4}$ chance of patterns being misaligned will thus fall down to $\frac{1}{2}$ when looking at the green channel. It thus becomes necessary to look at the red and blue channels – even though the green channel is still helpful to distinguish between the two groups of pixels.

In the rest of this section, we review related works on demosaicing analysis in image forensics.



(a) Authentic image (b) Forged image

Figure 0.7: Colours in which pixels are sampled in an authentic and forged image. In the forged area of the second image, there is a $\frac{3}{4}$ probability that the patterns of the authentic and forged area are misaligned, causing a shift in the otherwise-periodic CFA.

In a pioneer paper on demosaicing analysis, Popescu and Farid [6] propose to jointly estimate a linear model for the demosaicing algorithm and detect which pixels have been sampled in a given channel with an expectation-maximization (EM) algorithm. The demosaicing algorithm is estimated on pixels detected as interpolated (i.e. not sampled), as a linear combination of neighbouring pixels in that channel. Sampled pixels are detected as pixels where the linear combination yields a result far from the correct value of the pixel. A pseudo-probability map of each pixel being sampled is then computed. Assuming the linear model is correct, sampled pixels will be correctly detected and there will be a strong 2-periodicity of the map, which can easily be seen as a peak in the Fourier transform of the image. However, in a region which has been altered, the estimated linear model will no longer be correct, either because the demosaicing estimation appears differently or because there are no demosaicing traces left at all. The 2-periodicity peak will thus locally disappear, and can be detected as potential evidence of a forgery. As many other methods looking for inconsistent traces without looking at the specific pattern, the algorithm is only used on the green channel, where demosaicing is usually considered easier to detect.

The same concept is used by Liu, Zhao, Ni *et al.* [7], where the computed pseudo-probability map is analysed with a neural network to distinguish true forgeries from post-processing artefacts such as those coming from JPEG compression. The original method proposes to find regions in which the periodicity disappears; however, if the pseudo-probability map is correct, then shifts in the mosaic's periodicity

should be reflected on it as well. González Fernández, Sandoval Orozco, Garcia Villalba *et al.* [8] extends this method by visualizing the pseudo-probability map with a discrete cosine transform (DCT) instead of a Fourier transform. Shifts in the mosaic's periodicity then appear as a change of sign in the DCT domain, and can thus be detected. The main limitations of these methods come from the linear estimation itself. While this may have been a perfectly reasonable assumption at the time of the original article, most commonly-used demosaicing algorithms nowadays are highly nonlinear, as seen in Section 0.5. Furthermore, they behave differently in different regions of the image, and what may appear as an absence or incoherence of demosaicing can instead be simply a region of a different nature, that the same demosaicing algorithm processed differently.

Ferrara, Bianchi, De Rosa *et al.* [9] also propose to look for regions where demosaicing traces are locally absent. Working on the green channel, they assume to know the global pattern of the image[6]. They apply a fixed predictor, typically bilinear demosaicing, and look at the difference of variance between the two lattices corresponding to supposedly-sampled and interpolated pixels. In the presence of demosaicing artefacts, the variance is significantly higher on the sampled pixels' lattice, which cannot be estimated accurately from interpolated ones. In the absence of such traces, however, the variance is equal in both lattices. Bayesian analysis with Gaussian Mixture Models at multiple block scales then highlight regions where demosaicing traces are absent in an otherwise-demosaiced image.

Kirchner [10] propose to estimate the pattern of an image by performing inverse demosaicing with a simple demosaicing algorithm such as bilinear interpolation. They artificially recreate the mosaic of the image and estimate the masked pixel-channels with the fixed algorithm. Even when the image was not originally demosaiced with the same algorithm, the second demosaicing usually yield the best results in the original pattern, enabling its identification.

Dirik and Memon [11] computes two features related to demosaicing. The first one attempts to classify the pattern of an image. To do this, they perform inverse bilinear demosaicing, as in Kirchner [10], locally and restricting themselves to non-smooth blocks, in which the artefacts are more visible. The second feature is an analysis of noise in the view of the CFA pattern. They estimate the noise of an image with a standard denoising algorithm, and compare the variance of the estimated noise on the different patterns. Indeed, the variance of the noise should be higher on sampled pixels, provided it comes from the real image and was not added manually after demosaicing.

Building on the fact that most demosaicing algorithms avoid interpolated pixels along a steep gradient, Swaminathan, Wu and Liu [12] propose to make three different linear models of the demosaicing algorithms, for a smooth, horizontal or vertical gradient.

Le and Retraint [13] extend their method to forgery detection: working on the green channel, they estimate the three linear models in both possible diagonal patterns to select the pattern yielding the closest error globally. To filter out regions that may bring instability to the detection, they then only work on smooth regions.

[6] Knowing the CFA pattern of a full image is usually not too difficult, especially when only interested in the diagonal pattern such as here.

Features based on the ratio of variances between the sampled and interpolated lattices are constructed. This feature is normally distributed in demosaiced authentic images; a statistical test is then use to locate regions that significantly deviate from the EM-estimated normal distribution.

Choi, Choi and Lee [14] work on a more subtle feature of demosaicing. They notice that interpolated pixels are more likely to be intermediate values compared to their neighbours, whereas sampled pixels are more likely to be local extrema. Counting the number of intermediate values in each possible pattern yields an estimation of the correct pattern. They apply the same idea in [64] to detect colour changes in an image based on demosaicing artefacts. While the original method is limited to identifying the pattern, and does not try to detect forgeries, in Chapter 3 (Intermediate Values Counting for CFA Pattern identification), we will analyse, implement and extend it to do so.

Shin, Jeon and Eom [15] notices that most demosaicing detection methods work separately on each channel, whereas most demosaicing algorithms make extensive use of information in other channels. In order to mimick that behaviour, they propose to work on channel differences, in other words on the maps $R - G$ and $B - G$, where R, G, B represent the red, green and blue channels of the image. They then study the difference of variance between the possible patterns to identify the correct pattern as the one with the highest variance.

Very recently, Park, Moon and Eom [16] try to avoid basing their detection on reinterpolation. Indeed, reinterpolation's results are subject to the suitability of the kernel, which is hard to estimate correctly. In particular, when inter-channel transfers take place during demosaicing, the variance can become lower in the correct pattern, thus causing pattern misidentifications, as we will also notice in Chapters 2 (CFA Identification with Differential Operators) and 3 (Intermediate Values Counting for CFA Pattern identification). They replace the reinterpolation by a singular value decomposition in each channel, so as to remove background information and obtain a more reliable residual.

All in all, current demosaicing analysis methods suffer from two flaws:

- The **large variety of demosaicing methods**, makes the detection difficult with fixed predictors, while the nonlinearity and numerous inter-channel transfers of demosaicing make the estimation of a demosaicing algorithm complex, especially on a single image.

- Demosaicing happens at the very beginning of the camera processing pipeline. As such, demosaicing traces are heavily influenced by anything that takes place after demosaicing. In particular, JPEG compression is quick to remove the highest frequencies of an image, where demosaicing traces lie. As such, demosaicing analysis methods have little to no robustness to JPEG compression, and can only analyse uncompressed images. Image downsampling also removes all traces of demosaicing if the downsampling factor is at least 2, and heavily alters those traces even before that.

The second flaw limits the applicability of demosaicing analysis to high-quality images. While it is unlikely it will be useful to detect forgeries in low-quality images

such as those found on social medias, which are usually downsampled and compress, this analysis is still relevant in other fields where high-quality images can be found, such as criminal investigations or photographic contests. Nevertheless, even in those fields, some degree of robustness to JPEG compression is desirable. Indeed, even high-quality images are often stored with a JPEG quality factor of 95 or even 90, which allows for a reduced image weight for barely any visually perceptible loss.

Other methods for forgery detection

While this thesis will focus on demosaicing traces, other traces can also be exploited to detect image forgeries.

Noise-level-based methods analyse the noise model of images (see Section 0.5) to find regions with a different amount of noise, that could result from tampering. Mahdian and Saic [17] perform local wavelet-based noise level estimation using a median absolute deviation estimator. Lyu et al. [18] relies on the kurtosis concentration phenomenon. More recently, Noisesniffer [19] defines a background stochastic model enabling the detection of local and statistically-significant noise anomalies. These methods can potentially detect a relatively wide variety of forgeries, as each can alter the noise level.

JPEG compression leaves blocking effects and quantization of the DCT coefficient of each block. JPEG forensic tools can thus be divided into two categories. BAG [20] and CAGI [21] analyse blocking artefacts, while other methods analyse the DCT coefficients. More precisely, CDA [22] and I-CDA [23] are based on the AC coefficient distributions, while FDF-A [24] is based on the first digit distribution of AC coefficients. Zero [25] counts the number of null DCT coefficients in all blocks and deduces the grid origin. These methods can only work when the forgery was done after a first JPEG compression, but usually yield very good results when this is the case.

Generic methods All the above-presented method address separately the different kind of traces that can be analysed. The variety of setups before and after forgery makes exhaustiveness difficult, yet results obtained by such specific methods are self-explanatory. Another possibility is to develop multi-purpose tools to classify and/or localize forgeries independently of the setup and forgery type.

Splicebuster [26] computes the noise residual of an image after a high-pass filter, and uses the co-occurences of said residuals as local features characterizing the signature of an image. A Gaussian-uniform mixture model is then used to detect and localize regions where the signature is different from the rest of the image.

Noiseprint [27] extends on Splicebuster by using Siamese networks to extract another noise residual from an image. The network is trained on pairs of patches to extract the same residual if the patches come from the same camera, using the output on one patch as target for the other patch. On patches from different cameras, the model is instead trained to yield different residuals.

Self-consistency [28] analysis also uses a Siamese network with the goal of detecting whether two patches are likely to share the same EXIF metadata, and thus to have been processed with the same pipeline. They make use of N-Cuts segmentation [29] to automatically cluster and detect relevant traces of forgeries.

One can also attempt to detect forgeries directly; for instance ManTraNet [30] is a bipartite end-to-end network, trained to detect image-level manipulations with one part, while the second part is trained on synthetic forgery datasets to detect and localise forgeries in the image.

With such methods, exhaustiveness is theoretically possible. However, results are not self-explanatory and their decisions are harder to justify. Indeed, it is difficult to understand for which reason a model detects an inconsistency. Those methods may also be blind to some kind of traces, for instance, all tested multi-purpose tools are shown in Chapter 1 (Non-Semantic Evaluation of Image Forensics Tools) to be blind to shifts in the CFA mosaic or in the JPEG grid, although a few can, to some extent, detect changes in the CFA algorithm or in the JPEG compression quality. This blindness to shifts in the CFA pattern or JPEG grid is not surprising, as most of those methods make use of convolutional neural networks (CNNs). Indeed, as CNNs are invariant to translation, they cannot detect shifts in a periodic pattern, except perhaps at the very border of the shift.

Furthermore, learning-based methods can be limited by the training data, and may fail to generalize well in uncontrolled scenarios. This is in particular the case for methods such as ManTraNet, which try to directly detect forgeries. There are many different ways to create forgeries, as such a model trained on images forged in one way may fail to generalize to other forgeries.

Datasets for image forensics

Authentic image datasets are necessary to many forgery detection algorithms. They may be used to train models that can be trained on authentic images, such as Noiseprint [27] and Self-consistency [28]. The Raise [31] dataset contains 8156 raw images of various scenarios. The Dresden Image Database [32] provides 16,961 authentic images taken with 27 different cameras. Among them, 1,491 pictures taken with three different cameras, the Nikon D200, D70 and D70s, are provided unprocessed in a RAW format. Specific images may also be useful for experiments. For instance, the noise-free test images dataset [33] contains 16 almost-noiseless images. Those images were carefully downsampled to remove most of the noise, as well as traces from demosaicing. In the absence of previous demosaicing, we can use these images to simulate ourselves various demosaicing algorithms in different patterns. Similarly, the Pixelshift200 dataset [34] contains 210 high-quality images without demosaicing traces, that were obtained by merging 4 images taken almost-simultaneously while shifting the CFA of a camera.

There is also considerable literature proposing datasets of image forgeries, which are needed for the evaluation of forensic tools. An early example is the Columbia Dataset [35], which only contains spliced 128×128 grayscale blocks for which no masks are provided. New benchmarks were proposed in 2009 with CASIA V1.0 and V2.0 [36]. These datasets included splicing and copy-move attacks, with a total of 8000 pristine images and 6000 tampered images. Post-processing was introduced as a counter-forensics technique. MICC F220 and F2000 datasets [37] as well as the IMD dataset [38] provide further benchmarks for copy-move detection. These datasets were constructed in an automatic way. While the first two randomly select

the region of the image to be copy-pasted, IMD dataset performed snippets extraction. Other datasets addressing copy-move forgeries with post-processing counter-attacks are also available [39], [40].

Image forgery-detection challenges are another source of benchmark datasets. The National Institute of Standards and Technology (NIST) organizes, since 2017, an annual challenge for which different datasets are released [41]. It includes automatically and manually generated forgeries of considerable variety, and can thus be useful to evaluate image forgery detection in uncontrolled scenarios.

Some datasets aim at performing forgeries imperceptible to the naked eye. A good example is the Korus dataset [42], [43] which contains 220 pristine images and 220 handmade tampered images targeting object removal or insertion.

The recent DEFACTO dataset [44] is constructed on the MSCOCO dataset [45] and includes a wide range of forgeries such as copy-move, splicing, inpainting and morphing. Semantically meaningful forgeries are generated automatically but with several biases such as copy-pasting objects in the same axis or only performing splicing with simple objects.

Automatic detection of forgeries

Many forgery detection methods do not perform automatic detection. Instead, they yield a heatmap of regions that appear to be forged, and let the user decide on whether the image is indeed forged. However, the user trying to know whether an image is forged does not necessarily have the knowledge to interpret the results. Furthermore, visual analysis of all images is not possible if many images are to be inspected. To make the detection truly automatic, a forgery detection method should ideally provide a binary output of the detection.

In order to achieve this, Self-consistency [28] makes use of n-cuts segmentation to select regions of the heatmap on which the method strongly responds. Le and Retraint [13] use normality tests to decide on the authenticity of the image.

In this context, *a contrario* analysis can prove useful. Introduced by Desolneux, Moisan and Morel [46], it has already been successfully applied to other forgery detection methods [19], [25]. Based on Gestalt theory, this detection paradigm performs automatic thresholding of the data by controlling an upper bound of the number of false alarms (NFA) one might expect. Given a background hypothesis H_0 and a significance function S that represents the significance attributed to an observation, we can compute the p-value of an observation x, that is, the probability of a random observation to be at least as significant as x:

$$p(x) = \mathbb{P}_{y \sim \mathcal{H}_0}(S(y) \geq S(x)). \tag{3}$$

While the p-value provides an idea of the importance of a detection, it does not account for the number of observations that are being made; in presence of a large quantity of observations one can expect spurious detections even with a small threshold of the p-value. The *a contrario* framework instead proposes to put a threshold on the NFA, which is obtained by multiplying the p-value by the number of tests, real or potential. For example, when computing the NFA of rectangles on an image, the number of tests is the total number of possible rectangles in the image – even if not

all rectangles are tested. Associating an NFA to each detection and keeping only detections whose score is below the threshold ε amounts to fix an upper bound ε on the expected number of false alarms in the whole image:

$$\mathbb{E}_{x\sim\mathcal{H}_0}\left(|\{x|\mathrm{NFA}(x) < \varepsilon\}|\right) \leq \varepsilon. \tag{4}$$

The NFA of a detection belongs in $]0, +\infty[$, with scores closer to 0 corresponding to more significant detections. An NFA of 10^{-3}, for instance, means that under the background hypothesis $\mathcal{H}_0$, the expected number of detections at least as significant is at most 10^{-3}. So we expect at most one false detection every 1000 images.

0.7 Outline of this thesis

In this thesis, we focus on the analysis of demosaicing artefacts for forgery detection. Before delving into this, however, we study the problem of the evaluation of forensic tools in Chapter 1 (Non-Semantic Evaluation of Image Forensics Tools).

Chapter 1: How to evaluate forgery detection methods?

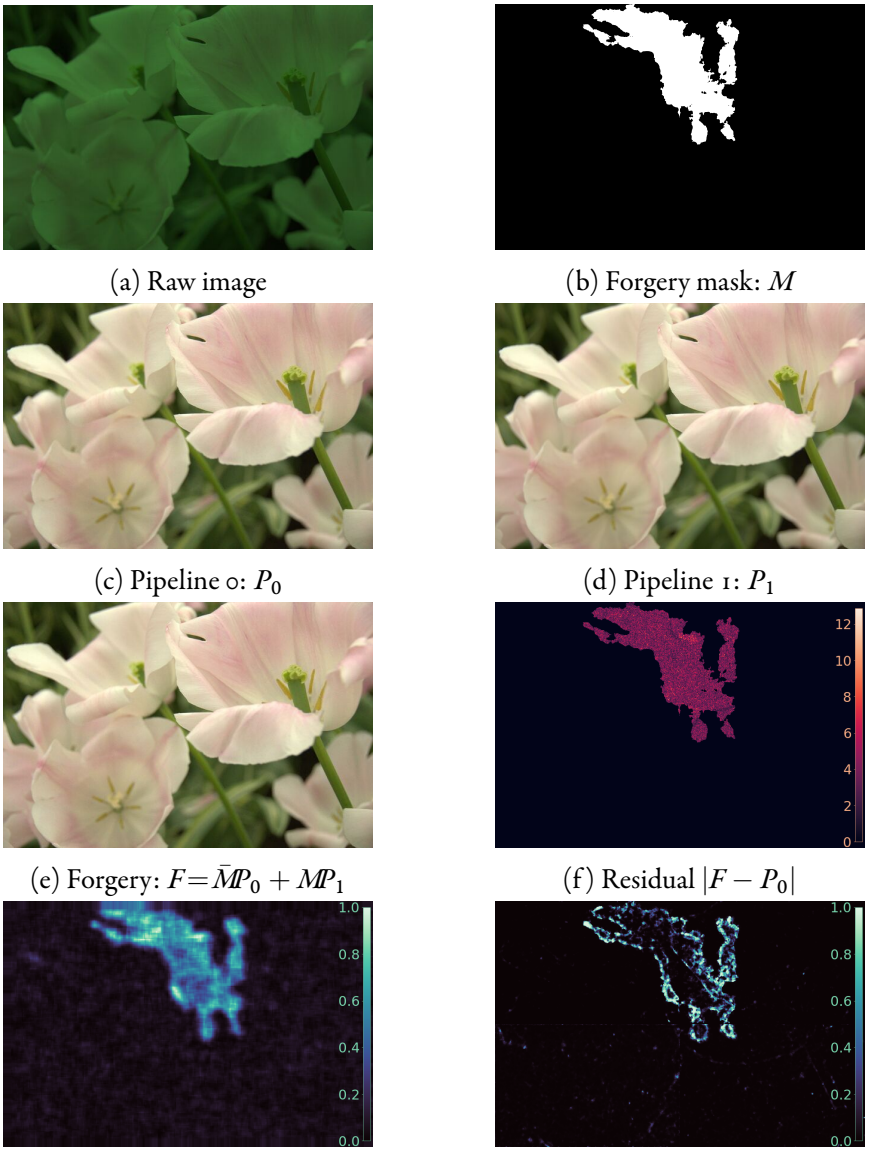
Image forensics algorithms are mainly evaluated by their performance in benchmark challenges. This practice has several limitations: in many cases, the same database is split into training and evaluation data. As a consequence, algorithms are trained and evaluated on images that have gone through similar image processing pipelines, forgery algorithms and anti-forensic tools. Hence, there is no guarantee that such learning-based methods will work in the wild, where those parameters vary much more. Regardless of the variety of the training set, the question arises of whether the forgeries are being detected by trained detectors for semantic reasons, or because of local inconsistencies in the image.

Indeed, while semantic analysis of an image can provide hints, the rigorous proof of a forgery should not be based only on semantic arguments. The situation is similar to the dilemma arising from the observations of Galileo, which contradicted the accepted knowledge of his time. In the words of Bertolt Brecht [47]:

> Galileo: How would it be if your Highness were now to observe these impossible as well as unnecessary stars through this telescope?
> The Mathematician: One might be tempted to reply that your telescope, showing something which cannot exist, may not be a very reliable telescope, eh?

The telescope could have been unreliable, indeed, and a scientific inquiry on the instrument could have been justified. However, concluding, as the Mathematician does, that the telescope was unreliable *just* based on the contents of the observations is not prudent. Similarly, the proof of a forgery must be based on image traces, not on semantic arguments, because the semantics of an image are usually the purpose and not the means of a forgery.

With these considerations in mind, we propose a methodology and a database to evaluate image forensic tools on images where authentic and forged regions only



(a) Raw image (b) Forgery mask: M

(c) Pipeline 0: P_0 (d) Pipeline 1: P_1

(e) Forgery: $F = \bar{M}P_0 + MP_1$ (f) Residual $|F - P_0|$

(g) Noiseprint [27] result (h) ManTraNet [30] result

Figure 0.8: In Chapter 1 (Non-Semantic Evaluation of Image Forensics Tools), different image formation pipelines are applied to the same RAW image to obtain two images, that are combined to obtain a forged image. The authentic and forged regions present different camera pipeline traces, but are otherwise perfectly coherent. The last row shows the result of two forensic tools on this image.

differ in the traces left behind by the image processing pipeline. Using this methodology, we create the *Trace database* by adding various forgery traces to raw images from the Raise [31] dataset, as shown in Fig. 0.8. This procedure avoids the difficulties of producing convincing and unbiased semantic forgeries, which often requires manual work. We create several datasets, each of which corresponding to a specific pipeline inconsistency, such as a different noise level or compression pattern. This gives us insight into the sensitivity of forensic tools to specific traces, and thus highlights the complementarity of different methods.

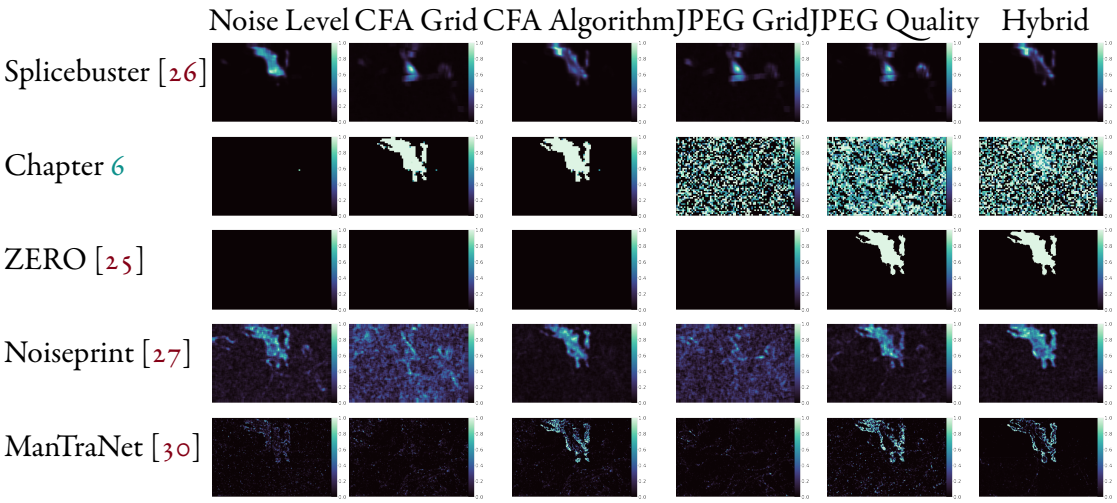


Figure 0.9: Visualization of the results of several methods for one image on all the datasets of Chapter 1. Some methods, such as Noiseprint or the one proposed in Chapter 6, correctly detect the forgeries in the relevant images, but tend to make noise-like false detections in the images for which they cannot see the forgery. Automatically selecting the relevant detections of an algorithm would make it easier to use without needing interpretation, which is why Chapter 7 will do it when extending Chapter 6's method. The image and forgery mask can be seen in Fig. 0.8.

Using the newly created database, we then conduct an evaluation of existing forensic tools, see Fig. 0.9 for an example.

The rest of the thesis focuses purely on demosaicing traces analysis.

Chapters 4 and 3: Direct analysis of demosaicing traces

After analysing the problem of evaluating forensic methods, we begin our search for a demosaicing analysis method. In Chapter 2 (CFA Identification with Differential Operators), we try to highlight sampled pixels from interpolated ones with simple numerical cues from a differential operator. We compare the response of the four patterns pair by pair with a statistical test, to detect significantly impossible patterns. Images are then declared forged when no single pattern is possible everywhere, as seen in Figure 0.10.

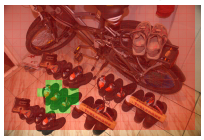
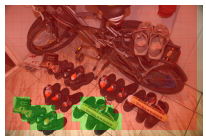
The proposed method is very robust to false detections. However, the linearity and channel-independence of the study makes detections scarce when the demosaicing algorithm is too advanced, or when multiple patterns are present. Accepting results from the strongest channel, instead of refusing to make a detection when two channels are incoherent, can yield a slightly better detections, but full control over the number of false positives can no longer be achieved, as seen in Figure 0.11.

Chapter 2 (CFA Identification with Differential Operators) thus highlights the difficulty of a coherent analysis of demosaicing traces.

In Chapter 3 (Intermediate Values Counting for CFA Pattern identification), we keep the idea to use a simple numerical cue to highlight interpolated pixels from sampled ones. However, we try to follow a more subtle approach. Instead of directly

(a) Found forged regions are in red, and regions where the exact CFA grid was identified are in green

(b) Original image

(c) Regions where the respectively $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$, $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$, $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$ and $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ Bayer matrix positions are deemed possible (in green) or significantly impossible (in red).

Figure 0.10: The method in Chapter 2 (CFA Identification with Differential Operators) detects which patterns are significantly impossible. Images from [38].

highlighting interpolated pixels, we instead use one of their properties: as noticed by Choi, Choi and Lee [14] and highlighted in Figure 0.12, interpolated pixels are more prone to be intermediate values to detect in which pattern an image has been sampled. We analyse, implement and extend their method to detect the CFA pattern. We then use this information to find regions that are inconsistent with the global image. We attribute a confidence score to each detection, which can then be thresholded to provide a binary map of detected forgeries. Although this method does not yield coherent results on a few demosaicing algorithms, it is overall good at detecting the mosaic, at least on uncompressed images, as seen in Figure 0.13.

An online demo for this chapter is available at https://www.ipol.im/pub/art/2021/355/. As part of the Envisu4 project, the method presented here has also been integrated in the forensics browser plugin InVID & WeVerify, an online tool and plugin for journalists and fact-checkers to verify the authenticity of images and check for traces of tampering.

While the method is usable, these issues make it still unsatisfactory. It is possible to detect local inconsistencies even within wrong detections, however erroneous pattern detections cannot be avoided against all demosaicing algorithms without leveraging inter-channel transfers. Furthermore, we wish to improve the robustness to JPEG compression. This leads us to abandon the idea of direct detection of interpolated and sampled pixels. Instead, we try to reverse-engineer the demosaicing algorithm itself, so as to estimate the potential mosaic in which a demosaicing algorithm was effectively used.

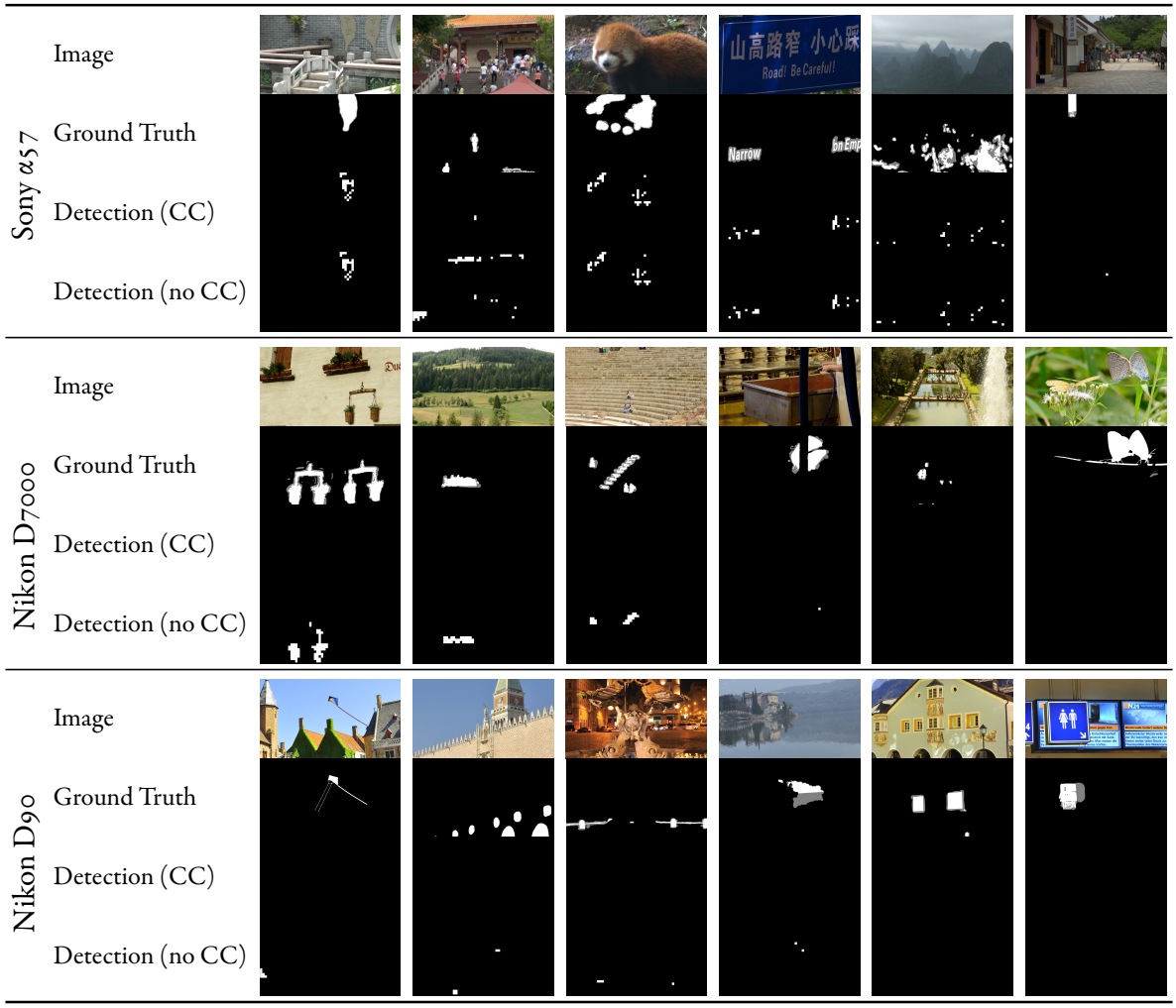


Figure 0.11: Visual results on images of the Korus dataset depending on whether channel consistency (CC) is applied or not, with Chapter 2 (CFA Identification with Differential Operators)'s method. No detections were made on images from the Canon 60D camera, which is thus not shown here. Overall, not enforcing channel consistency enables many more detections to be made, at the cost of a few false positives such as the leftmost image of the Nikon D90 camera, or the second and sixth images of the Sony α57 camera.

Chapters 4 and 5: Reverse-engineering the demosaicing algorithm

Chapters 2 (CFA Identification with Differential Operators) and 3 (Intermediate Values Counting for CFA Pattern identification) explored direct approaches to distinguish sampled pixels, and *in fine* to reveal the correct CFA pattern. However, those approaches were unsatisfactory. While Chapter 3 (Intermediate Values Counting for CFA Pattern identification) provided good results, the simplicity of those direct approaches could not take into account the properties of each specific demosaicing algorithm, leading to erroneous results on images demosaiced by such algorithms. In Chapter 4 (Linear Estimation of the Demosaicing Algorithm), we propose instead to follow a reverse-engineering approach so as to more accurately reflect the specificities of each image and demosaicing algorithm. We create a linear

18	56	94	85	76	96	116	104
49	56	63	52	41	64	88	87
80	56	32	19	6	33	60	70
59	62	66	49	32	59	87	88
38	69	100	79	58	86	114	106
40	51	63	74	85	75	66	63
42	34	26	69	112	65	18	21
38	47	57	75	94	72	50	35

(a) Red channel

139	240	154	16	94	56	72	20
92	131	168	76	72	94	24	43
85	24	100	48	102	224	130	72
60	107	160	68	64	122	200	153
92	184	125	0	50	0	133	108
52	155	156	76	136	117	224	127
146	228	111	12	110	108	107	44
56	114	48	90	184	141	52	90

(b) Green channel

Figure 0.12: Red and green channels of a toy image demosaiced with bilinear interpolation in the $\begin{smallmatrix}R & G\\ G & B\end{smallmatrix}$ pattern. Red values correspond to positions where the value was interpolated. Highlighted cells correspond to pixels that take an intermediate value, i.e. that are not a local extremum among their direct neighbours. While sampled pixels can have intermediate values, many more can be found among interpolated pixels in both the red and green channels. The blue channel, not shown here, behaves similarly to the red one. This observation is at the core of Choi, Choi and Lee [14], whose method is analysed, implemented and extended in Chapter 3 (Intermediate Values Counting for CFA Pattern identification).

estimation of the demosaicing that would be associated with each of the four possible CFA patterns. The residuals from these models yield a local estimation of the CFA pattern in the image. An *a contrario* approach is then applied to find regions whose detected pattern significantly deviate from the rest of the image. We show that while a linear estimation can be sufficient to find the image's CFA pattern, the many nonlinearities of demosaicing, as well as the natural texture of images, make this global and linear model locally unreliable.

Nevertheless, as seen in Figure 0.14, the *a contrario* method deployed here seems satisfactory, although it naturally cannot filter out inconsistencies that are significant in the block votes themselves. The localization masks are not perfect and may require visual analysis of the block votes to precisely localize algorithms; while this can negatively impacts the scoring of the underlying method, it is of little practical importance as long as the proposed approach enables automatic detection of forged images while limiting the number of tests to perform, and thus the computational cost. This approach being good enough, we will not further extend work on *a contrario* analysis. The approach we followed here will be reused with our final method in Chapter 7 (Internal Learning to Improve Adaptability). This enables us to focus on the identification of the CFA mosaic itself in Chapters 5 (Demosaicing to Detect Demosaicing) and 6 (Positional Learning for Demosaicing Analysis), with less focus on the subsequent analysis for forgery detection.

Methods from Chapters 2 (CFA Identification with Differential Operators), 3 (Intermediate Values Counting for CFA Pattern identification) and 4 (Linear Estimation of the Demosaicing Algorithm) have two issues: their inability to properly take into account the inter-channel transfers of demosaicing in Chapters 2 (CFA Identification with Differential Operators) and 3 (Intermediate Values Counting for CFA Pattern identification) and the need to go beyond overly simplistic linear

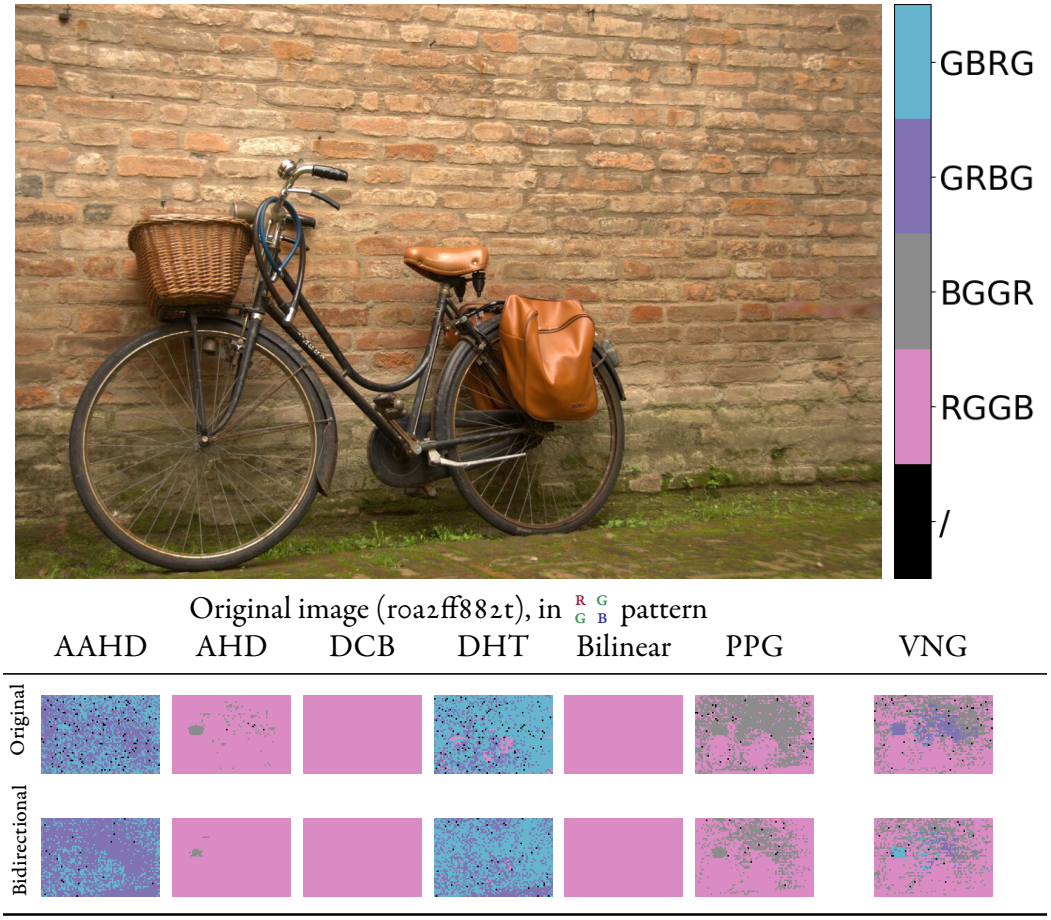


Figure 0.13: Results of Chapter 3 (Intermediate Values Counting for CFA Pattern identification)'s method with the original and proposed filter, on one authentic image with the 7 different demosaicing algorithms. Both methods work perfectly on the DCB- and Bilinear-demosaiced images. With the AHD, PPG and VNG methods, both the original isotropic and the bidirectional filters have trouble discerning between the two patterns sharing the same diagonal, but the bidirectional detection makes fewer mistakes. Periodically textured regions like the basket can create a localized shift in the detected mosaic, which could be mistaken for a forgery. With the AAHD and DHT algorithm, the method consistently detects the wrong diagonal.

assumptions in Chapters 2 (CFA Identification with Differential Operators) and 4 (Linear Estimation of the Demosaicing Algorithm). In Chapter 5 (Demosaicing to Detect Demosaicing), we try to naturally avoid these two issues by using a collection of existing demosaicing algorithms, and instead accept the more natural assumptions that stem from these. Indeed, we can perform demosaicing using each of the four possible mosaics. The correct mosaic is more likely to yield an image closer to the original one, as seen in Figure 0.15. We explore the possibilities of so-called **double demosaicing** to identify the CFA pattern. Local selection of the most suitable demosaicing algorithm enables one to detect the CFA pattern even when the original demosaicing algorithm is unknown. This method naturally avoids the two main issues of the previous chapters, namely their inability to take into account the inter-channel transfers of demosaicing and the need to go beyond overly simplistic

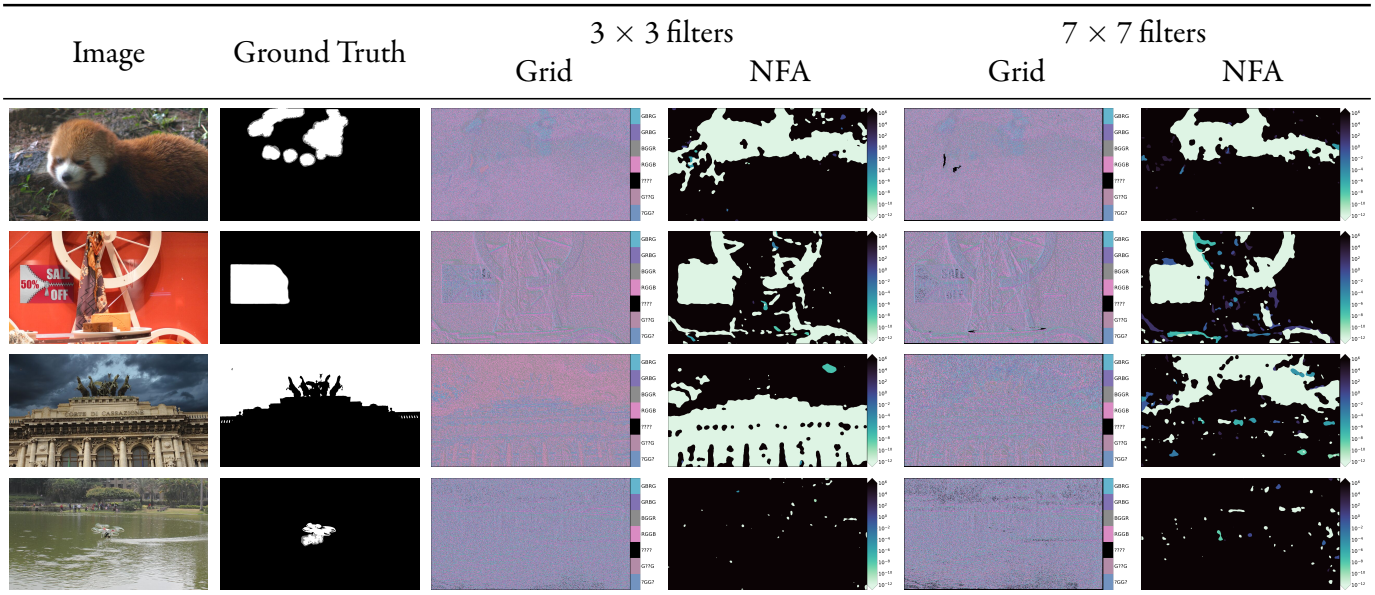


Figure 0.14: Results of Chapter 4 (Linear Estimation of the Demosaicing Algorithm)'s method on Korus images, from top to bottom from the Sony α57, Nikon D7000, Nikon D90, Canon 60D cameras. On the first image, the forgery is accurately detected. Even though the detected mask is too large, a visual analysis of the detected grid enables precise localization of the forgery. On the second image, the forgery is detected, but a second region of the image is mistakenly detected. On the third image, the forgery is inverted when using the smaller filter. Although this negatively affects the score, this is not actually a problem for the detection; as the method still shows the two regions are inconsistent with one another. Finally, on the fourth image, no traces of demosaicing are present. Some regions are incorrectly marked as inconsistent. Those inconsistencies are already present in the detected grids: those defects come from the linear estimation itself, not from the NFA thresholding.

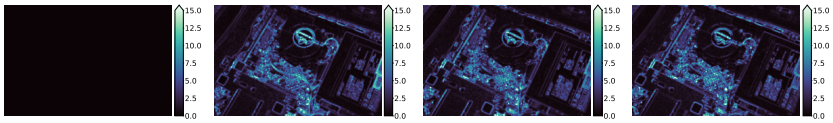
linear assumptions. We simulate the two demosaicing steps to analyse in which conditions double demosaicing can be used to detect the pattern. Furthermore, introducing JPEG compression between the two demosaicing enables us to study the limits of CFA pattern detection on compressed images.

When the initial algorithm is not known, the pattern can be selected with a simple strategy. A list of algorithms can be used, and the algorithm that yields the lowest residual is chosen locally. Expectedly, this strategy yields excellent results when the initial demosaicing algorithm is in the list of tested algorithms. Even when this is not the case, however, this method usually improves on a comparison using a single algorithm.

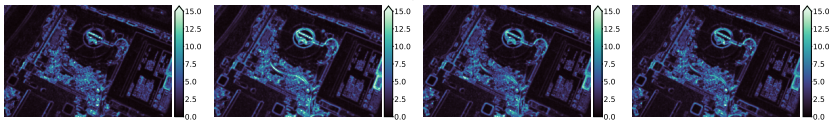
Often, images are found in a compressed state. Still in Chapter 5 (Demosaicing to Detect Demosaicing), we study the robustness of double demosaicing when JPEG compression is applied after the initial demosaicing. Although the detection is made harder by compression, the correct pattern can still be found at the image level even when the original demosaicing algorithm is not known.

The main drawback of this method comes from the difficulty of making reliable

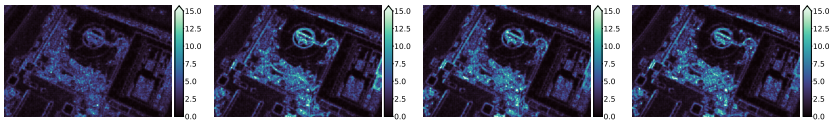
(a) Input image, HA-demosaiced in the $\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}$ pattern.



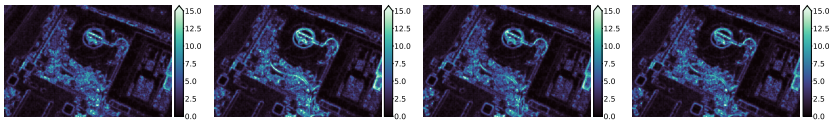
(b) Residuals when the input image is demosaiced again with the same algorithm (HA) in the four positions, from left to right $\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}$ (correct pattern), $\begin{smallmatrix}\mathrm{B}&\mathrm{G}\\\mathrm{G}&\mathrm{R}\end{smallmatrix}$, $\begin{smallmatrix}\mathrm{G}&\mathrm{R}\\\mathrm{B}&\mathrm{G}\end{smallmatrix}$, $\begin{smallmatrix}\mathrm{G}&\mathrm{B}\\\mathrm{R}&\mathrm{G}\end{smallmatrix}$. The residual is zero when the correct pattern is used, making the pattern identification easy.



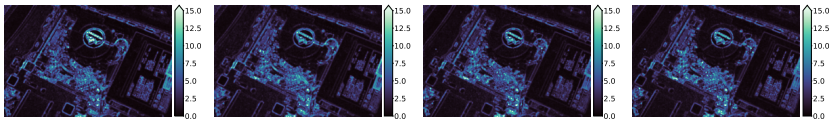
(c) Here, the input image is analysed with a different algorithm (LMMSE). The residual in the correct pattern (left) is no longer zero, but is still weaker than on the incorrect patterns.



(d) The input image is JPEG-compressed ($Q = 90$ before the second demosaicing, with the same (HA) algorithm. Again, although the correct pattern's residual is not zero, it is still weaker than in other positions.



(e) The input image is JPEG-compressed ($Q = 90$ before the second demosaicing, with a different (LMMSE) algorithm. The residual is still weaker with the correct pattern.



(f) Here, the (uncompressed) input image is analysed with yet another algorithm (ARI).

Figure 0.15: **Double Demosaicing:** In Chapter 5 (Demosaicing to Detect Demosaicing), an image was demosaiced with HA and analysed with several algorithms, both uncompressed and after JPEG compression. When the image is not compressed between the two demosaicing operations, and if the algorithm used is the same, the residual will be zero in the correct mosaic pattern, allowing for an easy identification of said pattern. When the image is compressed, the residual is no longer zero in the correct pattern, but is still weaker than in the other patterns as long as the compression is moderate. If a different algorithm is used for analysis, results vary: Analysing the image with LMMSE yields a lower residual in the correct pattern, but this is no longer the case when using ARI, which is not suited to analyse HA-demosaiced images. Residuals are blurred ($\sigma = 1.7$) for better visualization.

Figure 0.16: In Chapter 6 (Positional Learning for Demosaicing Analysis), we train a CNN to detect the horizontal and vertical modulo-2 positions of each pixel, as seen in those targets. CNN are translation-invariant, they do not directly know the position of the pixels; as such they have to rely on external cues such as demosaicing traces. This positional training will thus implicitly train the network to replicate the mosaic of an image and its inconsistencies.

detections at a local scale. While it is possible, with a very high confidence, to detect whether and in which pattern a whole image has been demosaiced, this decision becomes much harder to make locally. In small 32×32 blocks, the contrast of the residual across patterns is still biased towards detecting a lower residual in the correct position; however this contrast is not significantly higher than the contrast on images without a mosaic, in which no detection should be made. This is especially the case on highly-compressed images.

Overall, this method can be used to analyse the image at a global scale, helped by its surprisingly good robustness to JPEG compression. It is not yet possible, however, to use it locally. Local analysis would be needed to detect mosaic inconsistencies and thus potential forgeries.

This fourth chapter in the quest for reliable evidence of demosaicing traces again fails to yield a universal answer. Even the most natural assumption of closeness between different demosaicing algorithms is not always valid locally, especially under strong compression. This is understandable; even though different algorithms may behave closely, this is no longer necessarily the case if one of the two algorithms is followed by heavy post-processing.

All in all, none of the tested models until now are perfectly satisfactory and reliable. Direct detection, while able to identify the grid in many cases, is flawed against several demosaicing algorithms, and is hampered by the difficulty of designing features that reflect demosaicing algorithms' features. Reverse-engineering of the demosaicing algorithm of a specific image is not flawless either; even the all in all very natural assumption that the demosaicing algorithm will locally behave closely to at least one known algorithm is not always true, and fails to do miracles on highly-compressed images.

Chapters 6 and 7: Positional learning to replicate the image's mosaic and its disruptions

Faced with the difficulty of our problem, we decide to go back to the most bare and natural assumption about demosaicing traces: they feature a strong 2-periodic component. Without any other assumption, we propose to detect the phase of that

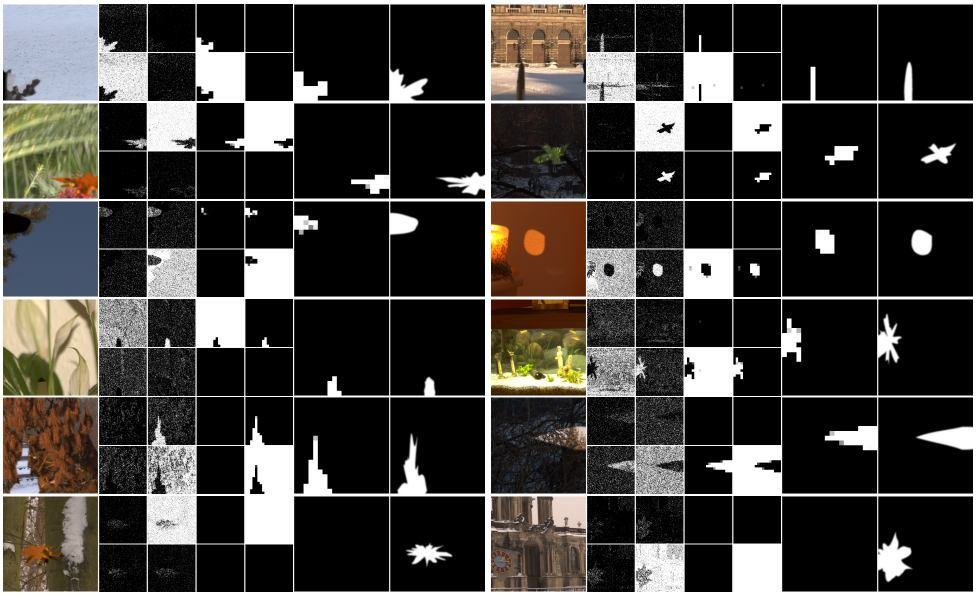


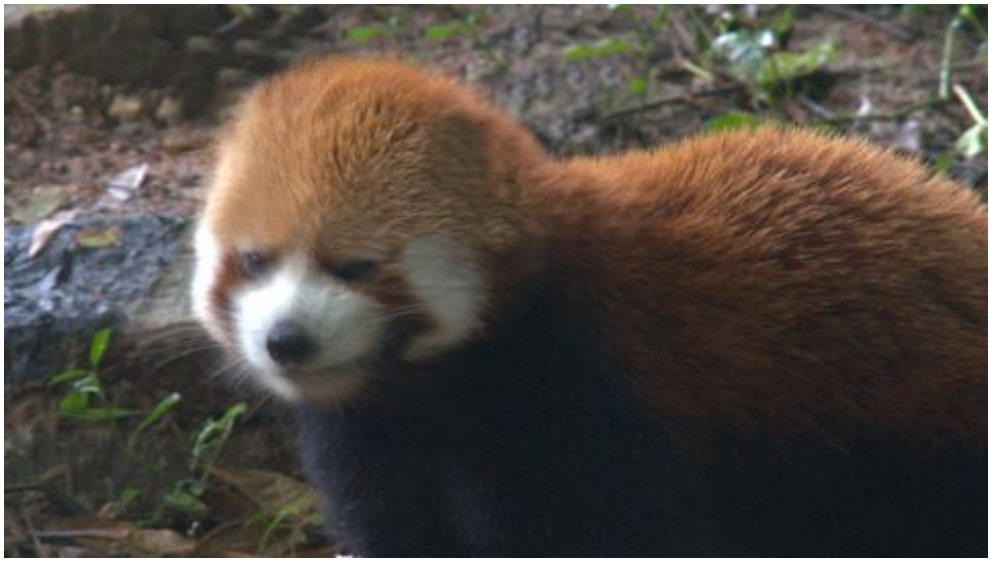
Figure 0.17: Results from Chapter 6 (Positional Learning for Demosaicing Analysis). Detections on the CFA Forgeries dataset, presented in Chapter 1 (Non-Semantic Evaluation of Image Forensics Tools). For each image, in order: Forged image, pixelwise predictions for each of the 4 grids, blockwise predictions for each of the 4 grids, detected forged blocks, ground truth. The mosaic of the image and the forgery is aligned for the two images in the last row, which is why no detection can be made with our method.

component. In order to do this, we introduce positional learning. Leveraging the translation-inheritance and the high representative power of convolutional neural networks (CNN), we train one to detect the modulo-$(2, 2)$ position of each pixel, as seen in Figure 0.16. Implicitly, the CNN will rely on demosaicing traces to provide its output; said output will thus mimic the phase of the 2-periodic component.

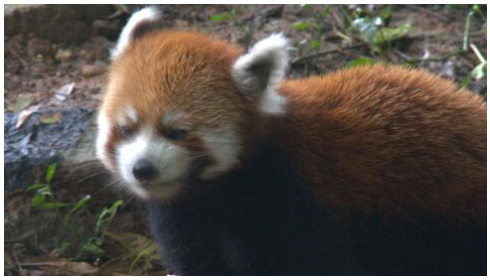
When a forgery disrupts the image's mosaic, the network's output reflects this disruption, enabling the detection of the forgery as errors in the output. This method is fully self-supervised, requiring only authentic images for training. Furthermore, given several similar images whose authenticity is not clear, it is possible to fine-tune the model on the images to increase robustness to JPEG compression. In practice, however, it is rare to have a large quantity of similar images to analyse.

This is why, in Chapter 7 (Internal Learning to Improve Adaptability), we show that such a network can be fine-tuned on a single, potentially forged image, to adapt to it. Doing so greatly increases robustness to JPEG compression and other post-processing. We further improve the method by adding a *a contrario* layer to automatically detect and localize significant inconsistencies in an image's mosaic. The method proposed in this final chapter beats state-of-the-art mosaic detection methods and generic forensic methods alike on uncompressed datasets. It remains relevant on slightly-compressed datasets, and provides a complementary insight to other methods, including generic learning-based models, which are blind to the mosaic traces we analyse.

Single-image internal learning brings several challenges. Fewer sample makes

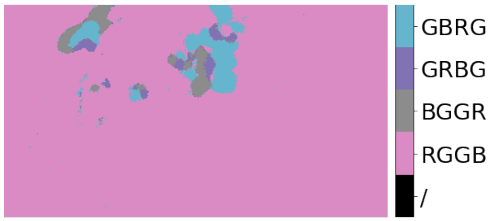


(a) Forged image



(b) Original image

(c) Forgery mask



(d) Detected demosaicing grids

(e) Detected forgeries

Figure 0.18: Results of Chapter 7 (Internal Learning to Improve Adaptability)'s method on an inpainted image from the Korus [42], [43] dataset. Local detection of the demosaicing pattern not only enables detection of the forgery, but also shows the patches used during inpainting.

overfitting more likely, yet if the network overfits it will correctly detect the position of the forgery's pixel, and the forgery will not be detected. More importantly in our case, JPEG compression poses difficulty to single-image fine-tuning. The proposed network uses demosaicing traces because those are the primary source of information on the modulo 2 position of the images. However, JPEG encoding compresses the image in 8×8 blocks, the traces it leaves behind thus have a strong 8-periodic component. Fine-tuning on a single compressed image may thus divert the network from looking at demosaicing traces to make it analyse JPEG compression traces instead. When fine-tuning on multiple images, this would be alleviated

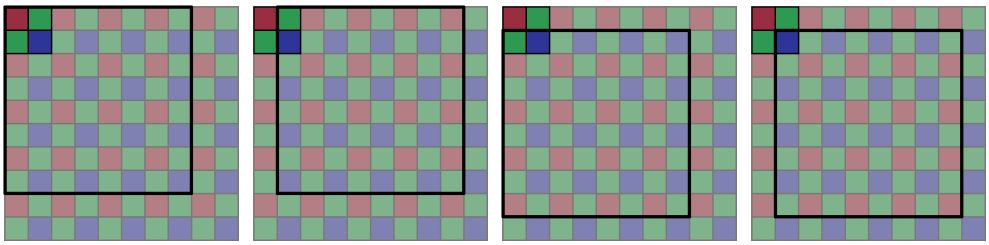


Figure 0.19: The JPEG grid and the Bayer pattern can be aligned in four different ways. By simultaneously training the network on the four possible alignments, we force it to analyse demosaicing traces over JPEG, instead of directly using the JPEG traces.

by the different alignments between the CFA and the JPEG grid on each image: a network trained on detecting JPEG traces on one alignment would fail on image aligned otherwise. With a single image, analysis of JPEG traces can directly lead to the correct modulo-2 position.

To avoid this problem, we propose to pretrain the network on manually-compressed images at different JPEG–CFA alignment, as seen in Figure 0.19. The network thus learns to detect CFA artefacts over JPEG compression. While this is not enough to produce good results on compressed images, it leads the model on a better track before internal learning: Previously prevented from using the JPEG grid position, starting to use this grid would have an immediate short-term cost for the network. Local optimization thus effectively incentivizes the network not to detect the JPEG grid position.

Even with this JPEG pretraining trick, single-image internal learning can seem prone to overfitting. However, even if the network overfits on the image, training is done on the hypothesis that the full image is authentic. In other words, fine-tuning incentivizes the network to conclude that everything is authentic. As a consequence, if the image is actually authentic, the risks of making a false detection are lower than with the pretrained-network; by design, even overfitting will not induce new false detections. In the more interesting case where the image is indeed forged, the network will also adapt to the post-processing to learn demosaicing traces and detect the position of pixels. Forged regions in images are usually small compared to the total size. As a consequence, even though the forged regions would steer the network towards detecting their pixels' positions correctly, they would produce evidence contradicting that of much larger authentic regions, and the network should thus not learn too much from forged regions. The small size and locality of the network is particularly important here to prevent it from being able to adapt to both, the authentic and the forged regions. While some amount of overfitting is probably unavoidable, its impact is actually shown to be very limited.

To the developped method, we add the *a contrario* layer already introduced in Chapter 4 (Linear Estimation of the Demosaicing Algorithm). This enables our method to detect inconsistent regions in the image, even if a single mosaic cannot be detected locally. Statistically-insignificant results are then filtered out based on a NFA threshold.

Our experiments show we can detect demosaicing pattern shifts better than

Method	Chapter 2	Chapter 3	Chapter 4	Chapter 5	Chapter 6	Chapter 7
A contrario detection	yes	no	yes	no	no	yes
Linear hypothesis	yes	no	yes	no	no	no
Inter-channel independence hypothesis	yes	yes	no	no	no	no
(self-supervised + internal)						
Adaptive to the image (reverse-engineering)	no	no	yes	yes	no	yes
Learning-based	no	no	no	no	yes (self-supervised)	yes
Other hypotheses	–	–	–	Similarity of algorithms	Demosaicing traces have a 2-periodic component	

Table 0.2: Summary of the properties of our methods.

other demosaicing detection methods, and more generally beats the state of the art on uncompressed forged images. The *a contrario thresholding* enables us to automatically threshold the outputs, limiting the number of false detections and simplifying combination of this method with other methods.

The main limitation of our method, and of demosaicing detection in general, is that demosaicing artefacts are subtle and located on the high frequencies. As a consequence, a strong JPEG compression, or downsampling, will remove the artefacts and make demosaicing detection impossible. That being said, the proposed method provides enough robustness to yield decent results at a compression quality level of 95, and is still able to find a few forgeries with a compression factor of 90. This is not enough to perform detection on low-quality images such as those found on social medias. Nevertheless, the method works on the usually high JPEG quality provided by cameras. This makes it relevant in fields such as photographic contests, criminal investigations, scientific misconduct investigations, or journalism, tasks where image authentication is often needed.

Finally, we note that the proposed method – and more generally demosaicing analysis – is fully complementary with more generic forensic methods such as Noiseprint [27]. Indeed, Chapter 7 (Internal Learning to Improve Adaptability) shows that such methods are entirely blind to shifts in the demosaicing pattern, whereas we focus exclusively on those.

The properties of our methods are summarized in Table 0.2.

0.8 Publications and presentations

- Q. **Bammey**, R. Grompone von Gioi and J.-M. Morel, 'Automatic detection of demosaicing image artifacts and its use in tampering detection', in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, See Chapter 4, 2018, pp. 424–429. DOI: 10.1109/MIPR.2018.00091

- Q. **Bammey**, R. Grompone von Gioi and J.-M. Morel, 'Reliable demosaicing detection for image forensics', in *2019 27th European Signal Processing Con-*

ference (EUSIPCO), See Chapter 2, 2019, pp. 1–5. DOI: 10.23919/EUSIPCO.2019.8903152

- Q. **Bammey**, R. G. v. Gioi and J.-M. Morel, 'An adaptive neural network for unsupervised mosaic consistency analysis in image forensics', in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, See Chapter 6, Jun. 2020

- Q. **Bammey**, R. Grompone von Gioi and J.-M. Morel, 'Image Forgeries Detection through Mosaic Analysis: the Intermediate Values Algorithm', *Image Processing On Line*, vol. 11, pp. 317–343, 2021. DOI: 10.5201/ipol.2021.355

- Q. **Bammey**, R. G. von Gioi and J.-M. Morel, 'Forgery detection by internal positional learning of demosaicing traces', in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2022, pp. 328–338

- Q. **Bammey**, T. Nikoukhah, M. Gardella *et al.*, 'Non-semantic evaluation of image forensics tools: Methodology and database', in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2022, pp. 3751–3760

- **my_tto**

- Q. **Bammey**, M. Colom, M. Gardella *et al.*, 'Sécurité multimédia', in ISTE, Jul. 2021, vol. 1, ch. Comment reconstruire l'histoire d'une image digitale, et de ses altérations ?, pp. 9–50, Directed by William Puech. English version in preparation. DOI: 10.51926/ISTE.9026.ch1

- Presentation to RESSI 2019 (*Rendez-vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information* / Meetings on Research and Teaching of Information Systems Security)

Chapter 1

Non-Semantic Evaluation of Image Forensics Tools: Methodology and Database

Abstract

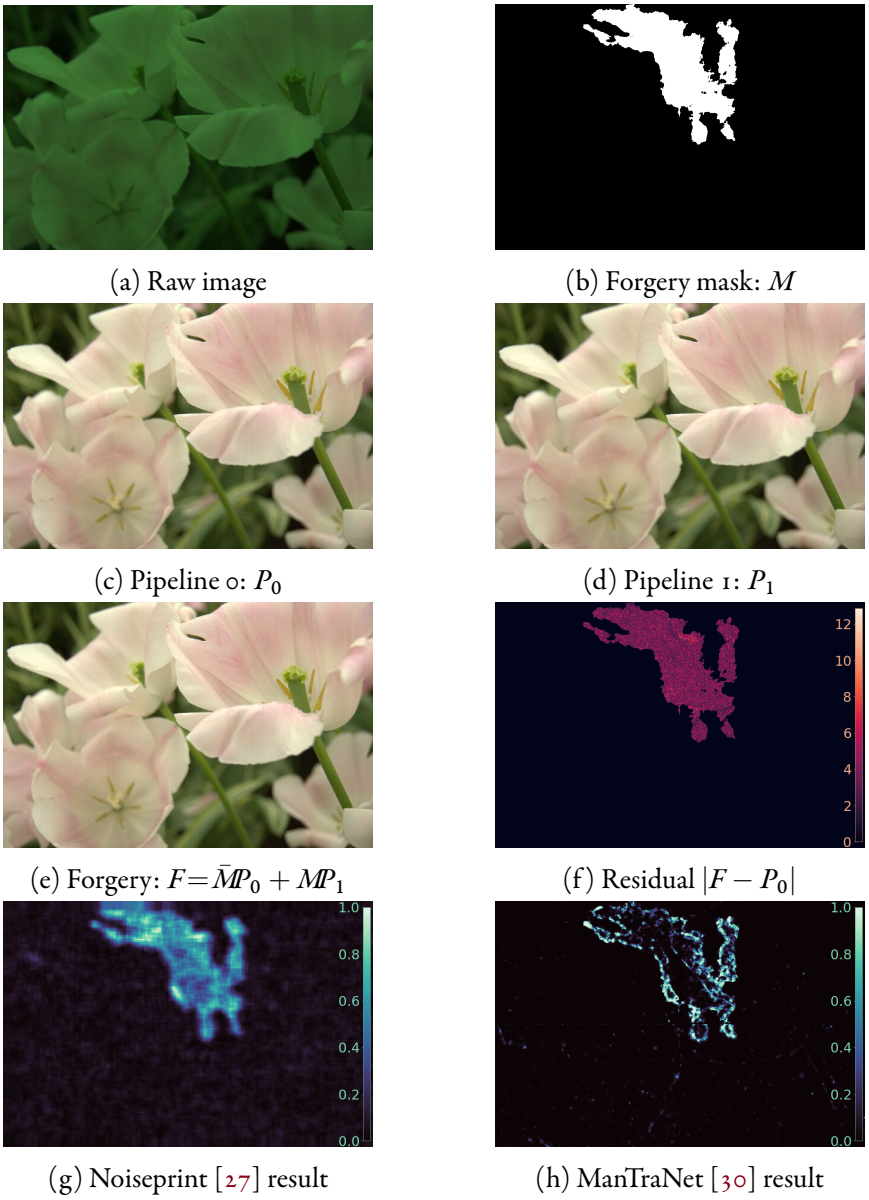
We propose a new method for evaluating image forensics tools, that characterizes what image cues are being used by each detector. Our method effortlessly creates arbitrarily large datasets of carefully tampered images where one to several detection cues are present. Starting with raw images, we alter aspects of the image formation pipeline inside a mask, while leaving the rest of the image intact. This does not change the image's interpretation; we thus call "non-semantic" such alterations, as they give no semantic cues to detectors. This method avoids the painful and often biased creation of convincing semantics. All aspects of image formation (noise, CFA, compression pattern and quality, etc.) can vary freely and independently in both the authentic and tampered parts of the image.

Based on this methodology, we create a database and conduct an evaluation of the main state-of-the-art image forensics tools, where we characterize the performance of each method with respect to each detection cue.

The proposed database tackles three different traces: Demosaicing, noise level and JPEG compression. The datasets related to demosaicing traces will be used for evaluation in Chapters 2 (CFA Identification with Differential Operators), 3 (Intermediate Values Counting for CFA Pattern identification), 4 (Linear Estimation of the Demosaicing Algorithm), 6 (Positional Learning for Demosaicing Analysis) and 7 (Internal Learning to Improve Adaptability).

1.1 Introduction

Digital images play an extensive role in our lives and forgeries are present everywhere [65]. Creating visually realistic image alterations is easy. Yet these modifications leave behind cues: each operation has an impact on the image in the form of a particular trace. Some forgery detection tools aims at detecting a specific trace in a suspicious image by finding local inconsistencies, while other methods, usually learning-based, are more generic. Semantic analysis of an image can provide



(a) Raw image (b) Forgery mask: M

(c) Pipeline 0: P_0 (d) Pipeline 1: P_1

(e) Forgery: $F = \bar{M}P_0 + MP_1$ (f) Residual $|F - P_0|$

(g) Noiseprint [27] result (h) ManTraNet [30] result

Figure 1.1: Different image formation pipelines are applied to the same RAW image to obtain two images, that are combined to obtain a forged image. The authentic and forged regions present different camera pipeline traces, but are otherwise perfectly coherent. The last row shows the result of two forensic tools on this image.

hints, but the rigorous proof of a forgery should not be based only on semantic arguments. The situation is similar to the dilemma arising from the observations of Galileo, which contradicted the knowledge of his time. In the words of Bertolt Brecht [47]:

> Galileo: How would it be if your Highness were now to observe these impossible as well as unnecessary stars through this telescope?
> The Mathematician: One might be tempted to reply that your telescope, showing something which cannot exist, may not be a very reliable telescope, eh?

The telescope could have been unreliable, indeed, and a scientific inquiry on the instrument could have been justified. However, concluding, as the Mathematician does, that the telescope was unreliable *just* based on the contents of the observations is not prudent. Similarly, the proof of a forgery must be based on image traces, not on semantic arguments, because the semantics of an image are usually the purpose and not the means of a forgery.

Image forensics algorithms are mainly evaluated by their performance in benchmark challenges. This practice has several limitations: in many cases, the same database is split into training and evaluation data. As a consequence, algorithms are trained and evaluated on images that have gone through similar image processing pipelines, forgery algorithms and anti-forensic tools. Hence, there is no guarantee that such learning-based methods will work in the wild, where those parameters vary much more. Regardless of the variety of the training set, the question arises of whether the forgeries are being detected by trained detectors for semantic reasons, or because of local inconsistencies in the image.

With these considerations in mind, we propose a methodology and a database to evaluate image forensic tools on images where authentic and forged regions only differ in the traces left behind by the image processing pipeline. Using this methodology, we create the *Trace database* by adding various forgery traces to raw images from the Raise [31] dataset, as shown in Fig. 1.1. This procedure avoids the difficulties of producing convincing and unbiased semantic forgeries, which often requires manual work. We create several datasets, each of which corresponding to a specific pipeline inconsistency, such as a different noise level or compression pattern. This gives us insight into the sensitivity of forensic tools to specific traces, and thus highlights the complementarity of different methods.

Our contribution is twofold:

1. we create a database of "fake" images with controlled inconsistencies in their formation pipeline,

2. using this database, we conduct an evaluation of existing forensic tools.

Most recent forgery-detection datasets start from pristine images and perform several sorts of forgeries on them [66]. Since the creation of early datasets [35], [36], [67], the number of tampering techniques has increased to include new ones such as colorization [68], inpainting [44], [68] and morphing [44], [69]. Post-processing and counter-forensic techniques have been increasingly used to produce visually imperceptible forgeries; but such approaches may also introduce detectable traces.

Efforts have also been made to automatically obtain large datasets. Yet, the resulting forged images are either semantically incorrect [37], [38] or biased [44]. Both scenarios pose problems for training neural networks, which risk overfitting on the forgeries' methods and semantic content.

The variety of forgery methods makes the evaluation of forensic tools difficult to interpret, as the performance depends on the suitability of the detection tool for the specific forgery method. In quantitative experiments, using multiple datasets, and especially datasets with varied forgeries, helps assess the quality of a forensic tool.

However, those results also become harder to interpret. On the other hand, while results using the proposed database will not be reflective of uncontrolled scenarios, they help precisely identify which traces a forensic tool can and cannot detect.

1.2 Image formation pipeline

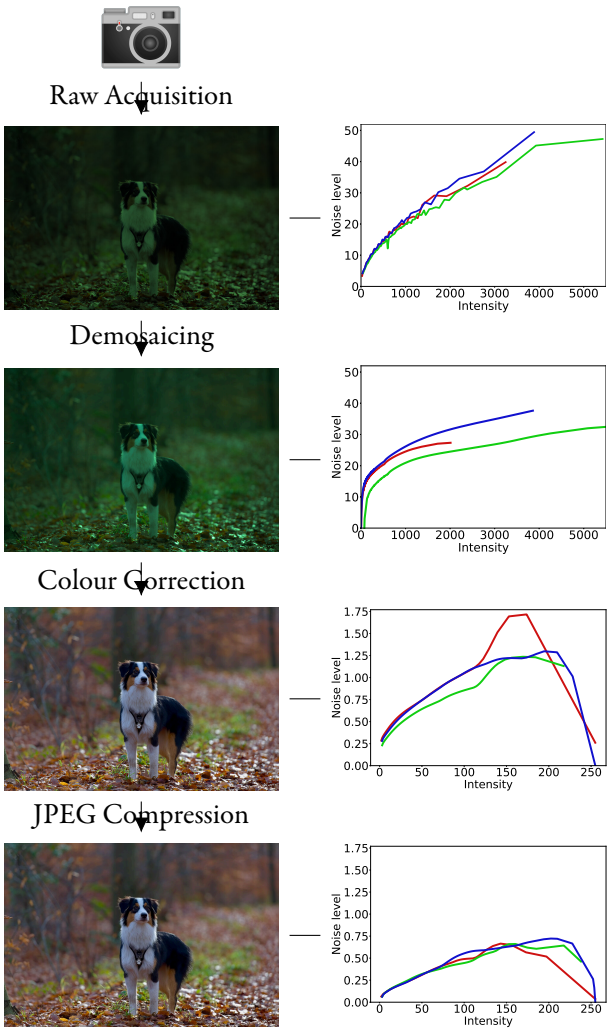


Figure 1.2: Evolution of the noise curves when passing through the successive steps of a (simplified) image processing pipeline.

Figure 1.2 summarises the image processing pipeline [70] and shows how the noise curves change at its different steps.

Raw image acquisition The value at each pixel can be modelled as a Poisson random variable [71]. Noise variance at this step thus follows an affine relation $\sigma^2 = A + Bu$ where u is the intensity of the ideal noiseless image and A and B are constants (see Fig. 1.2(a)). Furthermore, given the nature of the noise sources at this

step, noise can be accurately modelled as uncorrelated, meaning that noise at one pixel is not related with the noise at any other pixel.

Demosaicing Most digital cameras are equipped with a single sensor array. In order to obtain a colour image, a colour filter array (CFA) is placed in front of the sensor to split incident light components according to their wavelength. The raw image obtained from the sensor therefore is a mosaic containing a single colour component per pixel: red, green, or blue. Demosaicing methods interpolate the missing colours at each pixel to reconstruct a full colour image. After demosaicing (Fig. 1.2b), each channel has a different noise curve, and noise becomes spatially correlated.

Colour Correction In order to obtain a faithful representation of the colours as perceived by the observer, white balance adjusts colour intensities in such a way that achromatic objects from the real scene are rendered as such [4]. This is done by scaling each channel separately, thus also scaling differently the noise level of each channel. Given that the relationship between stimulus and human perception is logarithmic [5], cameras then apply a power law function to the intensity of each channel. After this step, known as gamma correction, the noise level is no longer monotonously increasing with the intensity.

JPEG compression The JPEG image standard is the most popular lossy compression scheme for photographic images [72]. The image goes through a colour space transformation and each channel is partitioned into non-overlapping 8×8-pixel blocks. The type-II discrete cosine transform (DCT) is applied to each of these blocks. The resulting coefficients are quantized according to a table and the coefficients are then compressed without additionnal loss. Due to the cancellation of high-frequency coefficients, the noise is reduced after compression.

1.3 The CFA Forgeries dataset

To evaluate one of our methods, presented in Chapter 6 (Positional Learning for Demosaicing Analysis), we wanted a dataset specifically made for demosaicing analysis. As detailed in Chapter 0.4 (Introduction), several datasets already exist to benchmark image forgery detection, most notably Coverage [73], CoMoFoD [39], Casia [36] and [38]. However, these datasets were created for generic copy-move detection. They do not allow for a demosaicing based detection. Indeed, the images of those datasets either do not present any trace of demosaicing, or were all demosaiced with the same algorithm. They are therefore useless for benchmarking CFA-based forgery detection algorithms.

The Dresden Image Database [32] provides 16,961 authentic images taken with 27 different cameras. Among them, 1,491 pictures taken with three different cameras, the Nikon D200, D70 and D70s, are provided unprocessed in a RAW format, which enabled us to perform demosaicing ourselves. Using these images, we created

Figure 1.3: Examples of forged images in the CFA Forgery dataset.

a new forgery detection database aimed specifically at the detection of forgeries by an analysis of CFA demosaicing inconsistencies.

To create the database, we cropped randomly each of the 1,491 images into smaller 648 × 648 pictures. We demosaiced them with one of eleven publicly available demosaicing algorithms: Bilinear interpolation, LMMSE [58], Hamilton-Adams [55], RI [60], MLRI [74], ARI [61], GBTF [59], contour stencils [75], adaptive inter-channel correlation [76], Gunturk [56] and self-similarity [57].

We then split the resulting set of images in three equal parts. One third of the images were left unmodified. In the second third, we took half of the images and used them to perform a splicing into the other half. Each pair of images had been previously demosaiced with the same algorithm. In the last third, we picked half the images again and used them to falsify the other half. However, we did not enforce pairs of images to be demosaiced with the same algorithm in this set. Note that the source images for the forgeries are not part of the resulting dataset; therefore, there is the same number of authentic and forged images. At least half the forged images were created with a source image demosaiced with the same algorithm as for the target.

To forge an image, we cropped the source image inside a random mask and pasted it onto the forged image. The masks were created as areas surrounded by random Bezier curves. They were enforced to contain at least one 64 × 64 square block, and to cover less than 10% of the image.

Examples of forged images in this dataset can be seen in Fig. 1.3.

From now on, we will refer to the dataset as the CFA Forgery dataset. We will use it further in Chapter 6 (Positional Learning for Demosaicing Analysis). It served as inspiration for, but is distinct from the Trace dataset and methodology, which we will detail in the rest of this chapter.

1.4 The Proposed Methodology

The CFA Forgery dataset is useful for testing demosaicing analysis methods, as it contains changes in demosaicing traces. However, it does not help testing the sensibility of generic methods to demosaicing traces, as the forgeries contain other traces. What we want is to create a dataset that enables one to test the sensitivity of methods to specific traces, without containing other traces.

We created a database of "forged" images which leaves the semantics of the images intact. The overall idea of our method is to take a raw image, process it with two different pipelines, and merge the two processed images as follows: the first image is

used for the authentic region and the second image for the "forged" area determined by a mask, as can be seen in Fig. 1.1. As a base we use the RAISE-1k dataset [31], which contains one thousand pristine raw images of varied categories, taken from three different cameras. We note that the variety of source cameras is not important to our database, as we erase the previous camera traces by downsampling the image, then resimulate the whole image processing pipeline ourselves, as explained below. Furthermore, our open source generation code can be applied on any other source of images, to automatically generate arbitrarily large quantities of "forged" images.

Methodology for the creation of the database A raw image already contains noise, furthermore its pixels are all sampled in the same CFA pattern. In order to reduce the noise and eliminate the CFA pattern, we start by downsampling each image by a factor 2. This enables us to choose the amount of noise to be added, and to mosaic the image in any of the four possible patterns. Once the image has been downsampled, we process the image with two different pipelines. The two images are then merged as explained above.

Forgery masks For each image we construct two different kinds of masks, which we shall call *endomasks* and *exomasks*. Since inconsistencies in the image processing pipeline are usually most visible at the border of the forgery, *endomasks* are obtained as regions of a segmentation of the image. To do this, we segment the original images with EncNet [77]. For each image, we take a pixel at random, and select the image region it belongs to. We accept the mask if its size is less than half the image's, otherwise we pick another pixel until we find a suitable mask. This ensures that each image has only one forgery, whose size is at most half the image's. Using such endogenous masks or *endomasks* corresponding to a region of the segmented image ensures almost invisible forgeries. Indeed their borders are natural image borders, as shown in Fig. 1.4.

The *exomasks* are instead unrelated to the image's content. To determine them, we start by pairing the images of the dataset according to their endomasks' sizes. Then, the endomask of each image is used as the exogenous mask, or *exomask*, of its paired image. Using a mask from another image ensures that the mask is not linked to the image's semantic. The chosen pairing enables comparisons separately on each image, as the size of the masks is similar. See Fig. 1.5 for examples of endo- and exomasks.

Multiple datasets One of our goals is to determine which inconsistencies each forensic tool is sensitive to. Changes in the image processing pipeline, done at different steps of the chain, lead to different inconsistencies (see Section 1.2). In consequence, we created five specific datasets, each of which features a specific change in the image processing pipeline. For each image, we started by randomly choosing the three parameters that are used for this image across all datasets:

- The mosaic pattern, chosen among the four possible offsets of the camera's Bayer pattern.

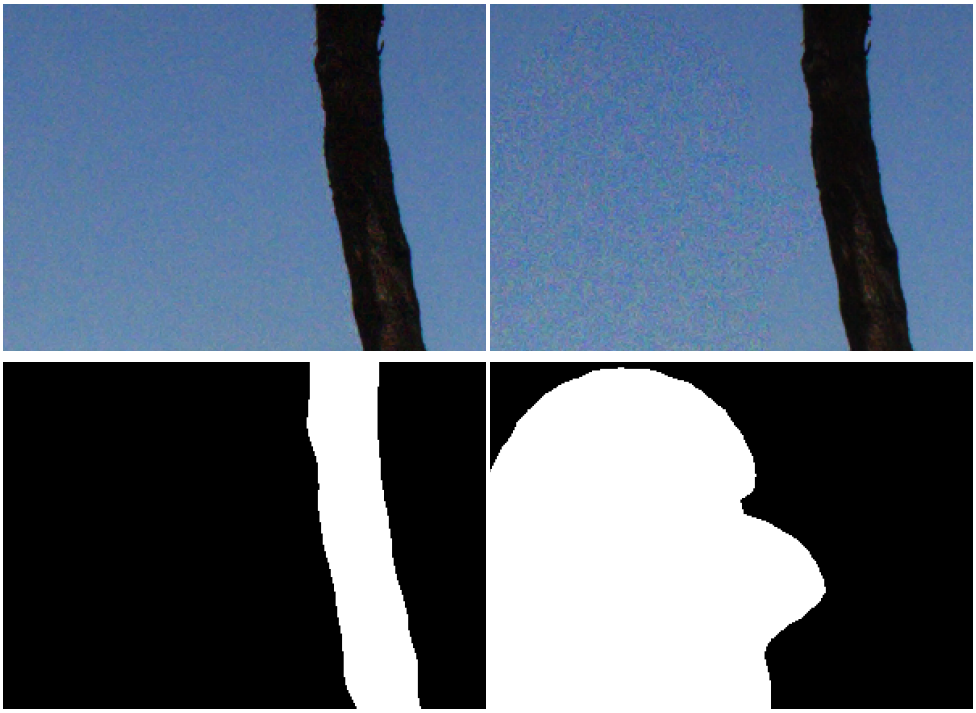


Figure 1.4: Details of the same image with forgeries made using the two masks. On the left, the endomask coincides with the image's structure, here a tree. The forgery is less conspicuous than on the right where the exomask is in the sky, where the borders do not coincide with the images' content.

- The demosaicing algorithm, chosen randomly among those available in the LibRaw library [78].
- The gamma-correction power.

The gamma correction is the same for both regions of the image, and the mosaic pattern is the same except for the CFA Grid, CFA Algorithm and Hybrid datasets. For each image, both the endo- and exomasks, constructed as explained above, are the same across all datasets.

Raw Noise Level dataset In this dataset we add random noise to each raw image before processing it. As pointed out in Section 1.2, noise variance in raw images follows a linear relation given by $\sigma^2 = A + Bu$, where A and B are constants and u is the noiseless image. We start by randomly selecting two different pairs of constants (A_0, B_0) and (A_1, B_1), in a range that ensures the resulting images look natural. Both images are then processed with the same pipeline. This dataset mimics the inconsistencies in noise models that could be found in spliced images.

CFA Grid dataset In this dataset we only change the mosaic pattern of the forged image inside the mask. Thus, the original image and the forged one would be identical if not for their mosaic grid origins. This kind of trace may appear (with probability $\frac{3}{4}$) when the forgery was an internal copy-move.

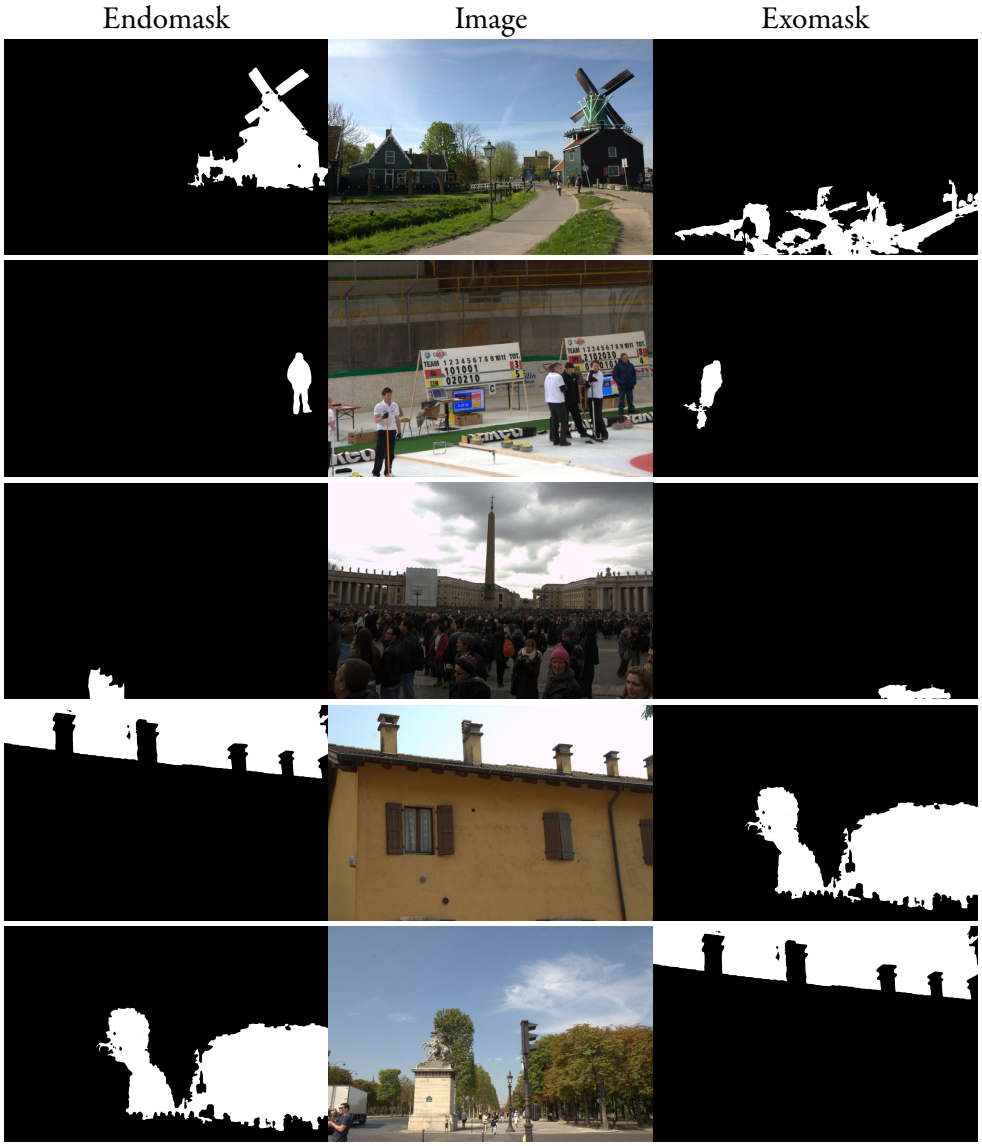


Figure 1.5: For each image, we use an *endomask* (left) taken from the image's segmentation, and an *exomask* (right) taken from another image and thus decorrelated from the image's contents. The last two images were paired during mask creation, thus the endomask of each becomes the exomask of the other.

CFA Algorithm dataset In this dataset, the two processing pipelines use different demosaicing algorithms. The demosaicing pattern is chosen independently for each pipeline. Thus there is a $\frac{1}{4}$ chance that they are aligned. A new mosaic pattern is also randomly chosen, thus having a $\frac{3}{4}$ chance of being different from the one of the main image. This dataset represents the change in the mosaic that would occur from splicing, as two different images most likely do not share the same demosaicing algorithms, and the alignment of their patterns after splicing is random.

JPEG Grid dataset In this dataset we only change the compression grid origin. Similarly to the CFA Grid dataset, if the forgery is an internal copy-move, the JPEG grid of the forged region is different from the grid in the authentic region, with probability $\frac{63}{64}$. The JPEG compression quality for both pipelines is then chosen randomly, keeping the values in a range that is typical of most compressed images and challenging enough for JPEG-based algorithms.

JPEG Quality dataset In this dataset, both the authentic and forged regions are processed with the same pipeline, except for the JPEG compression which is done in the two regions with different quality factors, again chosen uniformly between 75 and 100. Like with the CFA Algorithm dataset or the JPEG grid data, a new JPEG grid pattern is also randomly chosen, which has a $\frac{63}{64}$ chance of being different from the main region's grid. This dataset simulates the effect of the splicing of an image onto another, both images being compressed at different quality factors.

The hybrid dataset One could argue that although generic learning-based forensics tools may not be able to point out a single inconsistency in an image, they might be best suited to find multiple inconsistencies stacked together. Clearly, a splicing may introduce joint inconsistencies in noise level, JPEG encoding and demosaicing; while a direct copy-move can introduce alterations in the JPEG and CFA grids. To investigate such possibilities, in addition to the five specific datasets described above, we created a sixth, hybrid dataset. In this dataset, forgeries combine noise, demosaicing and/or JPEG compression traces. At least two of those traces are altered in each images.

1.5 Experiments

Evaluated methods

We used the constructed database to conduct an evaluation of image forensics tools. We tested both classic and SOTA forgery detection methods pertaining to different traces: noise-level-based detection methods Noisesniffer [19], Lyu [18], [79] and Mahdian [17], [79]; CFA-grid detection methods Shin, Jeon and Eom [15] as well as our methods from Chapters 3 (Intermediate Values Counting for CFA Pattern identification) and 6 (Positional Learning for Demosaicing Analysis) extended from Choi, Choi and Lee [14]; JPEG-based methods Zero [25], CAGI [21], [79], FDF-A [24], [79], I-CDA [23], [79], CDA [22], [79] and BAG [20], [79], as well

as generic methods Splicebuster [26], Noiseprint [27], ManTraNet [30] and Self-Consistency [28].

Evaluation Metrics

We evaluated the results of these methods using the Matthews correlation coefficient (MCC) [80]. This metric varies from -1 for a detection that is complementary to the ground truth, to 1 for a perfect detection. A score of 0 represents an uninformative result and is the expected performance of a random classifier. The MCC is more representative than the F1 and IoU scores [81], [82], partly as it is less dependant on the proportion of positives in the ground truth, which is especially important given the large variety of forgery mask sizes in the database.

The MCC was computed for each image, and then averaged over each dataset. As most surveyed methods do not provide a binary output but a continuous heatmap, we weighted the confusion matrix using the heatmap.

Results

The complete results are given in Table 1.1. Visualization of the detection by several methods on one image across all datasets can be seen in Figure 1.6. In the CFA and JPEG datasets, state-of-the-art methods that focus on those specific traces, such as those we propose in Chapters 6 (Positional Learning for Demosaicing Analysis) and 7 (Internal Learning to Improve Adaptability) for CFA and ZERO [25] for JPEG, perform much better than generic tools. This is partly expected, as those methods aim to detect exactly this specific trace. This observation is more nuanced in the Noise Level dataset where, depending on the type of mask considered, Noisesniffer [19] and Self-Consistency [28] achieve the best results. Indeed, exomasks cover a wider range of intensities enabling a better comparison between noise models, which is exploited by Noisesniffer. Also, half the forgeries present in this database are undetectable for this method since it is only able to detect forgeries having lower noise levels.

On the hybrid dataset, the scores of the specific methods are lower than on the specific datasets. For the JPEG-based methods, this is explained by the fact that one sixth of this dataset does not feature JPEG compression traces. For the CFA and Lyu and Mahdian noise-based methods, this is made worse by the fact that JPEG compression alters the previous noise and demosaicing artefacts, as shown in Fig. 1.2. In particular, CFA-based methods are notoriously weak on JPEG images, since JPEG compression removes the high frequencies, in which mosaic artefacts lie. This can be seen in Fig. 1.6, where our demosaicing detection method from Chapter 6 (Positional Learning for Demosaicing Analysis) cannot make any prediction on the hybrid image, where the main and forged region were compressed with quality factors of 93 and 75, respectively.

While multi-purpose forensic methods can, to some extent, detect noise-level inconsistencies, in the demosaicing algorithm and in the JPEG quality, they are blind to shifts in both the JPEG and CFA grids. This is not entirely surprising; with the exception of Splicebuster, the tested generic tools are based on mostly-

		Dataset					
		Noise Level	CFA Grid	CFA Algorithm	JPEG Grid	JPEG Quality	Hybrid
Noise-level-based	Noisesniffer [19]	0.128 (0.228) 0.091 (0.198)	-0.008 (0.070) -0.011 (0.073)	0.029 (0.153) 0.005 (0.111)	-0.007 (0.076) -0.009 (0.082)	0.052 (0.179) 0.020 (0.140)	0.098 (0.210) 0.061 (0.182)
	Lyu [18]	0.010 (0.090) 0.007 (0.137)	0.002 (0.093) 0.010 (0.157)	0.002 (0.094) 0.009 (0.159)	0.000 (0.089) 0.007 (0.148)	0.002 (0.091) 0.013 (0.156)	0.012 (0.097) 0.018 (0.150)
	Mahdian [17]	0.046 (0.146) 0.055 (0.171)	0.005 (0.082) 0.023 (0.159)	0.039 (0.128) 0.057 (0.183)	0.005 (0.086) 0.014 (0.146)	0.036 (0.132) 0.052 (0.180)	0.055 (0.158) 0.067 (0.191)
CFA-based	Chapter 6	0.007 (0.084) 0.021 (0.153)	0.682 (0.329) 0.665 (0.349)	0.501 (0.427) 0.491 (0.429)	0.023 (0.095) 0.018 (0.107)	0.029 (0.091) 0.020 (0.100)	0.133 (0.288) 0.128 (0.290)
	Chapter 3	0.026 (0.025) 0.030 (0.018)	0.603 (0.203) 0.575 (0.191)	0.420 (0.208) 0.385 (0.210)	0.001 (0.002) -0.001 (0.002)	-0.001 (0.003) 0.001 (0.001)	0.156 (0.114) 0.139 (0.116)
	Shin[15]	0.007 (0.101) 0.004 (0.123)	0.104 (0.166) 0.099 (0.171)	0.085 (0.172) 0.084 (0.179)	-0.002 (0.042) -0.005 (0.058)	-0.001 (0.043) -0.006 (0.059)	0.015 (0.109) 0.012 (0.114)
JPEG-based	Zero [25]	0.000 (0.000) 0.000 (0.000)	0.000 (0.000) 0.000 (0.000)	0.000 (0.000) 0.000 (0.000)	0.796 (0.349) 0.756 (0.387)	0.732 (0.413) 0.708 (0.421)	0.638 (0.451) 0.624 (0.453)
	CAGI [21]	0.004 (0.045) 0.003 (0.052)	0.000 (0.027) 0.000 (0.042)	0.002 (0.033) 0.001 (0.044)	0.038 (0.077) 0.023 (0.077)	0.044 (0.080) 0.028 (0.082)	0.031 (0.071) 0.021 (0.073)
	FDF-A [24]	0.031 (0.139) 0.014 (0.169)	-0.004 (0.087) -0.015 (0.139)	-0.003 (0.085) -0.017 (0.139)	0.226 (0.242) 0.216 (0.265)	0.228 (0.249) 0.216 (0.273)	0.203 (0.244) 0.187 (0.264)
	I-CDA [23]	0.000 (0.000) 0.000 (0.000)	0.000 (0.000) 0.000 (0.000)	0.000 (0.000) 0.000 (0.000)	0.416 (0.417) 0.423 (0.408)	0.422 (0.407) 0.414 (0.414)	0.381 (0.407) 0.385 (0.408)
	CDA [22]	-0.001 (0.034) -0.004 (0.068)	0.000 (0.055) -0.003 (0.098)	0.000 (0.052) -0.005 (0.097)	0.485 (0.339) 0.449 (0.351)	0.474 (0.344) 0.442 (0.350)	0.401 (0.360) 0.378 (0.354)
	BAG [20]	0.000 (0.015) 0.002 (0.029)	0.006 (0.078) 0.025 (0.164)	0.009 (0.079) 0.026 (0.164)	0.232 (0.461) 0.227 (0.459)	0.229 (0.458) 0.223 (0.455)	0.171 (0.430) 0.161 (0.430)
Multi-purpose	Noiseprint [27]	0.127 (0.200) 0.108 (0.232)	-0.001 (0.069) 0.002 (0.114)	0.066 (0.149) 0.060 (0.179)	0.013 (0.087) 0.016 (0.140)	0.178 (0.248) 0.138 (0.279)	0.153 (0.230) 0.128 (0.261)
	ManTraNet [30]	0.049 (0.091) 0.032 (0.099)	0.000 (0.040) -0.004 (0.065)	0.074 (0.169) 0.053 (0.165)	0.004 (0.023) 0.000 (0.043)	0.095 (0.164) 0.086 (0.171)	0.112 (0.169) 0.107 (0.176)
	Self- -Consistency [28]	0.082 (0.323) 0.154 (0.429)	0.028 (0.261) 0.077 (0.393)	0.036 (0.270) 0.082 (0.403)	0.011 (0.262) 0.060 (0.386)	0.078 (0.335) 0.151 (0.440)	0.138 (0.370) 0.246 (0.425)
	Splicebuster [26]	0.099 (0.188) 0.100 (0.217)	0.003 (0.085) 0.012 (0.157)	0.075 (0.157) 0.072 (0.202)	0.005 (0.083) 0.006 (0.135)	0.084 (0.175) 0.082 (0.220)	0.101 (0.192) 0.099 (0.215)

Table 1.1: Results of different state-of-the-art forensics tools on our six datasets, using the Matthews Correlation Coefficient (MCC), detailed in Sec. 1.5. The methods, on the left, are grouped by categories. As a baseline, a random classifier is expected to yield a score of 0. The mean of the MCC scores over each image of the dataset, as well as the standard deviation in parentheses, are shown for the exogenous mask and endogenous mask datasets. Grayed-out numbers represent results of methods on datasets that are irrelevant to said methods. The best two scores are underlined for each database.

convolutional neural networks, which are invariant to translation. Although Noiseprint [27] adapts its training scheme to be able to detect shifts in periodic patterns, it entirely fails to see the demosaicing grid, and does little better than random detecting JPEG grid inconsistencies.

Most methods perform similarly on the endomask and exomask datasets. Two

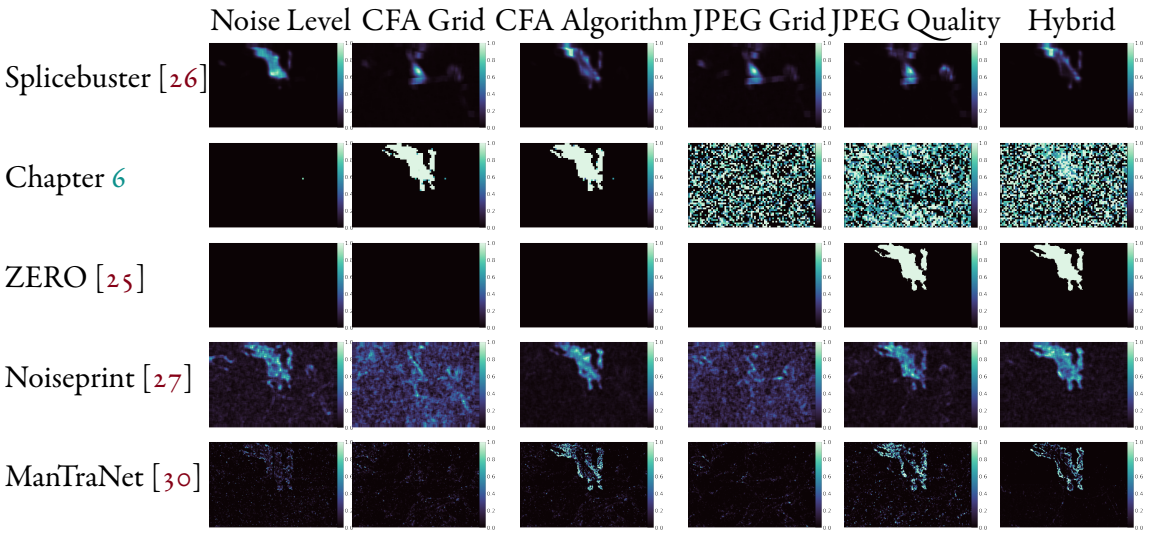


Figure 1.6: Visualization of the results of several methods for one image on all the datasets. Some methods, such as Noiseprint or the one proposed in Chapter 6 (Positional Learning for Demosaicing Analysis), correctly detect the forgeries in the relevant images, but tend to make noise-like false detections in the images for which they cannot see the forgery. Automatically selecting the relevant detections of an algorithm would make it easier to use without needing interpretation, which is why Chapter 7 (Internal Learning to Improve Adaptability) will do it when extending Chapter 6 (Positional Learning for Demosaicing Analysis)'s method. The image and mask can be seen in Fig. 1.1.

notable exceptions are Noisesniffer which underperforms on endomasks, and Self-Consistency, which works much better on endomasks. Both observations are easily explained: the noise model is better estimated by Noisesniffer on a flat region. The same explanation is valid for Noiseprint, which also loses performance with exomasks. In contrast, Self-consistency's content-awareness is lost when segmenting forgeries with exomasks. Regardless of the dataset considered, the scores obtained by all of the methods have a high standard deviation with respect to their mean value. This suggests that, given a dataset, the scores in each individual image are not concentrated around the mean but rather spread on a large range of values. Hence, even for methods having low scores, some good detections are likely to happen.

1.6 Discussion

The fact that most examined methods perform similarly on exo- and endomasks could lead us to conclude that we could use only one kind. However, comparing the results on both reveals the ability of some methods, such as SelfCconsistency, to perform content-aware localization.

The goal of this evaluation was not to rank different methods, but to offer a rigorous insight on the capabilities of each method. Knowing the kind of inconsistencies to which each forensic tool is sensitive helps understand and explain its detections in uncontrolled cases, and can help efforts to combine different methods. In

that sense, the proposed database is complementary to more traditional databases.

Methods that focus on detecting specific traces are often opposed to more generic methods. However, this study shows the complementarity and possible synergies between the two paradigms. For instance, results on the CFA Algorithm datasets showed that, even without explicitly training them, neural networks were sometimes able to detect changes in the demosaicing algorithm, a fact that is usually considered almost impossible, especially locally, except with the most basic demosaicing algorithms [6].

Our experiments also reveal a problematic issue with many of the tested methods. Even though they can yield decent scores, the standard deviations of theses scores over all images of the same dataset is often very high. Even though algorithms perform well on many forgeries, they also often yield false positives that require interpretation to be distinguished from true detections, such as Chapter 6 (Positional Learning for Demosaicing Analysis)'s method and Noiseprint in some datasets of the example image seen in Fig. 1.1. This is a critical point for many methods, as it makes them usable only to a trained eye.

1.7 Conclusion

Image forensics datasets are usually grouped according to forgery types (eg. splicing, inpainting, or copy-moves), and do not separate the semantic content from the actual traces left by the forgery. In this chapter, we proposed to remove the semantic value of forgeries so as to focus only on the traces. We designed a methodology to automatically create image "forgeries" that leave no semantic traces, by introducing controlled changes in the image processing pipeline. We built datasets by focusing on noise-level inconsistencies, mosaic and JPEG artefacts, and conducted an evaluation of some image forensics tools using this dataset.

Although we focused on three kinds of changes in the forgeries, the same methodology can be applied to more traces, including PRNU inconsistencies, multiple compression, or image manipulations such as resampling. In fact, we can address all forgeries where two different camera pipelines are involved. This includes copy-move, splicing and some methods of inpainting. Further work will incorporate other traces, such as those left by synthesis methods.

Our method can transform automatically large sets of images into forged images with fully controlled tampering cues and no bias that might cause overfitting. Besides evaluation of existing image forensics tools, this methodology could also be used to train forgery detection methods, although care would be needed so as not to overfit if using the same methodology for both training and evaluation.

In our specific case of demosaicing, the CFA Grid and Algorithm datasets will be used extensively in Chapters 2 (CFA Identification with Differential Operators), 3 (Intermediate Values Counting for CFA Pattern identification), 4 (Linear Estimation of the Demosaicing Algorithm), 6 (Positional Learning for Demosaicing Analysis) and 7 (Internal Learning to Improve Adaptability).

Chapter 2

CFA Identification with Differential Operators

Abstract

After analysing the problem of evaluating forensic methods, we begin our search for a demosaicing analysis method. In this first method, we try to highlight sampled pixels from interpolated ones with simple numerical cues from a differential operator. We compare the response of the four patterns pair by pair with a statistical test, to detect significantly impossible patterns. Images are then declared forged when no single pattern is possible everywhere. The proposed method is very robust to false detections. However, the linearity and channel-independence of the study makes detections scarce when the demosaicing algorithm is too advanced, or when multiple patterns are present. As a consequence, Chapter 3 (Intermediate Values Counting for CFA Pattern identification) will look for a more subtle cue. This study highlights the difficulty of a coherent analysis of demosaicing traces.

An interactive demo for this chapter is available at https://ipolcore.ipol.im/demo/clientApp/demo.html?id=77777000058.

2.1 Introduction

State-of-the-art CFA pattern identification methods already provide decent results for pattern classification, provided the window is large enough and JPEG compression is not too drastic. However, the results of such a classification are not sufficient when it comes to detecting forged images. Indeed, an incorrect CFA diagnosis on a single image block is enough to consider an image as falsified. Thus, a method with a 95% accuracy on 128×128 blocks would risk detecting a falsification nine times out of ten in a standard, authentic image of size 1024×768. In other words, controlling the false alarms rate over the many blocks of an image is necessary to detect forgeries.

In this chapter, we propose a new method to reliably detect the correct position of the Bayer matrix. We start by applying a high-pass filter tailored to highlight the difference between original and interpolated pixels. We then use a statistical test to compare the samples of the possible grid positions, in order to know which

positions are significantly impossible in the image. Comparing the results in the global image and across smaller windows enables us to detect and localise forged regions in images. The proposed methods is able to detect inconsistencies of images demosaiced with a simple algorithm, and does not yield any false positives in our experiments; however it is unable to make detections when the demosaicing is more advanced.

2.2 Method

The *a contrario* paradigm for reliable detections

One of the key goals of our method is not only to find which of the four possible patterns $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$, $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$, $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$ and $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ have been used, but to know how confident we are in our detection so as to control the number of false alarms.

Let $x_p, p \in \left\{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}, \begin{smallmatrix}G&R\\B&G\end{smallmatrix}, \begin{smallmatrix}G&B\\R&G\end{smallmatrix}, \begin{smallmatrix}B&G\\G&R\end{smallmatrix}\right\}$ be four samples, each containing n non-negative values. We consider that each CFA position p is represented by sample x_p – we will detail later how these samples are constructed. We assume that in the absence of demosaicing, ie. if there is no reason to favour one CFA pattern, then the values of all x_p are similar, but that if the image has been demosaiced and the used CFA pattern is $p^\star$, then the value of $x_{p^\star}$ are higher than those of the other x_p.

Assuming that an image has been demosaiced, the position of the Bayer matrix can be obtained by taking the mean of all x_p and selecting the highest one. However, it may be that the image has not been demosaiced, or that the traces of demosaicing can no longer be found, for example because of post-processing effects or because the image is too small. How then can we be certain that the values our algorithm returns are correct?

In order to get a reliable detection, we make use of the *a contrario* paradigm [83]–[85], detailed in Chapter 0.4 (Introduction). The approach is based on the non-accidentalness principle, according to which an observed geometric structure is perceptually meaningful only when its expectation is low under random background model. Detection thresholds can then control the expected number of false detections in this background, or *a contrario* model H_0. An observed structure is validated only when a test rejects the H_0 hypothesis. The detection threshold must take into account our multiple testing, as in Gordon et al. [86]. Assume that N_T tests are performed and that a variable u is observed at each one. We desire to set a threshold τ such that the sought structure is validated when $u \geq \tau$. Following the *a contrario* methodology, we define the Number of False Alarms (NFA) of a candidate by

$$\text{NFA} = N_T \mathbb{P}_{H_0}(U \geq u), \tag{2.1}$$

where $\mathbb{P}_{H_0}(U \geq u)$ is the p-value of observing under random hypothesis H_0 a value U as large or equal to u. The candidate is validated whenever NFA $< \varepsilon$, where ε is a predefined accepted mean number of false detections. This yields an implicit value for the threshold τ. Accepting all detections with a NFA score of $\varepsilon = 10^{-3}$, we should expect an average of one false positive for 1,000 images.

For CFA detection, we compare two samples x_1 and x_2, each having n non-negative values. We determine whether the first sample has significantly larger val-

ues than the second. For this, we compute the Mann-Whitney U statistic [87],

$$u = \sum_{i=1}^{n} \sum_{j=1}^{n} 1_{\{x_1^i > x_2^j\}} \tag{2.2}$$

where 1 is the indicator function and x_p^k is the k-th value of sample p. The value of u belongs to $[0, n^2]$. This value is zero when all samples of x_1 are smaller than the samples of x_2. Conversely, $u = n^2$ when all samples of x_1 are larger than the ones of x_2. We can now define a natural background model for our statistical test. Its null hypothesis H_0 is that all samples in both X_1 and X_2 were independently drawn from the same distribution, which must be correct if no demosaicing has been performed. Thus, $\mathbb{P}_{H_0}(X_1^i > X_2^j) = \mathbb{P}_{H_0}(X_1^i < X_2^j) = \frac{1}{2}$ for any i, j. This defines the Mann-Whitney U test [87] with the corresponding random variable U. For large samples, U is approximately normally distributed, which allows a simple computation of the p-value $\mathbb{P}_{H_0}(U \geq u)$.

To sum up, given two samples x_1 and x_2, the associated U statistic is computed with eq. 2.2. Then, the NFA value is given by eq. 2.1 and by the p-value of the Mann-Whitney test. Finally, the sample x_1 is declared significantly larger than x_2 if NFA $<$ ε.

Detecting the possible CFA patterns

The first thing to do is to find which pixels are original samples and which have been interpolated. An easy way to do this is to apply a differential operator like the discrete Laplacian, which highlights extremal values and thus pixels likely to have been interpolated. Following the example of [88] or [6], it is also possible to apply in each of the four possible grid positions a demosaicing algorithm – either fixed as in [88] or estimated as in [6] – and to compare the residuals.

One could simply compute a heat map with a differential operator – which can be implemented as a linear convolution followed by a point-wise absolute value operator. Yet this would not take into account that, depending on their positions on the Bayer matrix, the pixel subset from which interpolation is performed may vary. On the other hand, using a demosaicing algorithm in four positions must be done with caution. Preliminary experiments suggest that using a fixed demosaicing algorithm, such as the bilinear algorithm, does not yield good results when it is too different from the algorithm that was used to process the image, and an estimation of the demosaicing algorithm is prone to bias towards one of the four possible patterns.

As the possible patterns will be compared in pairs, we follow another approach that gives us both the simplicity and impartiality of differential operators and the ability to take different interpolation cases into accounts. Each test is a comparison between two patterns, pixels that are not original samples are thus interpolated using values from a known subset of pixels.

For example, if we compare the CFA patterns $\begin{smallmatrix} R & G \\ G & B \end{smallmatrix}$ and $\begin{smallmatrix} G & R \\ B & G \end{smallmatrix}$, we know the considered red values can only be interpolated from horizontal neighbours, blue values from vertical neighbours, and green values can be interpolated from all four adjacent neighbours, because none of the other pixels would be original in any of the

two considered patterns. Furthermore, we note that when an interpolation can be done in two directions simultaneously, many algorithms can decide to interpolate in only one direction, mainly to prevent interpolating across a strong edge. In order to mimic this behaviour, we do the interpolation simultaneously and then separately in the two directions, then for each pixel we take the result with the lowest absolute residual.

This leads us to define not one, but four different heat maps that will be used in the different comparisons. Let I be an image of shape $(X, Y, 3)$[1], we define the horizontal and (resp. vertical) heat map $H_h[x, y, c]$ (resp. $H_v[x, y, c]$) as the absolute difference between $I[x, y, c]$ and its two horizontal (resp. vertical) neighbours. They correspond to the interpolation of red or blue pixels on green-sampled locations. The straight cross heat map $H_c[x, y, c]$ is equal to either the absolute difference between $I[x, y, c]$ and the mean of its two horizontal neighbours or its two vertical neighbours or both, whichever of the three possibilities yields the lowest result. It corresponds to the interpolation of green pixels. Finally, the diagonal cross heat map $H_d[x, y, c]$ is equal to the absolute difference between $I[x, y, c]$ and the mean of its two or four diagonal neighbours following either or both of the diagonals, whichever of the three possibilities yields the lowest result. It corresponds to the interpolation of red pixels on blue-sampled locations, and vice versa.

With these four heatmaps, we now describe how to reliably compare different CFA positions. We start by assuming that each of the four CFA patterns $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$, $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$, $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$ and $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ are possible, and we look for CFA patterns that are significantly impossible. A grid position is considered significantly impossible if it is inferior to another grid position and the NFA score of the comparison between those too positions is below the set threshold. For each pair of CFA patterns $i, j \in \left\{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}, \begin{smallmatrix}G&R\\B&G\end{smallmatrix}, \begin{smallmatrix}G&B\\R&G\end{smallmatrix}, \begin{smallmatrix}B&G\\G&R\end{smallmatrix}\right\}$, $i \neq j$, we try to know which of the two grids is stronger than the other, and how significantly. We consider two cases, depending on the positions to compare.

The first case is when both positions share the same diagonal $\begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}$ or $\begin{smallmatrix}G&\cdot\\\cdot&G\end{smallmatrix}$, ie. if we are either comparing $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ with $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ or $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$ with $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$. Then we can only perform pattern comparisons using the red and blue channels. Since the interpolation between these two grids is done in diagonal, we look at the red and blue pixels of H_d. We average over each 2×2 block the red and blue pixels corresponding to CFA pattern i, and we do the same for j. We then compare the two samples and multiply its score by 6 since all pairs are compared. If the comparison is coherent across channels and significant, i.e. if its score is below the set NFA threshold, then the identified weaker of the two grids is marked as impossible.

The second case is when both positions do not share the same diagonal, ie. if $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ or $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ is compared with $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$ or $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$. We then can perform pairwise comparisons in all three channels. As most demosaicing algorithms start by interpolating the green channel, and use its values to demosaic the other two channels, we consider the green channel separately: For the green channel, we look at the pixels of H_c and compare those that correspond to an original pixel in i to those that correspond to an original pixel in j. For the red and blue channel, we also compare those that correspond to each position, but we look in different heatmaps depending on the compared patterns: If we are comparing $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ with $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$ or $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ with $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$ we look at

[1] where the last dimension represents the colour channels.

H_h for the red channel and H_v for the blue channel. In the other two cases, we look at H_v for the red channel and H_h for the blue channel. Each comparison score is multiplied by 6 as above and by two since green values are treated differently. Then, if the comparison of the green channel is significant, and if at least one of the two red and blue channels is significant and coherent with the green channel, the weaker of the two patterns is marked as impossible.

In both cases, the comparison requires channel consistency (CC), in that two channels need to be coherent for the detection to be approved. CC enforcement is double-edge; on the one hand it is a strong way of avoiding detections at a high risk of being erroneous, especially considering those regions may still be above the significance threshold. On the other hand, it forces the method to drop detections that could be correct. We thus propose to remove the CC enforcement. When the diagonal of the compared patterns is the same, if the red and blue channels are contradictory with one another, we accept the results of the most significant one. When the compared pattern do not share their diagonal, we accept the results of the green pattern, which is usually easier to analyse and thus presents more accurate results. The two variants, with and without CC enforcement, will be compared in the experiments.

Finding forgeries

The main use of demosaicing artefacts is to find forgeries in an image. This is done by partitioning the heat map H in small windows. As with the global image, we decide in each window which grid positions are possible and which are significantly not, with the difference that we multiply the comparison scores by the number of windows before thresholding on the NFA score, as we want to control the number of false alarms in each image, and not just in each window.

If a window is inconsistent with the global image, ie. if there is no grid position which is possible both in the global image and the window, then a forgery has been identified and localised. An example of this result can be found in Fig. 2.1a. In a genuine image, there should also be at least one grid position which remains possible throughout all windows. If this is not the case, then a forgery has been identified. The windows having an impossible grid configuration may contain a forgery. An example of this result can be found in Fig. 2.1c.

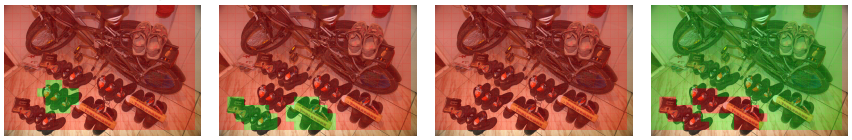
Since grid detection is usually easier and thus more precise in the global image than in smaller windows, inconsistent grids is the primary way of detecting forgeries. However, if an image is fabricated from several images of similar sizes, or if an object recovering an important part of the image itself has been copy-moved, the CFA grid position of the forged part of the image may have been accepted as possible in the global image. In such cases, impossible grid configurations could detect forgeries not found by the former.

2.3 Experiments

As a sanity check of our method, we confirm the absence of significant detections in images which have not been demosaiced. We used the 18 images from the noise-

(a) Found forged regions are in red, and regions where the exact CFA grid was identified are in green

(b) Original image



(c) Regions where the respectively $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$, $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$, $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$ and $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ Bayer matrix positions are deemed possible (in green) or significantly impossible (in red).

Figure 2.1: Example of forgery detection. Images from [38].

free images dateset [33]. Those images were downsampled by a factor of 8 and thus do not have any traces of demosaicing or JPEG encoding. We also used images of uniform and normal noise of different sizes: for each kind of noise, we made 10 images of size 128×128, 10 of size 256×256, 10 of size 512×512 and 10 of size 1024×1024, for a total of 80 noise images. The datasets were tested uncompressed as well as with JPEG compression of qualities 100, 99, 98, 95, 90, 80, 70, 60, 50, 30 and 10. No traces of demosaicing were found by our algorithm.

We then used the method with 32×32 windows on the Korus dataset [42], [43]. Results are presented in Table 2.1, and visual results in Figure 2.2.

When channel consistency was enforced, no false detections happened. Without channel consistency, however, false positives could not be prevented. Surprisingly, these false positives happened mostly on the Sony α57 camera, on which the method overall performed best despite the false positives.

Channel consistency is thus necessary to fully control false positives. Without it, the method can receive contradictory information from different channels. As this information is still significant, *a contrario* thresholding is unable to filter it out. On the other hand, channel consistency enforcement prevents many correct detections from being made. While the number of true positives did not change between the two variants on the Sony α57 images, no detections could be made with channel consistency on the Nikon D7000 and D90 images, whereas respectively 14 and 15 of the forgeries were detected without channel consistency. This shows that the proposed method is not able to fully understand demosaicing artefacts.

No detections were made on the Canon 60D images. This is consistent with SOTA methods, as well as with those that will be presented later in this thesis. Indeed, it seems that images from this camera on the Korus dataset do not feature any demosaicing traces, possibly due to downsampling or aggressive post-processing in the pipeline.

CC \ Camera	Canon 60D	Nikon D7000	Nikon D90	Sony α57	All
Yes	0.00	0.00	0.000	0.136	0.034
No	0.00	0.094	0.073	0.128	0.074

(a) Matthews Correlation Coefficient (MCC)

CC \ Camera	Canon 60D	Nikon D7000	Nikon D90	Sony α57	All
Yes	0/55	0/55	0/55	22/55	0/55
No	0/55	14/55	15/55	22/55	51/220

(b) Number of images in which at least one window was correctly detected as forged (true positive).

CC \ Camera	Canon 60D	Nikon D7000	Nikon D90	Sony α57	All
Yes	0/55	0/55	0/55	0/55	0/220
No	0/55	0/55	2/55	11/55	13/220

(c) Number of images in which at least one window was incorrectly detected as forged (false positive).

Table 2.1: Results of the method on the Korus dataset, with window size 32×32, depending on whether channel consistency (CC) is enforced. If channel consistency is enforced, no decision is made if the different channels do not accept the same solution. It prevents all false detections in the tested images, but causes many true detections to be missed.

Finally, we note that the number of correct detections of the method remains relatively low, even without channel consistency. While this is due in part to the method being unable to fully apprehend demosaicing artefacts, it is also due to the fact that many forgeries on the Korus dataset were created by inpainting, more precisely by cloning multiple small patches into a target area. This results in a multitude of different grids within a window. As we are looking for one specific, significant grid, the detection is more difficult when a window features multiple patterns. On the other hand, methods presented in Chapters 4 (Linear Estimation of the Demosaicing Algorithm) and 7 (Internal Learning to Improve Adaptability) look more generically for regions that are inconsistent with the rest of the image in terms of demosaicing patterns. These methods will thus be naturally able to detect more forgeries of this kind.

2.4 Conclusion

In this chapter, we constructed a method highlighting the difference between original and interpolated pixels in demosaiced images. This method does not require an estimation of the CFA interpolation algorithm, but still uses knowledge on the

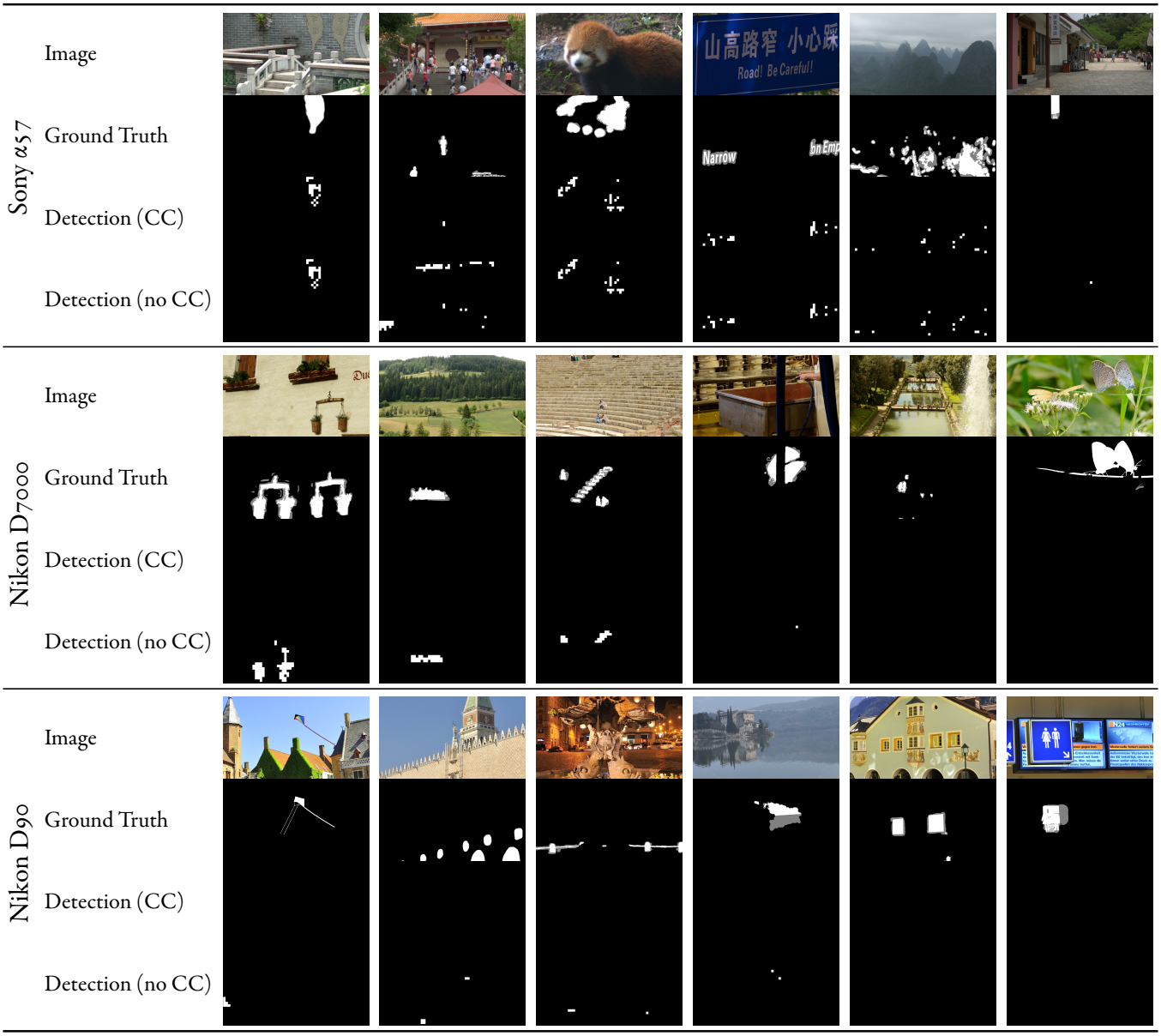


Figure 2.2: Visual results on images of the Korus dataset depending on whether channel consistency (CC) is applied or not. No detections were made on images from the Canon 60D camera, which is thus not shown here. Overall, not enforcing channel consistency enables many more detections to be made, at the cost of a few false positives such as the leftmost image of the Nikon D90 camera, or the second and sixth images of the Sony α57 camera.

specifics of demosaicing. We then explained how this method could be used to find not only traces of demosaicing, but also to get information on the position of the Bayer matrix. Inconsistencies about this information is a very strong clue of tampering in images, especially since the proposed approach yields a strong control of the false positives rate.

However, this method does not take into account the extensive inter-channel correlation that takes place during demosaicing. Without taking this entanglement

into account, it is either necessary to accept the results of the strongest channels – which leads to better classification scores but prevents a full control of the false alarms rate – or to only accept results that are coherent across channels – effectively enabling a control of the false alarms rate, but rendering the method unable to make decisions when the demosaicing algorithm is not simple enough.

Conjointly using the three colour channels to detect the CFA pattern is difficult, mostly because the existing demosaicking algorithms behave differently. Still, this approach will surely be necessary at some point – in addition to freeing ourselves from the assumptions of linearity – to fully master the analysis of demosaicing traces.

The numerical cue we used in this chapter was very simple. The next chapter will study another, better cue, that retains distinctive properties from interpolated samples.

Chapter 3

Intermediate Values Counting for CFA Pattern identification

Abstract

In the previous chapter, we tried to use a simple numerical cue to highlight interpolated pixels from sampled ones. We keep this idea, but try to follow a more subtle approach. Instead of directly highlighting interpolated pixels, we instead use one of their properties: as noticed by Choi et al., interpolated pixels are more prone to be intermediate values to detect in which pattern an image has been sampled. We analyse, implement and extend their method to detect the CFA pattern. We then use this information to find regions that are inconsistent with the global image. We attribute a confidence score to each detection, which can then be thresholded to provide a binary map of detected forgeries. Although this method does not yield coherent results on a few demosaicing algorithms, it is overall good at detecting the mosaic, at least on uncompressed images.

An online demo for this chapter is available at https://ipolcore.ipol.im/demo/clientApp/demo.html?id=355. As part of the Envisu project, the method presented here has also been integrated in the forensics browser plugin InVID & WeVerify[1].

3.1 Introduction

Demosaicing is basically an interpolation operation. As a consequence, interpolated pixels are more often intermediate values among their immediate neighbours, as seen in Figure 3.1. For instance, with the simple bilinear demosaicing, missing colours are directly averaged from the direct neighbours that were originally sampled in that colour, and are thus always intermediate values.

Of course, this simple behaviour is no longer true with more complex algorithms, which interpolate pixels using more samples among all three channels. Nevertheless, with most algorithms, an interpolated pixel is still more likely to be an intermediate value than a sampled one.

[1] Beta version for Chrome at https://chrome.google.com/webstore/detail/fake-news-debunker-by-inv/mhccpoafgdgbhnjfhkcmgknndkeenfhe.

18	56	94	85	76	96	116	104
49	56	63	52	41	64	88	87
80	56	32	19	6	33	60	70
59	62	66	49	32	59	87	88
38	69	100	79	58	86	114	106
40	51	63	74	85	75	66	63
42	34	26	69	112	65	18	21
38	47	57	75	94	72	50	35

(a) Red channel

139	240	154	16	94	56	72	20
92	131	168	76	72	94	24	43
85	24	100	48	102	224	130	72
60	107	160	68	64	122	200	153
92	184	125	0	50	0	133	108
52	155	156	76	136	117	224	127
146	228	111	12	110	108	107	44
56	114	48	90	184	141	52	90

(b) Green channel

Figure 3.1: Red and green channels of a toy image demosaiced with bilinear interpolation in the $\begin{smallmatrix}R & G\\ G & B\end{smallmatrix}$ pattern. Red values correspond to positions where the value was interpolated. Highlighted cells correspond to pixels that take an intermediate value, i.e. that are not a local extremum among their direct neighbours. While sampled pixels can have intermediate values, many more can be found among interpolated pixels in both the red and green channels. The blue channel, not shown here, behaves similarly to the red one.

Based on this, Choi et al. [14] proposed to identify the CFA pattern by counting intermediate values in all possible patterns. In this chapter, we describe, analyse and expand this method. The original article explains how to detect in which CFA pattern an image, or part of it, has been sampled. Starting from there, we detect which regions of an image are inconsistent with the main image, and attribute a confidence score to this detection. We also propose another way of computing intermediate values, which yields slightly better results.

This method does not require any linearity assumption, and is therefore better suited to analyse most demosaicing algorithms. However, it still does not take into account the inter-channel transfers that take place during demosaicing. This leads the method to yield contradictory results on algorithms who make extensive use of such transfers. Still, even though the correct pattern cannot be identified in these cases, the results are still relatively consistent across the image, enabling us to detect forgeries to some extent even on these more difficult images.

3.2 Method

During demosaicing, missing colours on each pixel are interpolated from its neighbours. As a consequence, pixels that are interpolated in a given channel are more likely to be an intermediate value, in other words, to be neither lower than all its direct neighbours nor higher than all of them. This is especially true with the simplest demosaicing algorithm, the bilinear demosaicing which interpolates the three channels separately.

The detection method analysed here counts the intermediate values corresponding to each of the four patterns. On the correct pattern, as most pixels are sampled, there should be fewer intermediate values than in the other patterns.

Intermediate values detection

Let I of shape (X, Y) be one colour channel of an image. The pixel at location (x, y) is considered an intermediate value if

$$\min(I_{x-1,y}, I_{x+1,y}, I_{x,y-1}, I_{x,y+1}) \leq I_{x,y} \leq \max(I_{x-1,y}, I_{x+1,y}, I_{x,y-1}, I_{x,y+1}).$$

We define $\mathcal{M}(I)$ as the mask of intermediate values of I. Its value is 1 if (x, y) is an intermediate value of I, and 0 otherwise.

If (x, y) is at the border of the image, at least one of $x \pm 1$ and $y \pm 1$ is out of the image boundaries. To avoid border effects, we would thus have to mask out a 1-pixel border around the image. However, doing this would cause an imbalance in the number of pixels corresponding to different patterns, in other words their would be, in border windows, more pixels corresponding to one pattern than to another. To solve the imbalance, we mask out a 2-pixels-wide border instead. More formally, the mask of intermediate values is therefore defined by

$$\mathcal{M}(I)_{x,y} \triangleq \begin{cases} 0 & \text{if } x \in \{0, 1, X-2, X-1\} \text{ or } y \in \{0, 1, Y-2, Y-1\} \\ 1 & \text{otherwise, if } \min(I_{x\pm1,y}, I_{x,y\pm1}) \leq I_{x,y} \leq \max(I_{x\pm1,y}, I_{x,y\pm1}) \\ 0 & \text{otherwise.} \end{cases}$$

The computation of this mask is described in Algorithm 1.

To limit demosaicing artefacts, many demosaicing algorithms tend to avoid interpolating against strong gradients, such as against an edge, and thus often only interpolate in one direction (in which the gradient is smaller). To take this into account, we propose to replace the original isotropic intermediate values mask with bidirectional filters, that separately consider horizontally and vertically intermediate values. We define the mask of horizontal intermediate values as

$$\mathcal{M}(I)^h_{x,y} \triangleq \begin{cases} 0 & \text{if } x \in \{0, 1, X-2, X-1\} \text{ or } y \in \{0, 1, Y-2, Y-1\} \\ 1 & \text{otherwise, if } \min(I_{x-1,y}, I_{x+1,y}) \leq I_{x,y} \leq \max(I_{x-1,y}, I_{x+1,y}) \\ 0 & \text{otherwise.} \end{cases}.$$

Vertical values are computed in a similar way as

$$\mathcal{M}(I)^v_{x,y} \triangleq \begin{cases} 0 & \text{if } x \in \{0, 1, X-2, X-1\} \text{ or } y \in \{0, 1, Y-2, Y-1\} \\ 1 & \text{otherwise, if } \min(I_{x,y-1}, I_{x,y+1}) \leq I_{x,y} \leq \max(I_{x,y-1}, I_{x,y+1}) \\ 0 & \text{otherwise.} \end{cases}.$$

With this definition, the bidirectional mask of intermediate values is then defined as the mean of the horizontal and vertical masks by

$$\mathcal{M}(I)_{x,y} \triangleq \frac{1}{2}\left(M(I)^h_{x,y} + M(I)^v_{x,y}\right).$$

The mask is therefore null at the border and where a pixel is not an intermediate value, equal to $\frac{1}{2}$ where the pixel is either horizontally or vertically an intermediate value, and equal to 1 when it is an intermediate value both horizontally and vertically. The computation of the bidirectional mask is detailed in Algorithm 2.

The original isotropic mask and the bidirectional one will be compared in Section 3.3. For the rest of this section, we consider R, G and B the masks of intermediate values obtained on the respectively red, green and blue channels of the image. Which of the two methods was used to compute those masks is irrelevant to the rest of the algorithm.

Algorithm 1: Mark intermediate values (original isotropic version)

```
1  function is_intermediate(arr)
       Input arr: Array of size (X, Y), one channel of an image
       Output mask: Array of size (X − 4, Y − 4), intermediate values mask
2      mask ≔ 0_(X−4,Y−4)
3      for x from 2 to X − 2 and y from 2 to Y − 2 do
4          mi ≔ min(arr_{x+1,y}, arr_{x,y−1}, arr_{x−1,y}, arr_{x,y+1})
5          ma ≔ max(arr_{x+1,y}, arr_{x,y−1}, arr_{x−1,y}, arr_{x,y+1})
6          if mi ≤ arr_{x,y} ≤ ma then
7              mask_{x−2,y−2} ≔ 1
8      return mask
```

Algorithm 2: Mark intermediate values (bidirectional variant)

```
1  function is_intermediate(arr)
       Input arr: Array of size (X, Y), one channel of an image
       Output mask: Array of size (X − 4, Y − 4), intermediate values mask
2      mask ≔ 0_(X−4,Y−4)
3      for x from 2 to X − 2 and y from 2 to Y − 2 do
4          m_h ≔ min(arr_{x−1,y}, arr_{x+1,y})
5          M_h ≔ max(arr_{x−1,y}, arr_{x+1,y})
6          m_v ≔ min(arr_{x,y−1}, arr_{x,y+1})
7          M_v ≔ max(arr_{x,y−1}, arr_{x,y+1})
8          if m_h ≤ arr_{x,y} ≤ M_h then
9              mask_{x−2,y−2} += 1/2
10         if m_v ≤ arr_{x,y} ≤ M_v then
11             mask_{x−2,y−2} += 1/2
12     return mask
```

Division into windows

The strategy to find forgeries using inconsistencies in the CFA patterns is to first find in which pattern the full image has been demosaiced, then to find the pattern used in different windows of the image. If the pattern detected in a window is different from the one detected for the full image, then this window is inconsistent with the rest of the image and can be considered as forged.

To improve the precision of detection, we do not simply use adjacent windows, but rather sliding windows with overlap. The window size W and stride are set as parameters of the algorithm. The stride determines the number of pixels between the left (or top) border of two consecutive windows, so that a stride equal to the window size leads to adjacent windows without overlapping, a stride equal to half the window size leads to a new window starting at the middle of the previous one, etc.

Using a lower stride will not drastically improve the detection, but may help

delineate a detected forgery more precisely, at the cost of a slower algorithm.

Finding the pattern

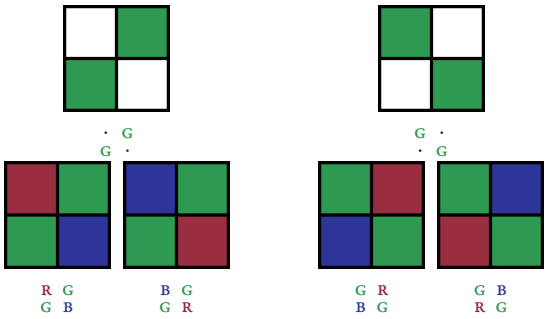


Figure 3.2: The four possible sampling patterns can be grouped by the diagonal on which the green channel was sampled: $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ and $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ share the $\begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}$ diagonal, whereas $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$ and $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$ share the $\begin{smallmatrix}G&\cdot\\\cdot&G\end{smallmatrix}$ one.

The four Bayer patterns can be divided into two subgroups by their diagonal: $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ and $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ share the $\begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}$ diagonal, whereas $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$ and $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$ share the $\begin{smallmatrix}G&\cdot\\\cdot&G\end{smallmatrix}$ diagonal. Because the Bayer CFA samples twice as many pixels in green than in red or blue, it is easier to find information on the pattern in the green channel. This is amplified by the fact that many demosaicing algorithms first interpolate the green channel by itself, but interpolate the red and blue channels using information from the green channel.

As a consequence, the presented method first tries to detect the diagonal pattern using the green channel ($\begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}$ or $\begin{smallmatrix}G&\cdot\\\cdot&G\end{smallmatrix}$), then uses the red and blue channels to compare the two potential patterns sharing that diagonal. We denote by R, G and B the masks of intermediate values on the respectively red, green and blue channels. (They will not be confused with the R, G, B channels that we no longer use in the sequel of this chapter). These masks can represent either the full image or a window of it. To maintain the balance between patterns, the masks must be of even size. For this reason, the window size must be even, and the last row/column of the full image is removed if necessary to ensure the evenness of the shape. Here we denote the shape of these masks $(2X, 2Y)$ for easier notations of the different positions on the CFA. We start by looking at the green channel for the diagonal grids. The intermediate value count corresponding to the $\begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}$ pattern is

$$C_{\begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}} \triangleq \sum_{x=0}^{X}\sum_{y=0}^{Y}\left(G_{2x+1,2y}+G_{2x,2y+1}\right)$$

while the count corresponding to the $\begin{smallmatrix}G&\cdot\\\cdot&G\end{smallmatrix}$ pattern is

$$C_{\begin{smallmatrix}G&\cdot\\\cdot&G\end{smallmatrix}} \triangleq \sum_{x=0}^{X}\sum_{y=0}^{Y}\left(G_{2x,2y}+G_{2x+1,2y+1}\right).$$

The count difference of the diagonal is then defined as

$$\Delta_{\text{diag}} \triangleq \frac{1}{2X\cdot Y}\left(C_{\begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}} - C_{\begin{smallmatrix}G&\cdot\\\cdot&G\end{smallmatrix}}\right)$$

This difference is positive if the detected diagonal is $\begin{smallmatrix}\mathrm{G} & \cdot \\ \cdot & \mathrm{G}\end{smallmatrix}$, and negative if it is $\begin{smallmatrix}\cdot & \mathrm{G} \\ \mathrm{G} & \cdot\end{smallmatrix}$:

$$D \triangleq \begin{cases} \begin{smallmatrix}\mathrm{G} & \cdot \\ \cdot & \mathrm{G}\end{smallmatrix} & \Delta_{\text{diag}} > 0 \\ \begin{smallmatrix}\cdot & \mathrm{G} \\ \mathrm{G} & \cdot\end{smallmatrix} & \Delta_{\text{diag}} < 0 \\ -1 & \Delta_{\text{diag}} = 0 \end{cases}.$$

The normalization by $\frac{1}{2XY}$ means that the resulting difference belongs to $[-1, 1]$, and is equal to ± 1 if all pixels in one of the patterns are intermediate values, whereas the other pattern has no intermediate values (XY is the number of 2×2 blocks in a mask of shape $(2X, 2Y)$, and we sum two pixels in this block for each pattern). Note that the ± 1 limit is only theoretical: even with bilinear demosaicing, where all interpolated pixels are intermediate values, sampled pixels can be intermediate too, for instance where they belong to a slope. As a consequence, the difference will not reach those values in natural cases.

Once we know the main diagonal, we can compare the two patterns sharing that diagonal. The green channel does not provide any information on this, so we use the red and blue channels. The count of intermediate values corresponding to each pattern is

$$\begin{aligned} C_{\begin{smallmatrix}\mathrm{R} & \mathrm{G} \\ \mathrm{G} & \mathrm{B}\end{smallmatrix}} &\triangleq \sum_{x=0}^{X} \sum_{y=0}^{Y} \left(R_{2x,2y} + B_{2x+1,2y+1}\right) \\ C_{\begin{smallmatrix}\mathrm{B} & \mathrm{G} \\ \mathrm{G} & \mathrm{R}\end{smallmatrix}} &\triangleq \sum_{x=0}^{X} \sum_{y=0}^{Y} \left(R_{2x+1,2y+1} + B_{2x,2y}\right) \\ C_{\begin{smallmatrix}\mathrm{G} & \mathrm{R} \\ \mathrm{B} & \mathrm{G}\end{smallmatrix}} &\triangleq \sum_{x=0}^{X} \sum_{y=0}^{Y} \left(R_{2x+1,2y} + B_{2x,2y+1}\right) \\ C_{\begin{smallmatrix}\mathrm{G} & \mathrm{B} \\ \mathrm{R} & \mathrm{G}\end{smallmatrix}} &\triangleq \sum_{x=0}^{X} \sum_{y=0}^{Y} \left(R_{2x,2y+1} + B_{2x+1,2y}\right) \end{aligned}.$$

The count differences of the two pattern pairs are then defined as

$$\begin{aligned} \Delta_{\begin{smallmatrix}\mathrm{R} & \mathrm{G} \\ \mathrm{G} & \mathrm{B}\end{smallmatrix}\_\begin{smallmatrix}\mathrm{B} & \mathrm{G} \\ \mathrm{G} & \mathrm{R}\end{smallmatrix}} &\triangleq \frac{1}{2XY}\left(C_{\begin{smallmatrix}\mathrm{R} & \mathrm{G} \\ \mathrm{G} & \mathrm{B}\end{smallmatrix}} - C_{\begin{smallmatrix}\mathrm{B} & \mathrm{G} \\ \mathrm{G} & \mathrm{R}\end{smallmatrix}}\right) \\ \Delta_{\begin{smallmatrix}\mathrm{G} & \mathrm{R} \\ \mathrm{B} & \mathrm{G}\end{smallmatrix}\_\begin{smallmatrix}\mathrm{G} & \mathrm{B} \\ \mathrm{R} & \mathrm{G}\end{smallmatrix}} &\triangleq \frac{1}{2XY}\left(C_{\begin{smallmatrix}\mathrm{G} & \mathrm{R} \\ \mathrm{B} & \mathrm{G}\end{smallmatrix}} - C_{\begin{smallmatrix}\mathrm{G} & \mathrm{B} \\ \mathrm{R} & \mathrm{G}\end{smallmatrix}}\right) \end{aligned}$$

and are then combined into the main grid difference

$$\Delta_{\text{main}} \triangleq \begin{cases} \Delta_{\begin{smallmatrix}\mathrm{R} & \mathrm{G} \\ \mathrm{G} & \mathrm{B}\end{smallmatrix}\_\begin{smallmatrix}\mathrm{B} & \mathrm{G} \\ \mathrm{G} & \mathrm{R}\end{smallmatrix}} & D = \begin{smallmatrix}\cdot & \mathrm{G} \\ \mathrm{G} & \cdot\end{smallmatrix} \\ \Delta_{\begin{smallmatrix}\mathrm{G} & \mathrm{R} \\ \mathrm{B} & \mathrm{G}\end{smallmatrix}\_\begin{smallmatrix}\mathrm{G} & \mathrm{B} \\ \mathrm{R} & \mathrm{G}\end{smallmatrix}} & D = \begin{smallmatrix}\mathrm{G} & \cdot \\ \cdot & \mathrm{G}\end{smallmatrix} \end{cases}.$$

Finally, the main detected grid can be obtained as

$$M \triangleq \begin{cases} \begin{smallmatrix}\mathrm{R} & \mathrm{G} \\ \mathrm{G} & \mathrm{B}\end{smallmatrix} & D = \begin{smallmatrix}\cdot & \mathrm{G} \\ \mathrm{G} & \cdot\end{smallmatrix} \text{ and } \Delta_{\text{main}} < 0 \\ \begin{smallmatrix}\mathrm{B} & \mathrm{G} \\ \mathrm{G} & \mathrm{R}\end{smallmatrix} & D = \begin{smallmatrix}\cdot & \mathrm{G} \\ \mathrm{G} & \cdot\end{smallmatrix} \text{ and } \Delta_{\text{main}} > 0 \\ \begin{smallmatrix}\mathrm{G} & \mathrm{R} \\ \mathrm{B} & \mathrm{G}\end{smallmatrix} & D = \begin{smallmatrix}\mathrm{G} & \cdot \\ \cdot & \mathrm{G}\end{smallmatrix} \text{ and } \Delta_{\text{main}} < 0 \\ \begin{smallmatrix}\mathrm{G} & \mathrm{B} \\ \mathrm{R} & \mathrm{G}\end{smallmatrix} & D = \begin{smallmatrix}\mathrm{G} & \cdot \\ \cdot & \mathrm{G}\end{smallmatrix} \text{ and } \Delta_{\text{main}} > 0 \\ -1 & \text{otherwise} \end{cases}.$$

Both for the diagonal and main grids, if there is strict equality in the two counts detected, no grid is considered detected. Naturally, if no decision is taken on the diagonal, no main grid is selected either. The grid detection is detailed in Algorithm 3.

While Δ_{main} is later used to make decisions on forgeries, the two intermediary comparisons $\Delta_{\begin{smallmatrix}\mathrm{R} & \mathrm{G} \\ \mathrm{G} & \mathrm{B}\end{smallmatrix}\_\begin{smallmatrix}\mathrm{B} & \mathrm{G} \\ \mathrm{G} & \mathrm{R}\end{smallmatrix}}$ and $\Delta_{\begin{smallmatrix}\mathrm{G} & \mathrm{R} \\ \mathrm{B} & \mathrm{G}\end{smallmatrix}\_\begin{smallmatrix}\mathrm{G} & \mathrm{B} \\ \mathrm{R} & \mathrm{G}\end{smallmatrix}}$ are easier to understand visually, and are thus kept for visualization.

Our implementation of the count difference computation is slightly different from the description of the original article. In the original article, the difference is not normalised by $\frac{1}{2XY}$. More importantly, the difference is computed separately in the red and blue channels, and the strongest of the two is kept, whereas we use their sum. The reason for this is that the original article only tries to classify in which pattern an image has been sampled, without considering how confident one can be in the detection, or how to use it to detect forgeries. When only considering classification of an image or window into the four patterns, both the original article and our implementation provide the same results. However, adding the normalization and summing the two channels makes it easier for us to also compute a confidence value for the detections, which will be described in the next subsection.

Finally, we note that even though this algorithm is presented for one window, the grid detection is obviously performed on all windows simultaneously.

Forgery detection

Using the previously-described algorithms, we can compute the intermediate value masks in all channels, cut them into windows, and detect the diagonal and pattern of the global image and of each window. With this information, we could simply say that the windows which do not use the same pattern as the main grid correspond to forged regions. However, doing this creates many false positives, as the detection is not always correct. In first instance, if the grid of a window does not match the global image's grid, we can consider that window as forged with a confidence of $|\Delta_{\text{main}}|$ (or $|\Delta_{\text{diag}}|$ if looking at the diagonals). However, if the threshold is low, isolated detections of a given grid will be made by mistake. On the contrary, in a region with many windows sharing the same grid, only those above the threshold will be detected, so a high threshold will cause most of the detections to be missed. In both cases, using a fixed threshold will lead to mistakes that would be easy to avoid by looking at the map more globally.

We therefore propose to segment the windows into connected components by their grids. In other words, a connected component is a set of spatially connected windows whose detected pattern is the same. This segmentation is performed with `scikit-image` [89]. Components whose detected pattern is equal to the one of the global image are immediately discarded; they are not considered forged as they agree with the full image. For components whose detected pattern is different, we consider them as forged, with a confidence value which corresponds to the maximum absolute difference of count of all windows in that components (either $|\Delta_{\text{main}}|$ or $|\Delta_{\text{diag}}|$ depending on whether we are looking at the full pattern or the diagonal). In other words, the confidence of a component is the confidence of its most prominent window. The computation of the confidence by connected component is performed in Algorithm 4.

We apply this method separately to both the diagonal and the full pattern detection yields two confidence maps. We merge those two confidence maps into one by taking their pointwise maximum. Although the full pattern analysis can encompass the diagonal detection, in many cases the algorithm can only find the diagonal but hesitates on the full pattern. Hence, separating the detections enables the method

Algorithm 3: Find the grid

1 **function** `find_grid(R, G, B)`

  Input **R, G, B:** Arrays of even size $(2X, 2Y)$, as returned by `is_intermediate` or a sub-window of it, on the three colour channels

  Output **M:** Detected CFA pattern (one of $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$, $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$, $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$, $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$)

  Output **D:** Detected diagonal pattern ($\begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}$ or $\begin{smallmatrix}G&\cdot\\\cdot&G\end{smallmatrix}$)

  Output Δ_{main}**:** Difference of count of intermediate values between the two patterns sharing the same diagonal. Positive if the best pattern is $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ or $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$, negative if the best pattern is $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ or $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$.

  Output Δ_{diag}**:** Difference of count of intermediate values between the two diagonal patterns. Negative for $\begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}$, positive for $\begin{smallmatrix}G&\cdot\\\cdot&G\end{smallmatrix}$.

  `# First we select the best diagonal pattern using the green values`

2 $\mathrm{C}_{\begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}} \coloneqq \sum_{x=0}^{X}\sum_{y=0}^{Y} G_{2x,2y+1} + G_{2x+1,2y}$

3 $\mathrm{C}_{\begin{smallmatrix}G&\cdot\\\cdot&G\end{smallmatrix}} \coloneqq \sum_{x=0}^{X}\sum_{y=0}^{Y} G_{2x,2y} + G_{2x+1,2y+1}$

4 $\Delta_{\text{diag}} \coloneqq \frac{1}{2XY}\left(\mathrm{C}_{\begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}} - \mathrm{C}_{\begin{smallmatrix}G&\cdot\\\cdot&G\end{smallmatrix}}\right)$

5 $D \coloneqq \begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}$ **if** $\Delta_{\text{diag}} < 0$ **else** $\begin{smallmatrix}G&\cdot\\\cdot&G\end{smallmatrix}$

  `# Compare patterns with the same diagonal.`

6 $\mathrm{C}_{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} \coloneqq \sum_{x=0}^{X}\sum_{y=0}^{Y} R_{2x,2y} + B_{2x+1,2y+1}$

7 $\mathrm{C}_{\begin{smallmatrix}B&G\\G&R\end{smallmatrix}} \coloneqq \sum_{x=0}^{X}\sum_{y=0}^{Y} R_{2x+1,2y+1} + B_{2x,2y}$

8 $\mathrm{C}_{\begin{smallmatrix}G&R\\B&G\end{smallmatrix}} \coloneqq \sum_{x=0}^{\frac{X}{2}} R_{2x+1,2y} + B_{2x,2y+1}$

9 $\mathrm{C}_{\begin{smallmatrix}G&B\\R&G\end{smallmatrix}} \coloneqq \sum_{x=0}^{\frac{X}{2}} R_{2x,2y+1} + B_{2x+1,2y}$

10 $\Delta_{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}\_\begin{smallmatrix}B&G\\G&R\end{smallmatrix}} = \frac{1}{2XY}\left(\mathrm{C}_{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} - \mathrm{C}_{\begin{smallmatrix}B&G\\G&R\end{smallmatrix}}\right)$

11 $\Delta_{\begin{smallmatrix}G&R\\B&G\end{smallmatrix}\_\begin{smallmatrix}G&B\\R&G\end{smallmatrix}} = \frac{1}{2XY}\left(\mathrm{C}_{\begin{smallmatrix}G&R\\B&G\end{smallmatrix}} - \mathrm{C}_{\begin{smallmatrix}G&B\\R&G\end{smallmatrix}}\right)$

12 **if** $D = \begin{smallmatrix}\cdot&g\\g&\cdot\end{smallmatrix}$ **then**

13 $\quad\Delta_{\text{main}} = \Delta_{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}\_\begin{smallmatrix}B&G\\G&R\end{smallmatrix}}$

14 $\quad M \coloneqq \begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ **if** $\Delta_{\text{main}} < 0$ **else** $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$

15 **else**

16 $\quad\Delta_{\text{main}} = \Delta_{\begin{smallmatrix}G&R\\B&G\end{smallmatrix}\_\begin{smallmatrix}G&B\\R&G\end{smallmatrix}}$

17 $\quad M \coloneqq \begin{smallmatrix}G&R\\B&G\end{smallmatrix}$ **if** $\Delta_{\text{main}} < 0$ **else** $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$

18 **return** $M, D, \Delta_{\text{main}}, \Delta_{\text{diag}}$

to detect significant diagonal traces even when the full pattern cannot be detected. These confidence maps are useful to visualize the detection. However, they do not constitute by themselves a decision on the detection. They cannot either be used as a heat map: the maximal absolute value of the difference, 1, is never reached in actual cases, and even the most confident detections will rarely reach a score of 0.3.

To make a final decision on the image, we thus threshold the obtained confidence map by a given threshold γ. This is equivalent to performing hysteresis

thresholding with a lower threshold o and a higher threshold γ on each map $(M = g) \odot |\Delta_{\text{main}}|$ for each pattern g except the full image's pattern, and $(D \neq d_{\text{img}}) \odot |\Delta_{\text{diag}}|$, where d_{img} is the full image's detected diagonal, $\odot$ is the Hadamard product (pointwise multiplication), while the expression $(A = b)$ is an array of the same shape of A, equal to 1 where A takes the value b and 0 elsewhere.

Finally, all the outputs are resized to have one value per pixel, rather than per window. This is done with nearest neighbours interpolation for binary outputs, and with linear interpolation for continuous outputs. The computation of the forgery map is detailed in Algorithm 5.

Overall, the full algorithm can achieve linear complexity in the input size. Indeed, each individual step is linear, including the connected confidence computation since each block of the image belongs to at most one component and is thus only processed once. In practice, the vectorized Python implementations processes all blocks for each component, thus leading to a quadratic complexity. Although an optimal computation in another language could offer the optimal worst-case linear complexity, this is largely irrelevant since the number of inconsistent connected components usually does not scale linearly with the image size.

Algorithm 4: Connected confidence computation

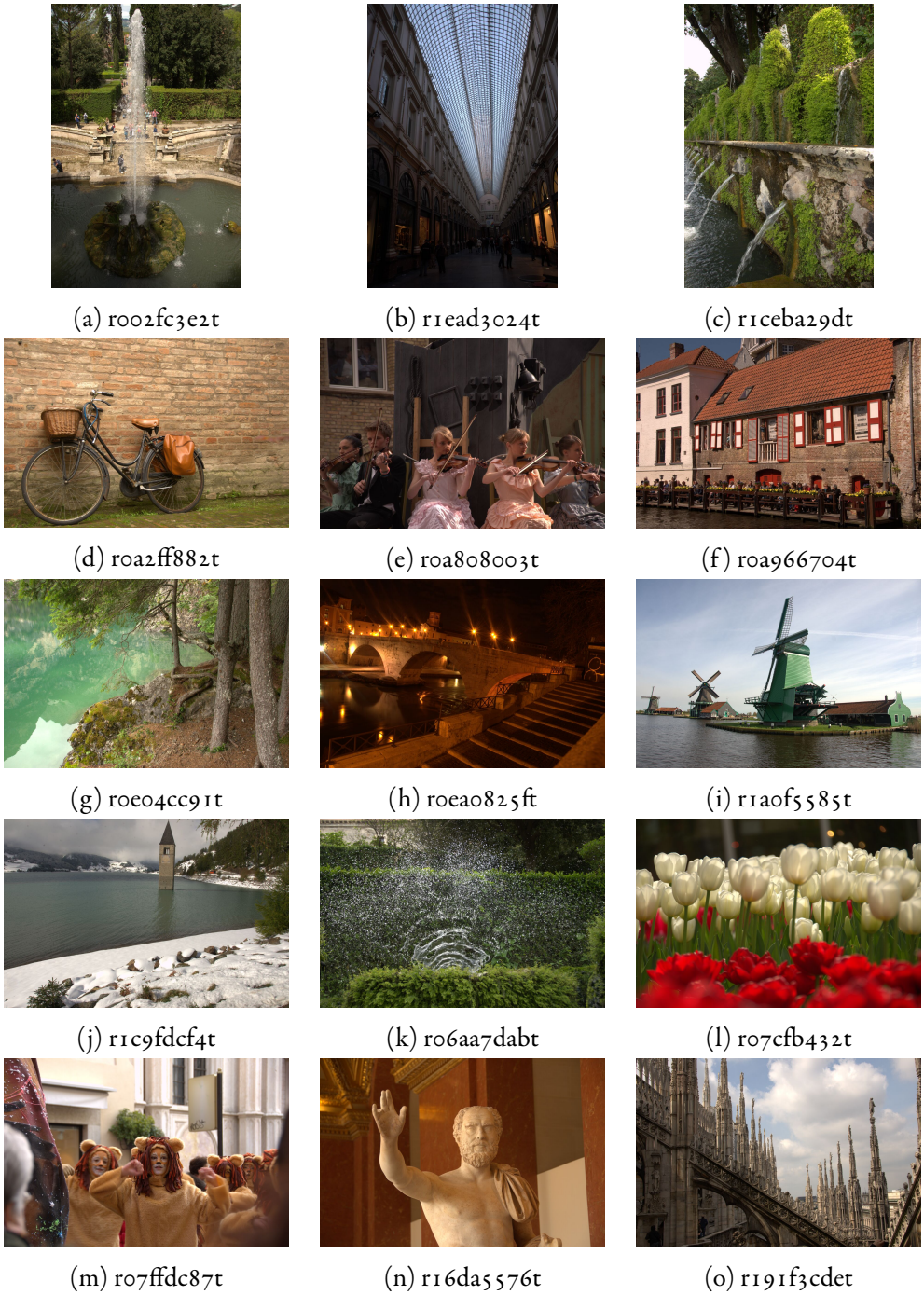
```
1 function connected_confidence(G, global_G, Δ)
      Input G: Grid/diagonal detected on each window, shape (X_W, Y_W)
      Input global_G: Grid/diagonal detected on the main image
      Input Δ: Either Δ_main or Δ_diag
      Output confidence: Confidence that each pixel is forged
2     labels ≔ label_connected(G, global_G)
3     confidence ≔ 0_{X_W, Y_W}
4     for label from 0 to max(labels) do
          # ⊙ denotes Hadamard product
5         confidence += max((labels = label) ⊙ |Δ|)
6     return confidence
```

Algorithm 5: Global algorithm

```
function find_forgeries(img, W, stride, threshold)
    Input img: Input image, size (X, Y, 3). X and Y must be even (the last
               row and/or column may be cut to ensure this).
    Param W: int, Window size
    Param stride: int, Distance between the left/top border of two
                  consecutive windows. Must divide W.
    Param γ: float, higher hysteresis threshold to select relevant
             inconsistencies.
    Output forged_full: Final map of detected forgeries (pointwise
                        maximum of forged_main and forged_diag)
    Output forged_{main, diag}: Detected forgeries after thresholding,
                                respectively on the full pattern and on
                                the diagonal
    Output confidence_{full, main, diag}: Confidence that each region is
                                          a forgery
    intermediate ≔ is_intermediate(img)
    windows ≔ create_sliding_windows(intermediate, W, stride)
    X_w, Y_w ≔ number of windows per column/row
    # Pattern and diagonal on the global image
    global_M, global_D, _, _, _, _ = find_grid(intermediate[:, :
      , 0], intermediate[:, :, 1], intermediate[:, :, 2])
    # Pattern and diagonal on each window
    M, D, Δ_main, Δ_diag ≔ 0_{X_w, Y_w}
    for x from 0 to X_w and y from 0 to Y_w do
        main_{x,y}, diag_{x,y}, Δ_{main_{x,y}}, Δ_{diag_{x,y}} ≔
          find_grid(windows_{x,y,0}, windows_{x,y,1}, windows_{x,y,2})
    # Inconsistent regions
    bad_{main, diag}_raw ≔ {M, D} ≠ global_{main, diag}
    bad_full_raw ≔ max(bad_diag_raw, bad_main_raw)
    # Connected confidence
    confidence_main ≔ connected_confidence(M, global_M, Δ_main)
    confidence_diag ≔ connected_confidence(D, global_D, Δ_diag)
    confidence_full ≔ max(confidence_main, confidence_diag)
    # Threshold
    forged_{full, main, diag} ≔ confidence_{full, main, diag} > γ
    return forged_{full, main, diag}, confidence_{full, main, diag}
```

3.3 Experiments

To evaluate the ability of this method to detect the CFA pattern correctly, we took 15 images from the Raise dataset [31], and demosaiced them using the 7 algorithms available in LibRaw: Bilinear interpolation, AAHD, AHD, DCB, DHT, PPG and VNG. 11 of these images are of size 4948×3280, the other 4 are of size 4310×2868. The selected images can be seen in Figure 3.3.



(a) r002fc3e2t (b) r1ead3024t (c) r1ceba29dt

(d) r0a2ff882t (e) r0a8080o3t (f) r0a966704t

(g) r0e04cc91t (h) r0ea0825ft (i) r1a0f5585t

(j) r1c9fdcf4t (k) r06aa7dabt (l) r07cfb432t

(m) r07ffdc87t (n) r16da5576t (o) r191f3cdet

Figure 3.3: These 15 images from the Raise dataset [31] were used during our experiments.

CFA pattern detection

We started by analyzing, at a global scale, whether the method is able to detect the correct pattern of the 15 images described above. Results can be seen in Table 3.1. One can see that the algorithm detects the correct grid in all 15 images when they were demosaiced with bilinear, AHD or DCB demosaicing. It also worked well on the PPG and VNG algorithms, despite a few mistakes in the full pattern identification against PPG or VNG-demosaiced images. These mistakes were solved when using bidirectional filters. When the image was demosaiced with AAHD or DHT, however, the algorithm consistently failed to detect even the diagonal, and consequently also fails on the full pattern, in both versions of the algorithm.

Demosaicking	Diagonal	Full pattern
AAHD	0/15	0/15
AHD	15/15	15/15
DCB	15/15	15/15
DHT	3/15	3/15
Bilinear	15/15	15/15
PPG	15/15	13/15
VNG	15/15	14/15

(a) Original isotropic intermediate values

Demosaicking	Diagonal	Full pattern
AAHD	0/15	0/15
AHD	15/15	15/15
DCB	15/15	15/15
DHT	2/15	2/15
Bilinear	15/15	15/15
PPG	15/15	15/15
VNG	15/15	15/15

(b) Bidirectional filters for intermediate values

Table 3.1: Identification of the main diagonal and of the full pattern on the 15 images. For each demosaicing algorithm, we show how many of the 15 images had their diagonal/full pattern correctly detected by the method. In its original version, the algorithm works very well when the demosaicing is done with AHD, DCB or Bilinear demosaicing, with a few errors on the full pattern against PPG- or VNG-demosaiced images. It fails to detect even the diagonal on AAHD- and DHT-demosaiced images. Bidirectional filters for intermediate value computation yields perfect results on PPG and VNG, but still fails against AAHD- and DHT-demosaiced images.

Looking at the results on image r0a2ff882t in Figure 3.4, we can see again that the results depend on the demosaicing algorithm used by the method. All windows were detected correctly against DCB and Bilinear demosaicing, but the algorithm was confused on the diagonal pattern in the PPG-demosaiced image, though bid-

irectional filters for the intermediate value computation partly alleviated this problem. On the VNG-demosaiced image, there are also false detections on the diagonal itself. More importantly, the basket of the bike caused errors with AHD, PPG and VNG demosaicing. This was to be expected with a periodic structure that fools the detection. The result might easily be misinterpreted as a forgery. As can be seen on Figure 3.5, however, using bidirectional filters yields a very low relative confidence for the identification of the basket's grid compared to the rest of the image. As a consequence, a reasonable thresholding level should still enable one to automatically discard this false detection.

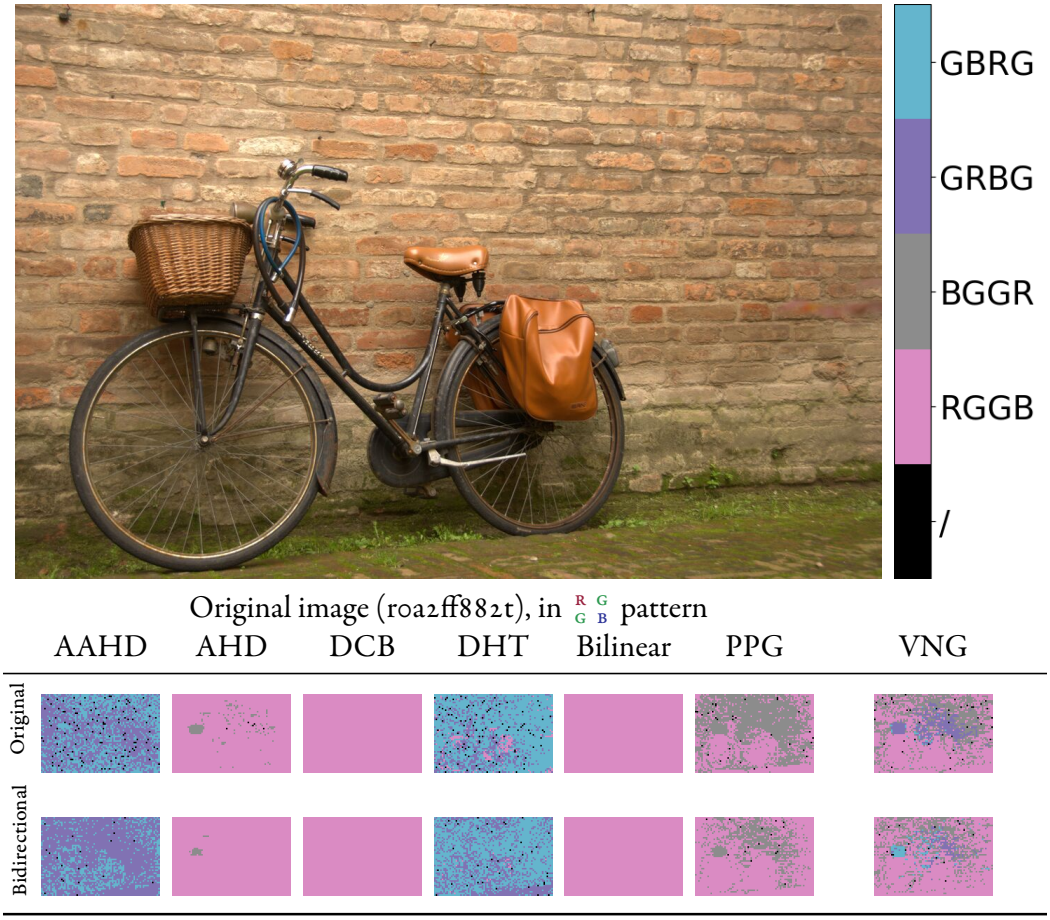
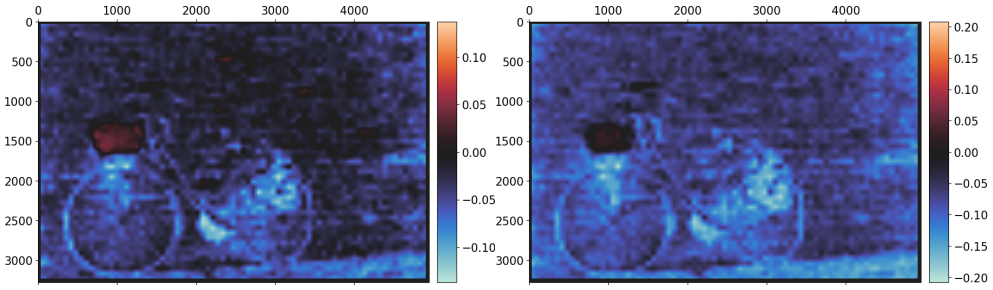


Figure 3.4: Results of the method on 64×64 windows, both with the original isotropic intermediate value mask and the proposed bidirectional one, on one image with the 7 different demosaicing algorithms. Both methods work perfectly on the DCB- and Bilinear-demosaiced images. With the AHD, PPG and VNG methods, both the original isotropic and the bidirectional filters have trouble discerning between the two patterns sharing the same diagonal, but the bidirectional detection makes fewer mistakes. Periodically textured regions like the basket can create a localized shift in the detected mosaic, which could be mistaken for a forgery. With the AAHD and DHT algorithm, the method consistently detects the wrong diagonal.



(a) Isotropic (b) Bidirectional

Figure 3.5: This figure shows, on the AHD-demosaiced bicycle image, the difference of counts of intermediate values corresponding to the $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ and $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ patterns, on the red and blue channels. This count is what is used by the algorithm to decide on a grid. A negative difference corresponds to the correct $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ pattern, a positive difference to the incorrect $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ pattern. The difference is normalized by dividing it by the size of the block (64×64). The texture in the basket area leads to a locally consistent shift in the position of the intermediate values. The error is slightly less prominent when a bidirectional mask is used, but is still consistently in favour of the wrong grid.

Most images that are found on the web are JPEG-compressed. It is thus vital to test the robustness of this algorithm to JPEG compression. JPEG compression quickly discards the highest frequencies, at which CFA artefacts are located. As a consequence, it would be illusory to expect results on heavily-compressed images. However, being able to detect the CFA pattern on low-compression images extends the application range of a CFA grid detection method. We show results after JPEG compression on Figures 3.6, 3.7. JPEG compression was done with the Pillow library [90]. On the two studied images, we can see that even the highest-quality compression with a JPEG quality factor of 100 caused many errors in the pattern detection, though the algorithm remained largely usable, especially when only looking at the diagonal. A quality factor of 100 does not specifically remove the high frequencies, however the discretization in the frequency domain already includes a loss of information. At JPEG quality 98, the algorithm no longer detected the correct pattern, except in the easier case of the bilinear demosaicing algorithm. However, it could still detect the diagonal of most windows, albeit with a few errors. Finally, at JPEG quality 95, the algorithm was unable to find anything.

All in all, JPEG compression remains the biggest limitation of this method, and of CFA detection in general.

In Figures 3.8, 3.9, 3.10, we evaluate the robustness of the method to additive white Gaussian noise (AWGN). Because AWGN is not spatially correlated, it remains possible to detect the pattern in most cases with a noise of standard deviation $\sigma = 5$ (on $[0, 255]$-ranged images). More localized errors are made as the noise level increases, but thanks to the lack of spatial correlation of the noise (and consequently of the errors), the risk of mistakenly interpreting these as forgeries remains relatively low. Finally, we can see in Figure 3.10 that detecting the pattern over AWGN is made easier by using a larger window size, which averages the noise while keeping the artefacts. Of course, this comes at the price of potentially missing

smaller forgeries.

Median filtering has often been proposed as a counter-forensics measure to hide forgeries. Although it can be easily detected [91], we evaluate the robustness of the presented method to median filtering in Figure 3.11, using a median filter of footprint $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$. We can see that the results of the method are completely inverted, because median filtering shifts the intermediate values. As a consequence, images on which the correct diagonal was found before filtering now yield wrong detection, whereas the method finds the correct pattern on AAHD- and DHT-demosaiced images, where it was failing without median filtering. Figure 3.12 explains this phenomenon with a toy example. Without further elaborating, we note that only the diagonal detection is affected. As a consequence, if median filtering has been detected, detections can be made correct again by simply reverting the detected diagonal.

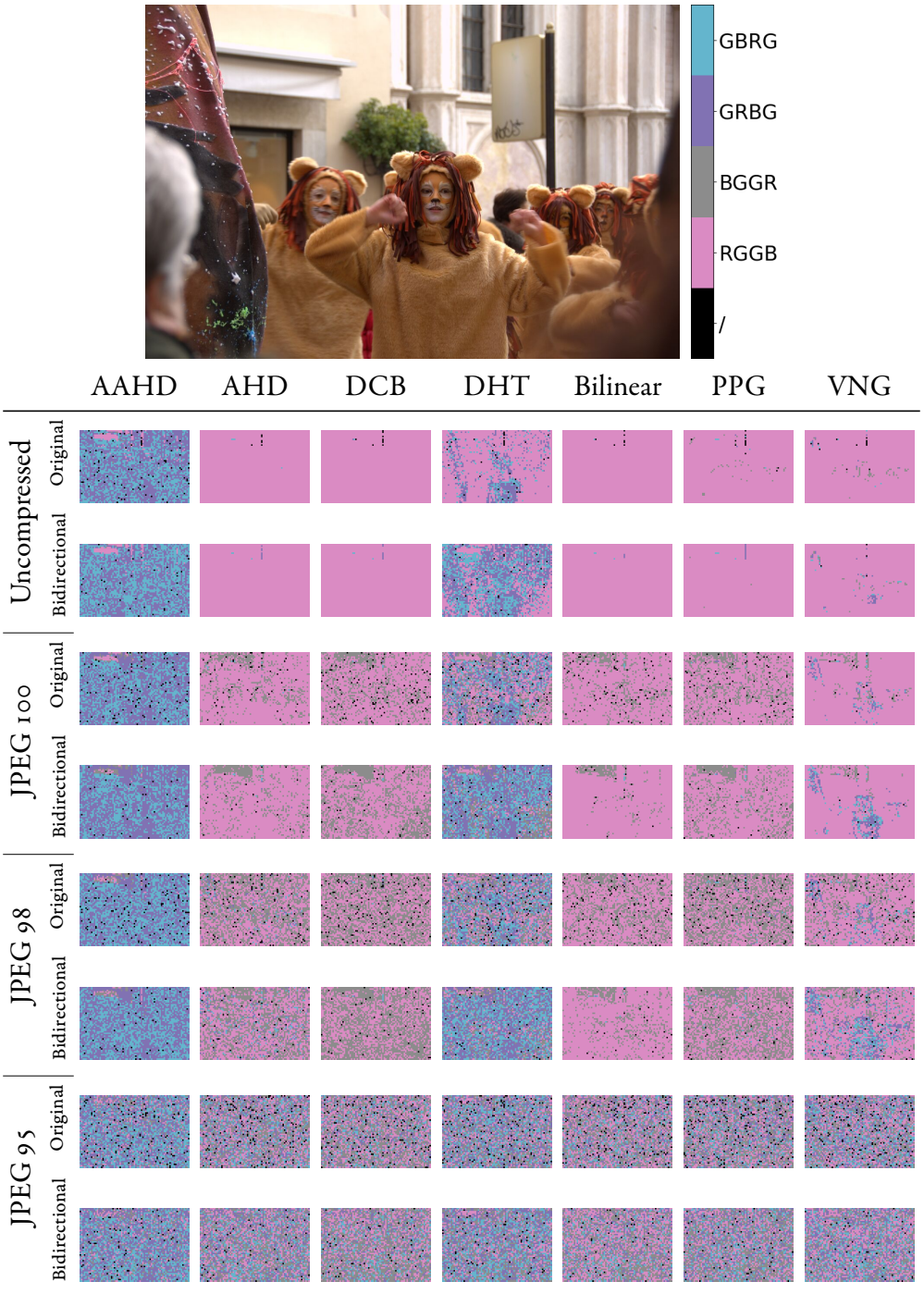


Figure 3.6: Detection of the method after JPEG compression. Results are shown on image r07ffdc87t, in $\begin{smallmatrix} R & G \\ G & B \end{smallmatrix}$ pattern, uncompressed and submitted to JPEG compression of quality 100, 98 and 95. Detections can still be made to some extent up to a compression level of 98, at least for the diagonal pattern. At quality 95, however, no detections are possible.

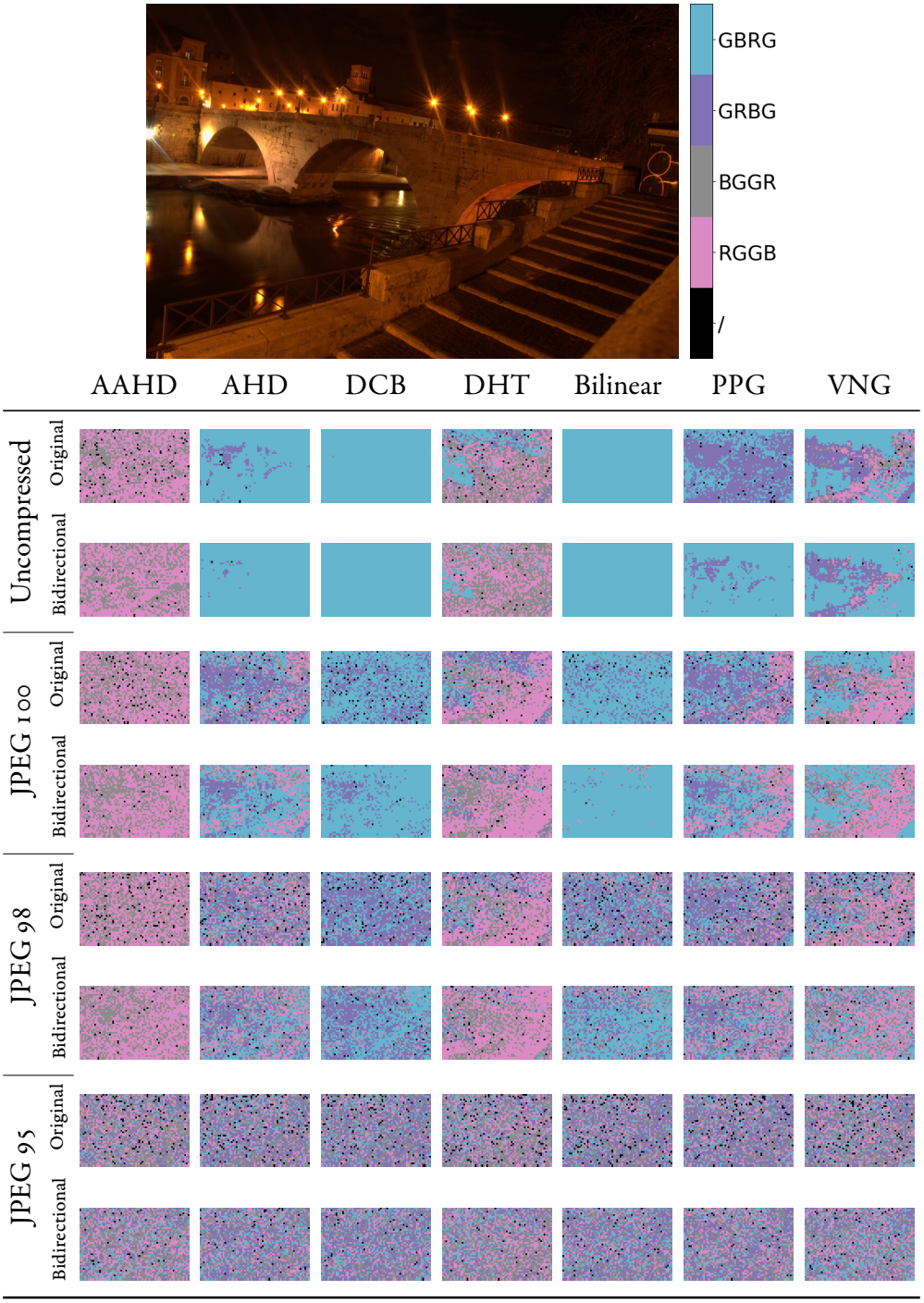


Figure 3.7: Detection of the method after JPEG compression. Results are shown on image r0ea0825ft, in $\begin{smallmatrix} G & R \\ B & G \end{smallmatrix}$ pattern, uncompressed and submitted to JPEG compression of quality 100, 98 and 95. On this image, which is more difficult to analyze than the one in Figure 3.6, errors are already present in the uncompressed image, the diagonal is also locally wrong on the stairs against VNG demosaicing, especially with the original isotropic intermediate values. These errors become more prominent against other demosaicing methods as well at JPEG quality 100 (highest possible), and detection becomes barely possible. At JPEG quality 98, contrarily to Figure 3.6, detection is mostly impossible, although the diagonal can still be found with local mistakes against bilinear and DCB demosaicing if bidirectional filters are used. Again, bidirectional intermediates provide a consistent, although small boost to JPEG robustness.

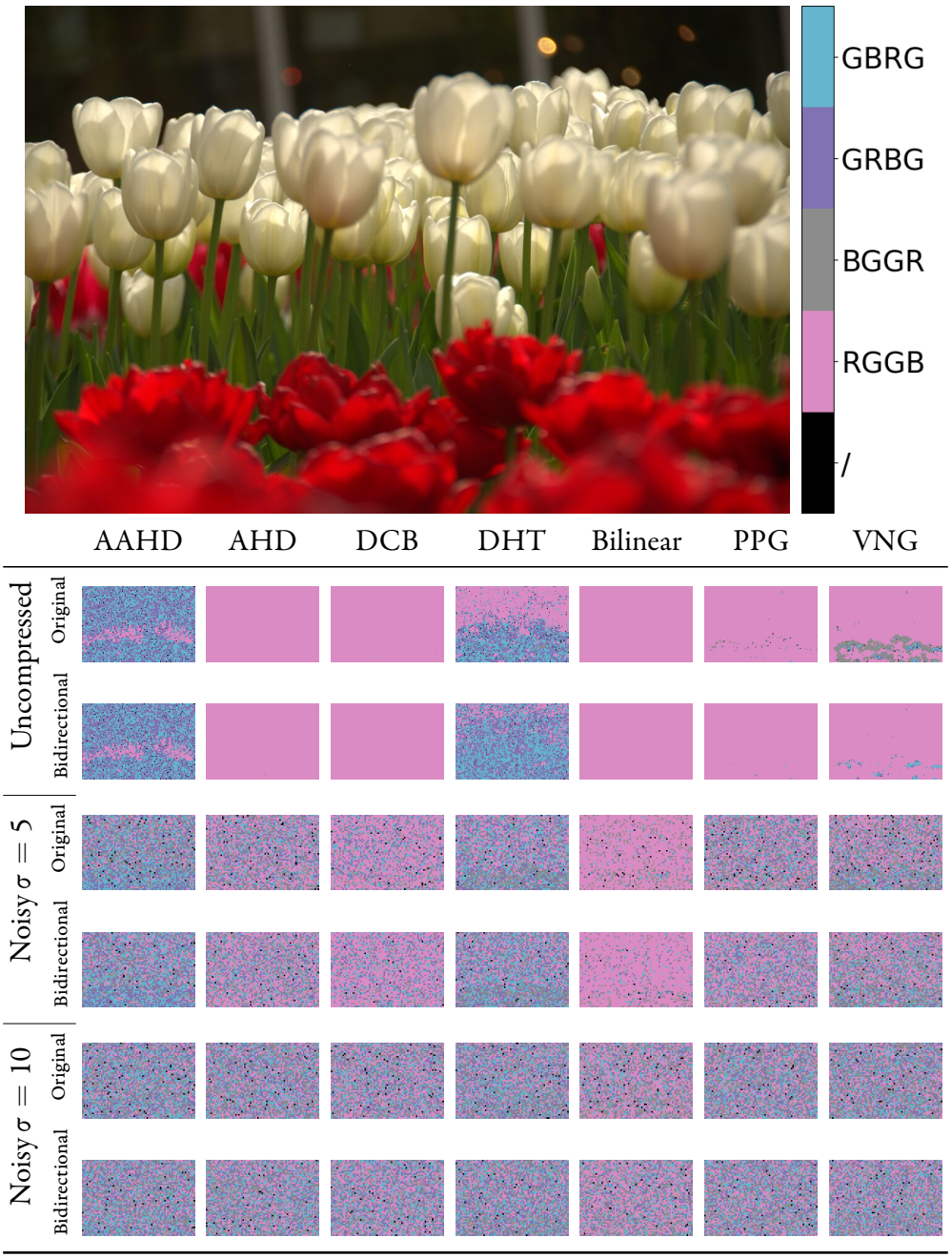


Figure 3.8: Robustness of the method to additive white Gaussian noise (AWGN), that can be added to images either for aesthetic reasons or to maliciously hide manipulations. Image r07cfb432t, in $\begin{smallmatrix}R & G\\ G & B\end{smallmatrix}$ pattern. We show results against noise of standard deviation from 0 (noiseless) to 10, window size 64×64. Because the noise is independent of the image, it does not create locally coherent errors that can hardly be distinguished from forgeries. However, the probabilities of a sampled or interpolated pixel being an intermediate value go closer to one another as more noise is added, making the detection harder.

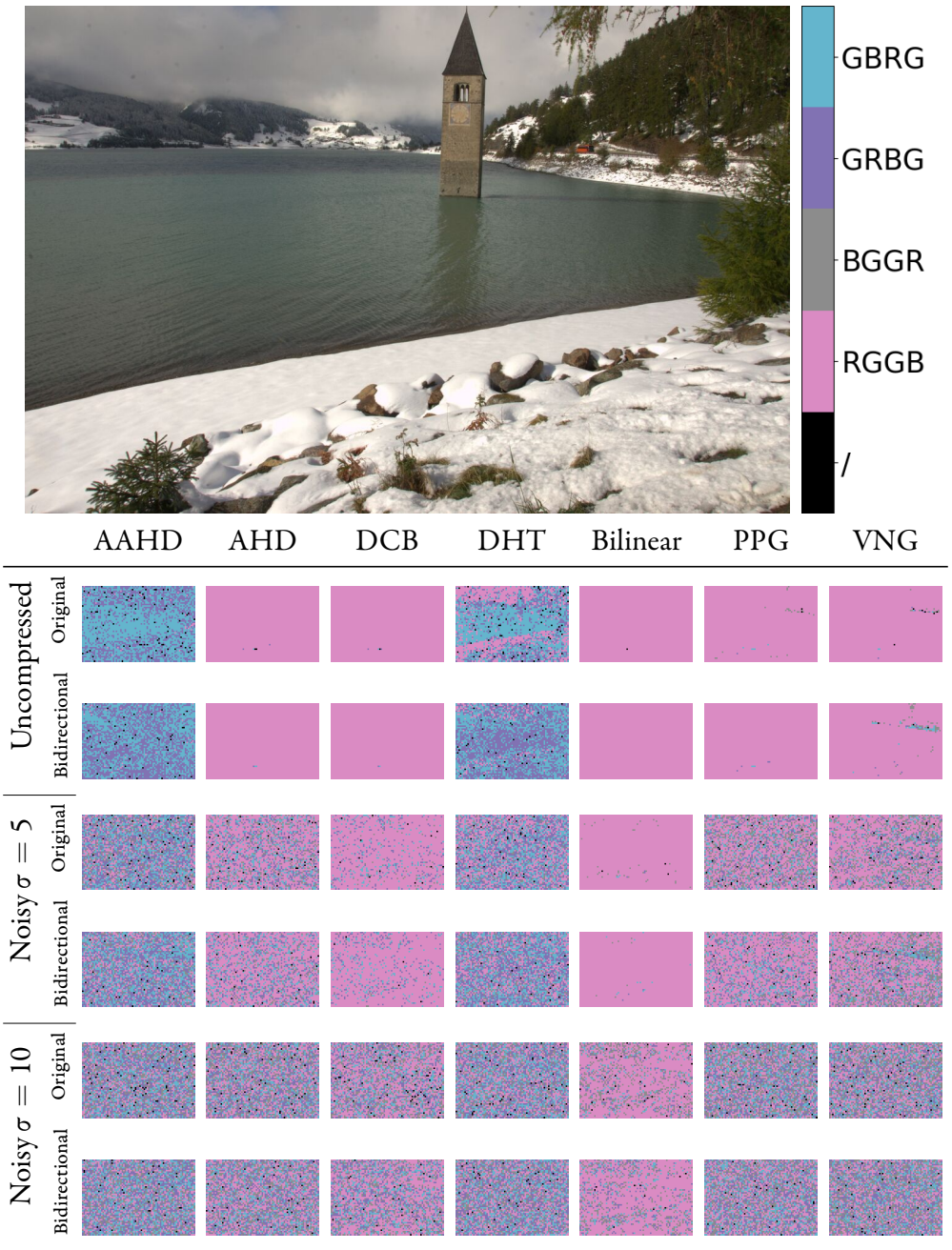


Figure 3.9: Robustness of the method to additive white Gaussian noise (AWGN), that can be added to images either for aesthetic reasons or to maliciously hide manipulations. Image r1c9fdcf4t, in $\begin{smallmatrix} R & G \\ G & B \end{smallmatrix}$ pattern. We show results against noise of standard deviation from 0 (noiseless) to 10, with window size 64×64. Because the noise is independent to the image, it does not create locally coherent errors that can hardly be distinguished from forgeries. However, the probabilities of a sampled or interpolated pixel being an intermediate value go closer to one another as more noise is added, making the detection harder.

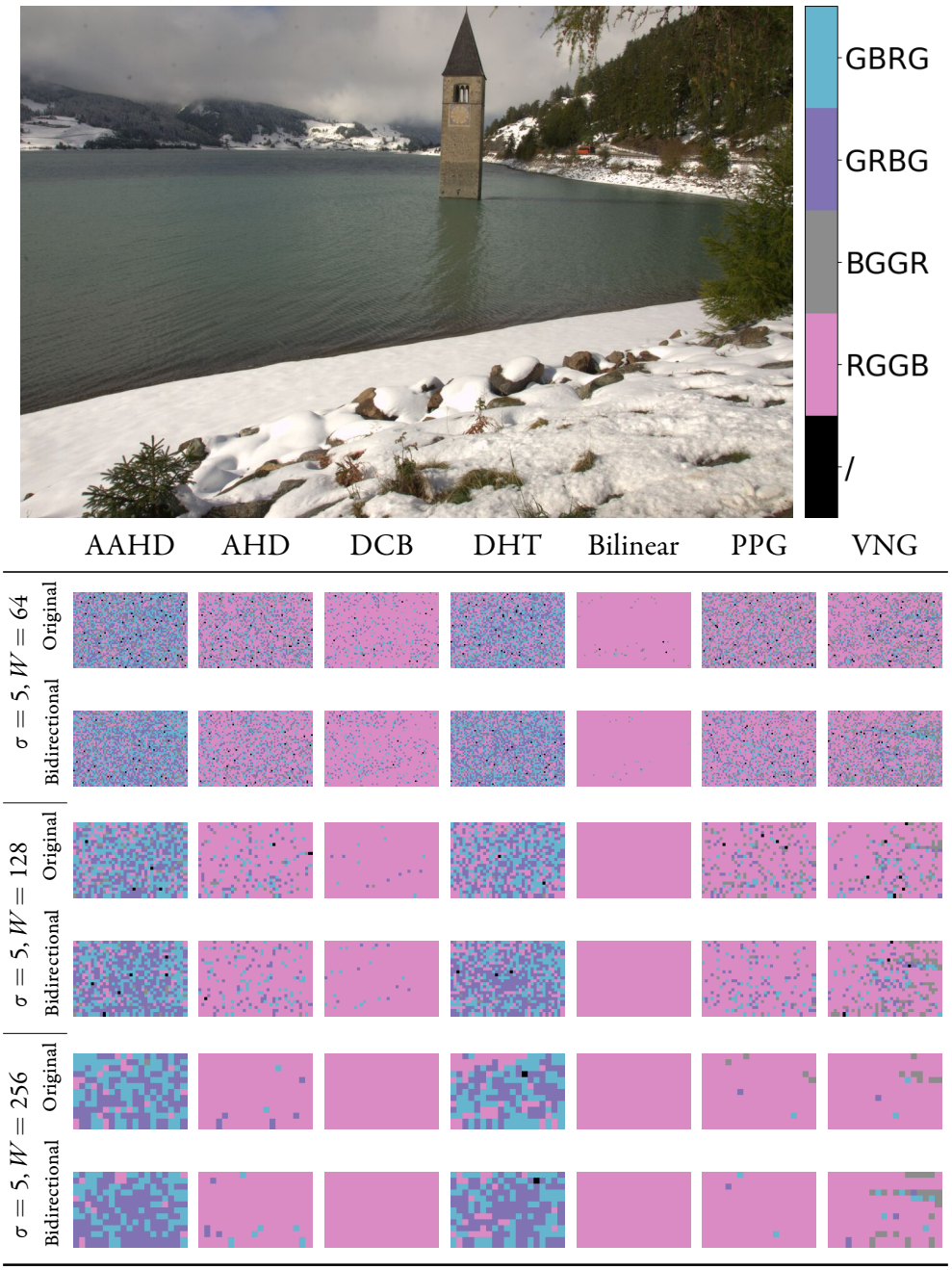
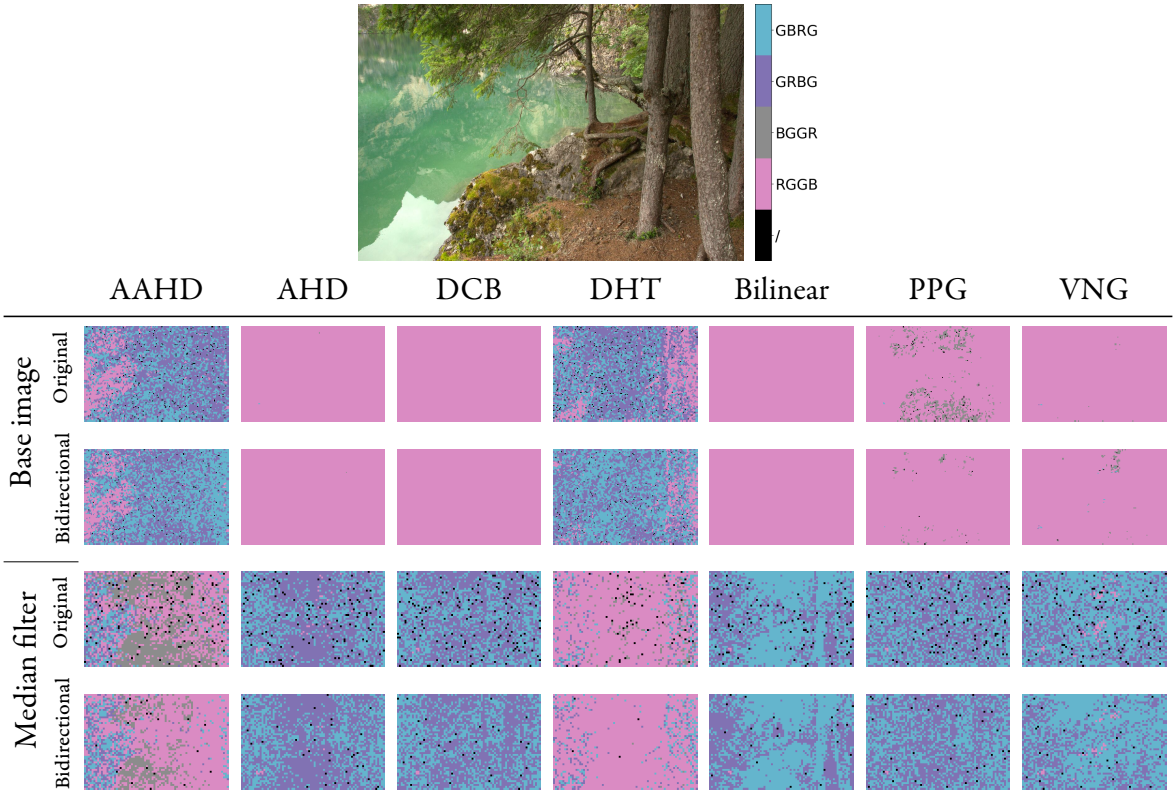
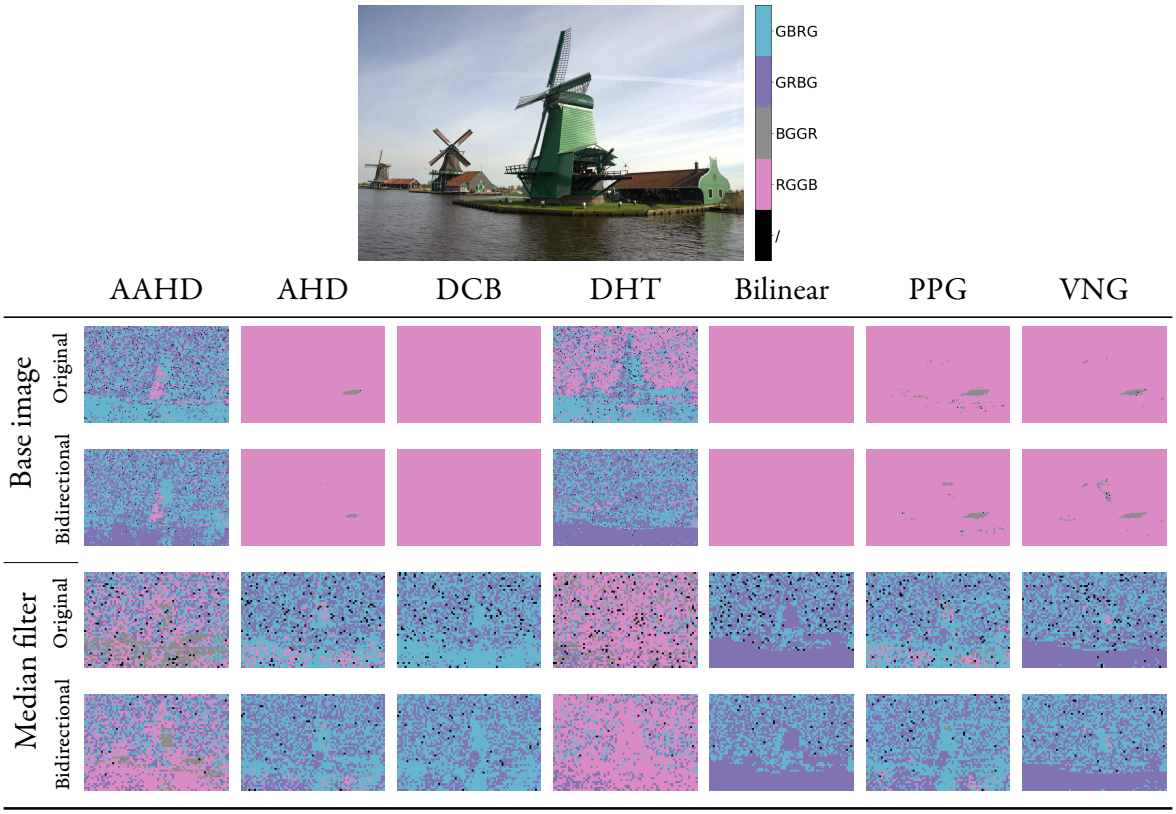


Figure 3.10: Robustness of the method to additive white Gaussian noise (AWGN), that can be added to images either for aesthetic reasons or to maliciously hide manipulations. Image r1c9fdcf4t, in $\begin{smallmatrix}\mathrm{R} & \mathrm{G}\\ \mathrm{G} & \mathrm{B}\end{smallmatrix}$ pattern. Noise standard deviation 5, with window sizes 64×64, 128×128 and 256×256. Because the noise is independent of the image and not spatially correlated, using bigger windows improves the robustness to it by providing more samples (at the cost of potentially missing smaller forgeries).

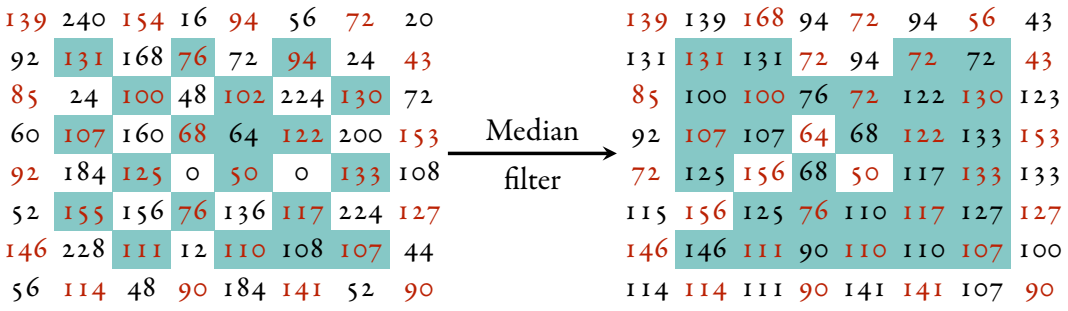


(a) Image r0e04cc91t, in $\begin{smallmatrix} R & G \\ G & B \end{smallmatrix}$ pattern.



(b) Image r1a0f5585t, in $\begin{smallmatrix} R & G \\ G & B \end{smallmatrix}$ pattern

Figure 3.11: Robustness to median filtering on 64×64 blocks. Median filtering shifts the green intermediate values, confusing the algorithm on the diagonal. It actually balances out the similar confusion caused by AAHD and DHT, leading to a better detection in those cases.



(a) Original array

(b) After median filtering

Figure 3.12: Values on an array, before and after median filtering. The array corresponds to the green channel of an image in the $\begin{smallmatrix} \cdot & G \\ G & \cdot \end{smallmatrix}$ position demosaiced with bilinear interpolation. Red values correspond to positions where the value was interpolated. Highlighted cells correspond to pixels that take an intermediate value. Notice the shift of intermediate values from one diagonal to another: originally, almost all (18 out of the 20) intermediate values are found in the interpolated pixels, but after median filtering, most (16 out of 29) are located in the sampled (black) position

Image forgery detection

The ultimate goal of the method is to find mosaic inconsistencies in an image. We used forgeries from the Trace database introduced in Chapter 1 (Non-Semantic Evaluation of Image Forensics Tools) to evaluate the method. The Trace database is constituted of 1000 images taken from the Raise dataset. Two forgery masks are made for each image: the endomask, obtained by taking a random object from the image's automatic segmentation, and the exomask, which is simply the endomask of another image of the set and thus do not correlate to the contents of the image. The concept of the database is to process the image with two different pipelines, and merge them with one of the forgery masks. Of the six datasets that are proposed, two are of interest to us:

- in the CFA Grid dataset, the two pipelines are the same, but the pattern of demosaicing changes (the algorithm is the same). The forgery thus has a different CFA pattern than the rest of the image.
- in the CFA Algorithm dataset, the two pipelines are the same, but the algorithm of demosaicing changes. A new CFA pattern is also chosen at random for the forged region, with a $\frac{1}{4}$ chance of being the same than the original image's.

For the quantitative experiments, we used the CFA grid with exomasks dataset. For the qualitative experiments, we used samples from both the CFA grid and CFA algorithm datasets. Unless otherwise specified, quantitative experiments were done with the Matthews Correlation Coefficient (MCC) [80]. This metric varies from -1 for a detection that is complementary to the ground truth, to 1 for a perfect detection. A score of 0 represents an uninformative result and is the expected performance of any random classifier. The MCC is more representative than the F1 and IoU scores [81], [82], partly as it is less dependent on the proportion of positives in the ground truth, which is especially important given the large variety of forgery mask sizes in the database. It is defined by

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}},$$

where TP, FP, TN and FN respectively represent the numbers of true positives (TP), false positives(FP), true negatives (TN) and false negative (fn). The score is computed for each image, and then averaged over each dataset. As the method does not provide a binary output but a continuous heatmap, we weighted the confusion matrix using the heatmap. For several results, we also provide the Intersection over Union (IoU), the F1 score and the Precision and Recall. Quantitative experimental results can be found in Table 3.2.

On Table 3.2a, we can see that using bidirectional filters slightly improves the overall results. This corroborates the visual results of the previous subsection. Using thresholding not only improves the understandability of the method, it also provides significant improvements over the scores.

Table 3.2c shows results from taking only the results of the diagonal detection, the full pattern or their combination by pointwise maximum. The strategy of merging the two maps by pointwise maximum is the good one: it performs almost as well as the diagonal map on forgeries who do not share their diagonal, almost as well as the full map on forgeries that do share their diagonal (and are thus invisible in the diagonal map), and thus performs much better on the overall database than any of the maps taken separately.

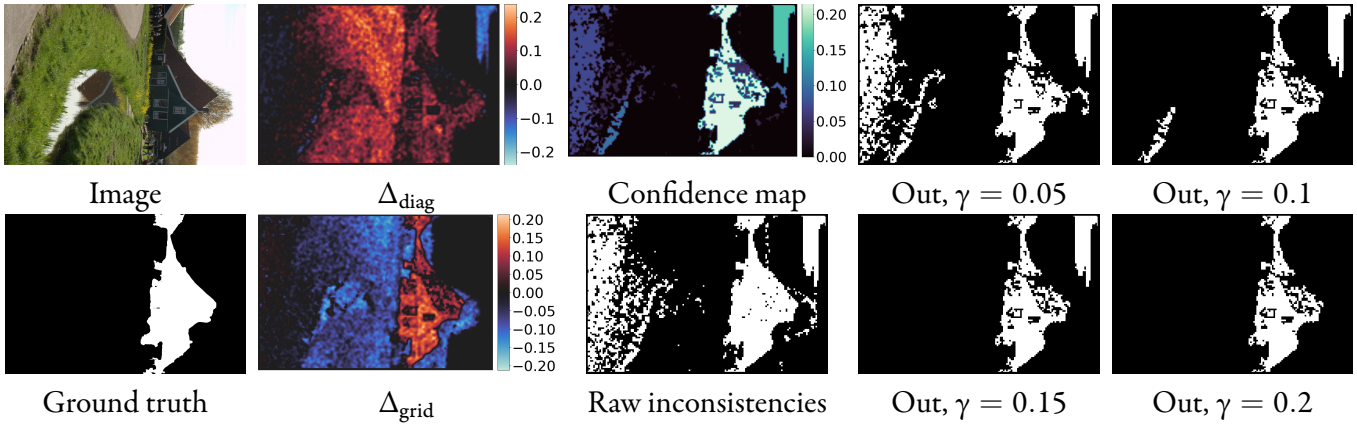
On Table 3.2d, we present the scores of the method depending on the demosaicing used to process the image. Unsurprisingly, the method does not work well on AAHD- and DHT-demosaiced images, as we saw in the previous subsection. However, because it is consistent in detecting the diagonal, it can still be used to see that two AAHD- or DHT-demosaiced regions use a different diagonal. This will be explored further below. Bidirectional filters work better than the original isotropic filters in most cases. The biggest gaps occur with PPG and AHD demosaicing, which explicitly interpolate in the smoothest direction [92], [93]. On the other hand, isotropic filters work better with simpler demosaicing methods such as bilinear demosaicing, which does not try to find a better direction for interpolation.

We examine the influence of the threshold in Table 3.2e. As fewer windows get detected, a higher threshold systematically means that the recall is lower. However, a higher threshold does not necessarily improve the precision; the best precision (and best score overall) is achieved with a 0.1 threshold, and higher thresholds yield a lower precision. This can be explained by the fact that the most confident detections often correspond to textured areas, where intermediate values are created by the texture more than the demosaicing, and are thus a source of false positives.

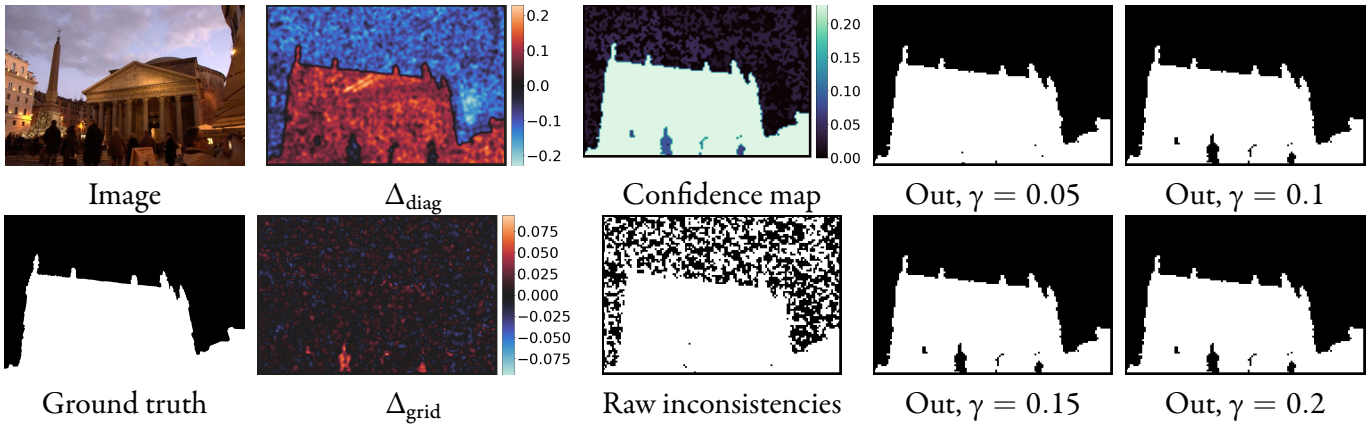
Finally, Table 3.2b shows the scores with different window sizes. While a window size of 32×32 yields the best results, this is inherently tied to the database in question. We saw earlier that increasing the window size would often lead to a better grid identification, but this also comes at the cost of missing small forgeries, and also failing to identify the borders of the bigger ones.

In Figure 3.13, we investigate the importance of the thresholding. Inconsistent false detections are usually not found in large connected regions of the same detected grid. As a consequence, even if some of those detections were to be significant, they would not cause a large number of false detections. On the other hand, regions detected because they are truly forged have a high chance of being actually forged. A confident result on one window of that region is thus enough to detect the whole region. Of course, this threshold is not fully automatic: it must still be set by the user, and will not filter out zones that are strongly detected for reasons other than a forgery, such as saturation or textured areas. The image in Figure 3.13b is interesting: as it was demosaiced with AAHD, its diagonal is not detected correctly (and no confident detection is thus made in Δ_{grid}). However, because the two regions do not share their diagonal, the inconsistency is still detected. The region demosaiced in $\begin{smallmatrix}\text{G}&\text{R}\\\text{B}&\text{G}\end{smallmatrix}$ is detected as being in the $\begin{smallmatrix}\cdot&\text{G}\\\text{G}&\cdot\end{smallmatrix}$ diagonal, whereas the region demosaiced in $\begin{smallmatrix}\text{B}&\text{G}\\\text{G}&\text{R}\end{smallmatrix}$ is detected as $\begin{smallmatrix}\text{G}&\cdot\\\cdot&\text{G}\end{smallmatrix}$: even though those predictions are wrong, they are still inconsistent with one another.

We further explore this phenomenon in Figure 3.14. As long as the two regions of an AAHD-demosaiced image do not share the same diagonal, they can still be



(a) Image ro6888d38t of the CFA Grid dataset with endomask, demosaiced with the VNG algorithm. The authentic region was demosaiced in the $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$ pattern, the forged region in the $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$ pattern.



(b) Image r1594b4b3t of the CFA Grid dataset with exomask, demosaiced with the AAHD algorithm. The authentic region was demosaiced in the $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$ pattern, the forged region in the $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ pattern.

Figure 3.13: Influence of the threshold on the removal of inconsistent false positives. Less-confident regions of the correct detection are kept, as long as they are connected to a more confident window, whereas inconsistent false detections are not connected to a large region: even if a few of them are confidently detected, most will be filtered out. Raw inconsistencies highlights every block whose detected pattern is inconsistent with the main image, regardless of significance.

detected. However, if the diagonal is wrong, then the algorithm will not look at the correct difference map to select the full pattern. As a consequence, it will be unable to make a consistent decision, and will thus not detect two different patterns that share the same diagonal. Nevertheless, the unused Δ maps still shows clear traces of the forgery. If one is suspicious that the diagonal is reversed (for instance because it is detected confidently over the whole image, whereas the full pattern is inconsistent), inverting the results of the diagonal, or visually examining the unused Δ map, can thus still reveal the forgery.

In the case of an image whose forged regions come from different algorithms, one of which being AAHD or DHT, the diagonal inversion means that if the two regions share the same diagonal (or even the very same pattern), the forgery could be detected, albeit for the wrong reason. However, this also means that if the two regions do not share the same pattern – an inconsistency which is usually easier to find, as seen in Table 3.2c –, the forgery then becomes invisible. This problem can be seen in Figure 3.15.

3.4 Conclusion

The presented method counts the number of locally intermediate values corresponding to each potential demosaicing pattern. The correct pattern, which contains all the originally-sampled pixels, is expected to have fewer intermediate values than the other patterns. The method starts by detecting which diagonal is used with the green channel, then uses the red and blue channels to distinguish between the two patterns sharing the detected diagonal. We proposed a different way of computing intermediate values, which yields slightly better results by exploiting the fact that demosaicing algorithms often interpolate on only one direction.

Doing this both on the full image and in local windows, we are able to detect locally inconsistent windows. The difference between the two compared counts of intermediate values in a window is normalized to serve as confidence for the detection of this window. Those windows are then grouped into connected components of windows that share the same detected grid, and the confidence of a component is set as the maximal confidence of its windows.

This confidence map enables easy visualization of the detections and their significance, and thresholding it filters out most of the incoherent false detections, while keeping the consistent detections. However, the threshold must be set manually.

The main limitation of this method, and of CFA forgery detection in general, remain its low robustness to JPEG compression. With this method, detections are already more difficult at a JPEG quality factor of 100, and become impossible at quality 95, effectively limiting its applicability range. We also saw that counter-forensic attacks based on the addition of white noise are quite effective. The classic median filter attack is instead relatively inefficient as it ends up inverting the interpolation masks, and the diagonal can thus still be detected.

Furthermore, of the 7 tested demosaicing algorithm, 2 cause an inversion of the detected diagonal. In the green channel, AAHD and DHT actually cause more intermediate values to appear on the sampled pixels.

While the method is usable, these issues make it still unsatisfactory. It is possible to detect local inconsistencies even within wrong detections, however erroneous pattern detections cannot be avoided against all demosaicing algorithms without leveraging inter-channel transfers. Furthermore, we wish to improve the robustness to JPEG compression. This leads us to abandon the idea of direct detection of interpolated and sampled pixels. Instead, the next two chapters will try to reverse-engineer the demosaicing algorithm itself, so as to estimate the potential mosaic in which a demosaicing algorithm was effectively used.

	MCC	IoU	F1
Isotropic, raw	0.518	0.490	0.573
Isotropic, $\gamma = 0.1$	0.592	0.567	0.622
Bidirectional, raw	0.543	0.515	0.595
Bidirectional, $\gamma = 0.1$	**0.610**	**0.584**	**0.642**

(a) Results with isotropic and bidirectional interme-diate values, raw and with connected confidence bothsizes. The method was used with bi-continuous and thresholded at $\gamma = 0.2$, on 32×32directional filters, continuous norm-windows.

Window size	MCC
16	0.592
32	**0.610**
64	0.412
128	0.163

(b) Results with different window alisation at $\gamma = 0.2$, 32×32 win-dows.

	All images	Same diagonal	Different diagonal
Main grid	0.476	**0.503**	0.461
Diagonal	0.429	0.000	0.671
Combined	**0.610**	0.501	**0.673**

(c) Influence of using only main grid inconsistencies, diagonal inconsistencies and their combination (pointwise maximum of the two detection maps), on the full database, and when only looking at images whose authentic and forged parts share/do not share the same diagonal. The diagonal is shared in 364 out of the 1000 images of the dataset. By combining the two maps, we can keep the best of both detections.

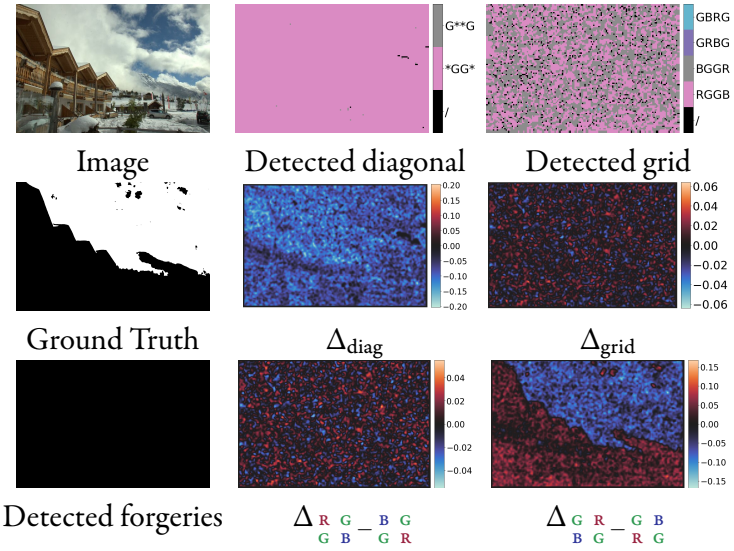
Algorithm	All	AAHD	AHD	DCB	DHT	Bilinear	PPG	VNG
#Images	1000	126	138	133	155	154	147	147
Isotropic	0.592	0.372	0.696	**0.786**	0.305	**0.742**	0.590	**0.657**
Bidirectional	**0.610**	**0.375**	**0.755**	0.763	**0.350**	0.649	**0.766**	0.613

(d) Results of the presented method depending on how the image was demosaiced. The method is used with bidirectional filters, on 64×64 windows, with hysteresis threshold-ing and combining the main grid and diagonal inconsistencies. Even though the method finds the wrong diagonal with the AAHD and DHT algorithms, it is consistent in doing so, and can thus still detect some forgeries, though not as well as against other demosaicing algorithms.

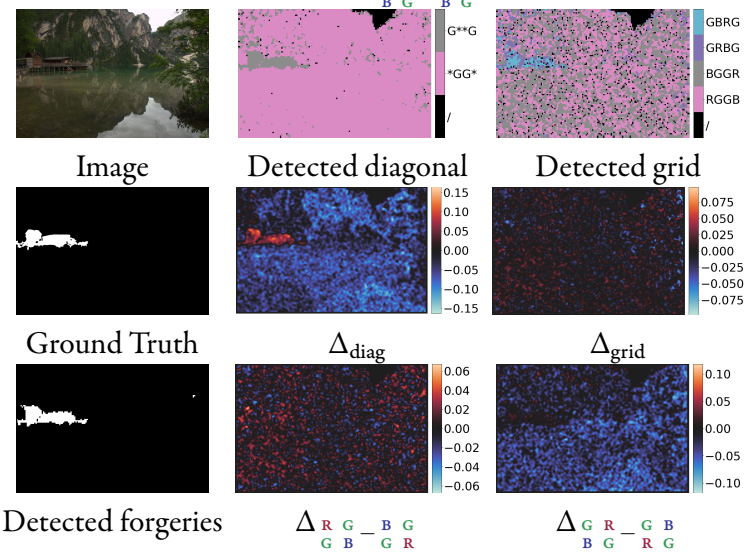
	MCC	IoU	F1	Precision	Recall
$\gamma = 0.05$	0.543	0.518	0.590	0.570	**0.733**
$\gamma = 0.1$	**0.610**	**0.584**	**0.642**	**0.670**	0.650
$\gamma = 0.15$	0.531	0.513	0.558	0.600	0.535
$\gamma = 0.2$	0.382	0.371	0.400	0.433	0.382

(e) Results with different metrics, raw and with confidence thresholding. The method was used with bidirectional filters, on 64×64 windows and combining the main and diagonal inconsistencies. Even though thresholding slightly lowers the recall, its gain in precision is much larger, thus yielding better MCC, IoU and F1 scores.

Table 3.2: Quantitative experiments on the Trace database of Chapter 1 (Non-Semantic Evaluation of Image Forensics Tools). Where parameters are not specified, these were used: bidirectional filters, continuous normalization at $\gamma = 0.2$, 32×32 windows, combined results of the full pattern and diagonal maps.



(a) Image r1d53fccat of the CFA Grid dataset, with endomask. Both regions are demosaiced with the AAHD algorithm, the authentic region in the $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$ pattern and the forged region in the $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$ pattern. Because the method wrongly detects the $\begin{smallmatrix}\cdot&G\\G&\cdot\end{smallmatrix}$ over the whole image, it only compares the two patterns sharing that diagonal, and $\Delta_{\text{grid}} = \Delta_{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}\_\begin{smallmatrix}B&G\\G&R\end{smallmatrix}}$ on almost all the image. As a consequence, no detection can be made on the grid level, and the forgery is not detected. Nevertheless, it appears clearly on $\Delta_{\begin{smallmatrix}G&R\\B&G\end{smallmatrix}\_\begin{smallmatrix}G&R\\B&G\end{smallmatrix}}$, which is not used in the algorithm.



(b) Image r15919202t of the CFA Grid dataset, with endomask. Both regions are demosaiced with the AAHD algorithm, the authentic region in the $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$ pattern, the forged region in the $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ pattern. Although the method finds the wrong diagonal in both regions, it still finds that the two regions use a different diagonal.

Figure 3.14: On those two AAHD-demosaicked images, the method finds the wrong diagonal. On the second image, it can still detect that the two regions use a different diagonal, and detects the forgery. On the first image, however, the two regions share the same diagonal. Using the diagonal as a basis to detect the full pattern, the method is unable to make any consistent detection on the full pattern when the diagonal is wrong, and thus does not find the forgery even though it is clearly visible when the correct diagonal is used. This shows the limits of using the detected diagonal as a strict basis for the full detection.

(a) Input image (b) Ground Truth (c) Detected inconsistencies. (d) Normalized difference between the two diagonal patterns.

Figure 3.15: Image r040b3002t of the CFA Algorithm dataset, with exomask. The authentic region is demosaiced with the AAHD algorithm in the $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$ pattern, the forged region is demosaiced with the DCB algorithm in the $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ pattern. Because the method consistently finds the wrong diagonal on AAHD-demosaiced images, but detects the correct diagonal on DCB-demosaiced images, it believes that the two regions share the same diagonal, even though they do not.

Chapter 4

Linear Estimation of the Demosaicing Algorithm

Abstract

In the last two chapters, we explored direct approaches to distinguish sampled pixels, and in fine to reveal the correct CFA pattern. However, those approaches were unsatisfactory. While Chapter 3 (Intermediate Values Counting for CFA Pattern identification) provided good results, the simplicity of those direct approaches could not take into account the properties of each specific demosaicing algorithm, leading to erroneous results on images demosaiced by such algorithms. We now propose instead to follow a reverse-engineering approach so as to more accurately reflect the specificities of each image and demosaicing algorithm. We create a linear estimation of the demosaicing that would be associated with each of the four possible CFA patterns. The residuals from these models yield a local estimation of the CFA pattern in the image. An a contrario approach is then applied to find regions whose detected pattern significantly deviate from the rest of the image. We show that while a linear estimation can be sufficient to find the image's CFA pattern, the many nonlinearities of demosaicing, as well as the natural texture of images, make this global and linear model locally unreliable.

An interactive demo for this chapter is available at https://ipolcore.ipol.im/demo/clientApp/demo.html?id=77777000095.

4.1 Introduction

In a seminal paper on demosaicing analysis, Popescu and Farid [6] proposed to use a two-class model of an image's pixels, by separating those that are originally sampled by the camera and those that were interpolated. They propose to use expectation-maximization (EM) to jointly estimate the sampled/interpolated status of each pixel, and a linear estimation of the used demosaicing algorithm. Spectral analysis of the residual error from the linear model can then reveal a 2-periodicity in the image's Fourier transform. Forgeries can then be detected as a phase-shift or a loss of periodicity of the residual. This detection methodology has been henceforth adopted by many papers.

However, EM can emphasize originally-insignificant inconsistencies. Analysing the significance of detections is thus difficult in such a scheme. Instead of jointly

finding sampled pixels and estimating the interpolation of missing values, we propose to separately estimate the interpolation on each of the four possible patterns. We do not know *a priori* which of these positions is used in the image; as such we have no choice but to estimate the interpolation method for each of these assumptions. If the approximation is correct, the image obtained with the estimated method from the mosaiced image should be closest to the original when the good grid position assumption is made, since interpolated pixels are easier to estimate from sampled pixels than the contrary.

If the detected pattern is locally inconsistent with the rest of the image, for instance if a small region is detected in the $\begin{smallmatrix}R & G\\ G & B\end{smallmatrix}$ pattern while the rest of the image is detected on the $\begin{smallmatrix}G & R\\ B & G\end{smallmatrix}$ one, then the image can be suspected to be forged in that region. However, considering all regions with a false pattern as forged would lead to an astonishingly high number of false positives, since an estimator is not perfect and can make many mistakes, especially on small regions. A simple heatmap of confidence in each detection is not ideal either, as the absence of an automatic threshold makes the end decision more difficult.

In this context, it becomes particularly useful to control the rate at which false positives are found. The *a contrario* framework [83] has been specifically developed in image analysis for that purpose. It considers a detection as meaningful if the expected number of similar detections obtained "just by chance" is below a specified false alarm rate. Here we aim at very low false alarms rates, to filter out detections that are not significant enough. See Chapter 0.4 (Introduction) for a general overview of *a contrario* detection.

In Section 4.2, we detail the linear estimation of the demosaicing method to detect the CFA pattern. The automatic detection of forgeries is then discussed in Section 4.3.

4.2 Linear estimation of the demosaicing algorithm

Considering a color image I of shape $(X, Y, 3)$, where $I[x, y, c]$ contains the value of the pixel at location (x, y) in the c^{th} channel – using the convention that the first channel is red, the second green and the third blue – we assume we know the position of the CFA grid G, of the same shape and with $G(x, y, c)$ equal to 1 if the c^{th} channel is the one sampled at position (x, y), and 0 otherwise. We construct the mosaicked image M of shape (X, Y) as

$$M[x, y] = \sum_{c=0}^{2} I[x, y, c]\, G[x, y, c].$$

If the grid position is correct, it corresponds to the image that was sampled by the camera.

Estimating the demosaicing method means we are looking for a function f such that $\mathrm{f}(M) \approx I$. We shall approximate the demosaicing with linear filters as proposed by [6]. Our method is actually simpler because we shall avoid using an expectation maximization (EM) by estimating optimal filters for each of the hypothesized CFA configurations. Furthermore we get a better filter approximation by allowing them

to use all observed channels, as is actually done in most modern demosaicing algorithms.

As we estimate a demosaicing method individually for each possible hypothesis on the grid position, we can directly find a least square optimal filter without recurring to EM. We can therefore directly find for each image the linear filter α of size $(2N+1, 2N+1)$ that minimizes

$$\sum_{c=0}^{2} \sum_{x,y} (I[x,y,c] - ((I \cdot G)[c] * \alpha)[x,y])^2$$

where $*$ represents convolution and $\cdot$ the entry-wise product. In order to address the second problem and take inter-channel correlation into account, we have to use the mosaiced image M instead of a single channel of the original image masked on the interpolated pixels and minimize

$$\sum_{c=0}^{2} \sum_{x,y} (I[x,y,c] - (M * \alpha)[x,y])^2 .$$

However, in order to solve the last problem and get an acceptable approximation, no less than 8 linear filters are to be estimated – one per position on the CFA grid/missing color channel pair:

- $\alpha_{R\to g}$ and $\alpha_{R\to b}$, which interpolate respectively the green and blue channels at the locations where the sampled pixel is red,
- $\alpha_{GR\to r}$ and $\alpha_{GR\to b}$, which interpolate respectively the red and blue channels at the locations where the sampled pixel is green on a line of reds,
- $\alpha_{B\to r}$ and $\alpha_{B\to g}$, which interpolate respectively the red and green channels at the locations where the sampled pixel is blue, and
- $\alpha_{GB\to r}$ and $\alpha_{GB\to b}$, which interpolate respectively the red and blue channels at the locations where the sampled pixel is green on a line of blues.

We estimate for instance $\alpha_{R\to g}$, the filter that interpolates the green pixel where the sampled pixel is red, by solving the linear systems that arise when we set to 0 the gradients of the L_2 error of the residual on pixels where the sampled channel is red. This yields the linear system $A\alpha_{R\to g} = b$, with

$$A[u+Nv, s+Nt] = \sum_{x,y} G[x,y,0]M[x+u, y+v]M[x+s, y+t]$$

and

$$b[u+Nv] = \sum_{x,y} G[x,y,0]M[x+u, y+v]I[x,y,1].$$

The other 7 filters can be estimated in a similar manner.

For each grid position $P \in \left\{ \begin{smallmatrix} R & G \\ G & B \end{smallmatrix}, \begin{smallmatrix} B & G \\ G & R \end{smallmatrix}, \begin{smallmatrix} G & R \\ B & G \end{smallmatrix}, \begin{smallmatrix} G & B \\ R & G \end{smallmatrix} \right\}$, let $\tilde{I}_P$ be the image interpolated from the corresponding mosaicked image with the filters estimated above.

We start by detecting the diagonal of the pattern, i.e. we detect whether the pattern uses the $\begin{smallmatrix}\cdot & \mathrm{G}\\ \mathrm{G} & \cdot\end{smallmatrix}$ or the $\begin{smallmatrix}\mathrm{G} & \cdot\\ \cdot & \mathrm{G}\end{smallmatrix}$ diagonal. This estimation is more easily done in the green channel, we thus define:

$$\tilde{D} = \arg\min_{P} \sum_{x,y} \|\tilde{I}_P[x,y,1] - I[x,y,1]\|^2, \tag{4.1}$$

the estimated diagonal $D^\star$ is then $\begin{smallmatrix}\cdot & \mathrm{G}\\ \mathrm{G} & \cdot\end{smallmatrix}$ if $\tilde{D} \in \left\{\begin{smallmatrix}\mathrm{R} & \mathrm{G}\\ \mathrm{G} & \mathrm{B}\end{smallmatrix}, \begin{smallmatrix}\mathrm{B} & \mathrm{G}\\ \mathrm{G} & \mathrm{R}\end{smallmatrix}\right\}$ and $\begin{smallmatrix}\mathrm{G} & \cdot\\ \cdot & \mathrm{G}\end{smallmatrix}$ otherwise. Note that in most cases, the green residuals in the $\begin{smallmatrix}\mathrm{R} & \mathrm{G}\\ \mathrm{G} & \mathrm{B}\end{smallmatrix}$ and $\begin{smallmatrix}\mathrm{B} & \mathrm{G}\\ \mathrm{G} & \mathrm{R}\end{smallmatrix}$ patterns will be very close, and so will be those in the $\begin{smallmatrix}\mathrm{G} & \mathrm{R}\\ \mathrm{B} & \mathrm{G}\end{smallmatrix}$ and $\begin{smallmatrix}\mathrm{G} & \mathrm{B}\\ \mathrm{R} & \mathrm{G}\end{smallmatrix}$ patterns. Indeed, most demosaicing algorithms make little to no use of the red and blue values to interpolate the green channel, although they make extensive use of all channels to interpolate the red and blue ones. See Chapter 0.4 (Introduction) for more details.

Now that the diagonal is known, we only have to distinguish between the two positions sharing that diagonal. This difference is computed in the three channels, although as explained above the difference will be mostly located in the red and blue channels:

$$P^\star = \begin{cases} \arg\min_{P \in \left\{\begin{smallmatrix}\mathrm{R} & \mathrm{G}\\ \mathrm{G} & \mathrm{B}\end{smallmatrix}, \begin{smallmatrix}\mathrm{B} & \mathrm{G}\\ \mathrm{G} & \mathrm{R}\end{smallmatrix}\right\}} \sum_{x,y,c} \|\tilde{I}_P[x,y,c] - I[x,y,c]\| & \text{if } D^\star = \begin{smallmatrix}\cdot & \mathrm{G}\\ \mathrm{G} & \cdot\end{smallmatrix} \\ \arg\min_{P \in \left\{\begin{smallmatrix}\mathrm{G} & \mathrm{R}\\ \mathrm{B} & \mathrm{G}\end{smallmatrix}, \begin{smallmatrix}\mathrm{G} & \mathrm{B}\\ \mathrm{R} & \mathrm{G}\end{smallmatrix}\right\}} \sum_{x,y,c} \|\tilde{I}_P[x,y,c] - I[x,y,c]\| & \text{if } D^\star = \begin{smallmatrix}\mathrm{G} & \cdot\\ \cdot & \mathrm{G}\end{smallmatrix} \end{cases}. \tag{4.2}$$

This gives us the estimated pattern for the global image. The pattern can also be estimated locally, by restricting the summation to a small region of the image. The small region in which the estimation is done must have even dimensions, for the comparison between patterns to be fair. Indeed, as the residual is zero on sampled pixels and channels, a comparison between two patterns requires the same number of residual pixels to be zero on each channel, at approximately the same location. Furthermore, it can happen that the residual is equal for two patterns. While this is almost impossible in the full image, it is more likely to happen in very small regions, especially if that region is flat in at least one colour component. If the green residuals are equal for the diagonal pattern estimation, then the diagonal pattern is estimated using all three channels. If the new residuals are still equal, then no pattern is considered as detected in the region. Similarly, if the two patterns sharing the detected diagonal yield equal residuals, no pattern is detected – although the diagonal itself is detected.

4.3 Automatic forgery detection

In the previous section, we detailed how the pattern could be detected both locally and globally. In addition to the global estimation, we thus perform local estimation of the diagonal and pattern in all the 2×2 blocks of the image, without overlap.

In the context of image forgery detection, it is often useful to provide an automatic decision on the authenticity of an image, so that the result does not require an expert interpretation. Furthermore, filtering out statistically insignificant results enables one to be more confident in the output of an algorithm. *A contrario* [46], [83] analysis is particularly suited to this context, and has already been successfully

applied to other forgery detection methods [19], [25]. Based on Gestalt theory, this detection theory performs automatic thresholding of the data by controlling an upper bound of the number of false alarms (NFA) one might expect. Given a background hypothesis H_0 and a significance function S that represents the significance attributed to an observation, we can compute the p-value of an observation x, that is, the probability of a random observation to be at least as significant as x:

$$p(x) = \mathbb{P}_{y\sim\mathcal{H}_0}(S(y) \geq S(x)). \tag{4.3}$$

While the p-value provides an idea of the importance of a detection, it does not account for the number of observations that are being made; in presence of a large quantity of observations one can expect spurious detections even with a small threshold of the p-value. The *a contrario* framework instead proposes to put a threshold on the NFA, which is obtained by multiplying the p-value by the number of tests, real or potential. For example, when computing the NFA of rectangles on an image, the number of tests is the total number of possible rectangles in the image – even if not all rectangles are tested. Associating an NFA to each detection and keeping only detections whose score is below the threshold ε amounts to fix an upper bound ε on the expected number of false alarms in the whole image:

$$\mathbb{E}_{x\sim\mathcal{H}_0}\left(|\{x|\mathrm{NFA}(x) < \varepsilon\}|\right) \leq \varepsilon. \tag{4.4}$$

The NFA of a detection belongs in $]0, +\infty[$, with scores closer to 0 corresponding to more significant detections. An NFA of 10^{-3}, for instance, means that under the background hypothesis $\mathcal{H}_0$, the expected number of detections at least as significant is at most 10^{-3}. So we expect at most one false detection every 1000 images.

Using this framework, we want to find suspicious regions in an image. We could limit ourselves to find regions that present a significant grid that is different from the grid of the global image. Yet, this would not enable us to detect areas with multiple small patches of different grids (as is frequently the case on inpainted images); nor would we see the localised absence of demosaicing.

Instead, we propose to detect regions where the detection is significantly erroneous, ie. where the network makes more mistakes than in the rest of the image. We apply the method separately on the detected diagonals and on the detected patterns, let E_d (resp. E_p) be a binary map which equals 1 for each block whose detected diagonal (resp. pattern) is different from D_g (resp. P_g). This is a map of the "wrong" blocks. The computation of those maps is described in Algorithm 6.

For the rest of the subsection, E represents either E_d or E_p. The empirical probability of any block on the image being wrong is denoted by p_0, and is computed as the mean of E. We want to find regions in which the error density is significantly higher than p_0. Note that blocks whose diagonal/pattern is undecided (due to equal residuals in both patterns) are ignored, they are not used in any counts.

Let us assume that, in a given rectangle, k out of the n blocks contained in the rectangle are incorrect. Under the background hypothesis that the probability of error is p_0, and assuming that the blocks are independent, the probability of having at least k wrong blocks in the area is the survival function of the binomial distribution $\mathrm{Binom}_{sf}(k, n, p_0)$. Yet a first obstacle to this simple strategy arises, as the grid

Algorithm 6: Error map computation

```
function compute_errormap(P)
    Input P: patterns or diagonal of each block, size (X, Y).
    Output E: Error map of P, same size.
    Output P_g: Global detected pattern or diagonal.
    # Global pattern
    P_g ≔ mode(P)
    E ≔ 0_{X,Y}
    for x from 0 to X and y from 0 to Y do
        if P_{x,y} ≠ P_g then
            E_{x,y} ≔ 1
    return E, P_g
```

values of different blocks are not independent. To achieve independence, we simulate down-sampling and divide k and n by d^2, where d is the radius of the linear filters used in the linear estimation. To account for the fact that in the binomial integers are then replaced by floating values, we use the Beta distribution to interpolate the binomial. The probability of having at least k wrong blocks in this area is thus evaluated by

$$p_{k,n,p_0} = I_{p_0}\left(\frac{k}{d^2}+1, \frac{n-k}{d^2}\right),$$

where I_x is the regularized incomplete Beta function. Under the *a contrario* framework, the number of tests is the possible number of rectangles in the image, that is, the number of blocks squared, multiplied by a factor 2 since we work separately on the patterns and diagonal. In consequence, the number of false alarms associated with the detection is defined by

$$\mathrm{NFA}_{k,n,p_0} = 2n_{\text{blocks}}^2 I_{p_0}\left(\frac{k}{d^2}-1, \frac{n-k}{d^2}\right). \tag{4.5}$$

The computation of the NFA is described in Algorithm 7.

Ideally, to detect forgeries in an image, we would compute the NFA of all the rectangles in the image. The score of a pixel would be the minimal score among the rectangles containing that pixel, and the score of an image would be that of the most significant rectangle. However, this poses several issues:

- The number of rectangles scales quadratically with the number of pixels in an image. Hence, checking all possible rectangles is not possible;
- even if a forgery is detected, some rectangles bigger than the forgery itself may still be significant, and the detection will therefore be too large;
- conversely, if part of a forgery is detected, we should detect nearby parts of the same forgery as well, even if they are not as significant as the detected part.

Algorithm 7: NFA computation

1 **function** `get_rectangle_nfa(`E`,` d`,` $x0$`,` $x1$`,` $y0$`,` $y1$`)`

  Input E: 1 if the block is erroneous, 0 if it is correct. Size (X, Y).

  Input d: Downsampling coefficient, given by the radius of the linear estimation filters.

  Input $x0, x1, y0, y1$: Coordinates of the rectangle whose NFA to compute

  Output ε: NFA of the rectangle

2 $p_0 \coloneqq \dfrac{1}{X \cdot Y} \displaystyle\sum_{x=0}^{X} \sum_{y=0}^{Y} E_{xy}$

3 $n_{\text{tests}} \coloneqq 2 * (X \cdot Y)^2$

  `# Number of erroneous blocks in the area.`

4 $k \coloneqq \displaystyle\sum_{x=x_0}^{x_1} \sum_{y_0}^{y_1} E_{xy}$

  `# Total number of blocks in the area.`

5 $n \coloneqq (x_1 - x_0) \cdot (y_1 - y_0)$

  `#` I_x `is the regularized incomplete Beta function.`

6 $NFA \coloneqq n_{\text{tests}} \cdot I_{p0}\left(\dfrac{k}{d^2} - 1, \dfrac{n-k}{d^2}\right)$

7 **return** NFA

As a consequence, we propose to first detect and separate all potential forgeries, and then to decide on their significance.

Still separately on the diagonal and full patterns, we use the map E of 2×2 blocks whose diagonal/pattern is erroneous. We apply a morphological closing to this map with a disk of size 2 to connect inconsistent blocks, and segment the resulting map into connected components. Components where the global pattern (respectively diagonal) represent more than 25% (respectively 50%) of the blocks are immediately rejected and not tested for forgeries.

Each of the remaining components is tested to determine whether it is a forgery. On each component, we test all the rectangles contained within the bounding box of the component, with a step of 16 pixels. The selected striding represents a compromise between precision and computation time, as a lower stride means more rectangles need to be checked.

Finally, we keep the NFA of the most significant rectangle. We set the score of the whole component to this NFA, thus solving the final two issues addressed above: only blocks that were in the component are given this NFA, and blocks out of a significant rectangle but still in the component are kept.

Forgery detection is performed separately on the full pattern and on diagonals, then the detected forgeries are merged. The final NFA map is the pointwise minimum of score maps of the patterns and diagonals NFA.

The NFA of a region is an upper bound on the expected number of regions that would be falsely detected as forged under the background hypothesis. We set the

Algorithm 8: Forgery detection

```
function get_forgery_mask(E, d, s)
    Input P: Patterns or diagonals of each block, size (X, Y).
    Input p_b: 0.5 if P represents diagonal, 0.25 if it represents patterns.
    Input d: Downsampling coefficient, given here by the size of the filters.
    Input s: Stride at which to search for rectangles. Here, s = 16.
    Output 𝒟: Forgery mask, each pixel represents the NFA detection
              score of its corresponding 2 × 2 block.
    # Initialize detections at infinity.
    𝒟 ≔ +∞
    E, P_g ≔ compute_errormap(P)
    # Morphological closing with a disk.
    E_c ≔ morphological_closing(E, disk_2)
    # Segment into connected erroneous regions
    labels ≔ label_connected(E)
    for label from 0 to max(labels) do
        # Get the mask
        ℳ = labels_{x,y} = label
        # Do not proceed if the main pattern is already above the base
          probability.
        if (1/|E_c|) Σ_{x∈E_c} 1_{E_C}(x) > 1 − p_b then
            ε ≔ 0 for ℛ rectangle within the bounding box of ℳ at step
              s do
                ε_ℛ ≔ get_rectangle_nfa(E, d, ℛ)
                ε ≔ min(ε, ε_ℛ)
            ℳ ≔ morphological_closing(ℳ, disk_8)
            # Pointwise minimum to update detections on the mask
            𝒟_|ℳ ≔ min(𝒟_|ℳ, ε)
    return 𝒟
```

threshold to $\varepsilon = 10^{-3}$, and the final, binary map keeps pixels whose NFA is below this threshold. Under the background hypothesis, the false detection rate therefore is expected to be below one for 1000 images. Of course, false detections can still be made when an authentic region is significantly more erroneous than the rest of the image, for reasons potentially linked to the image structure (for example the presence of textured areas) or post-processing such as an image resampling which can modify the CFA traces. However, this enables us to only select regions which are significantly more erroneous than the rest of the image, regardless of the reason, and provides us with a mathematically rigorous threshold providing automatic detection.

The same scheme will be reused in a similar manner in Chapter 7 (Internal Learning to Improve Adaptability).

Demosaicing Algorithm \ Filter size	3×3	5×5	7×7
BIL	16/16	16/16	16/16
HA	9/16	9/16	13/16
GBTF	15/16	14/16	16/16
CS	16/16	16/16	16/16
LMMSE	15/16	15/16	16/16
AICC	15/16	16/16	16/16
SSDD	15/16	7/16	9/16
AP	14/16	16/16	16/16
RI	16/16	16/16	16/16
MLRI	16/16	16/16	16/16
ARI	16/16	16/16	16/16
CDM	16/16	16/16	16/16

Table 4.1: This table represents how many of the 16 images were detected in the correct grid, depending on the used filter size of the method and the original demosaicing algorithm of the image. At the image level, the correct pattern is almost always detected against most demosaicing algorithms. When the image is demosaiced with Hamilton-Adams (HA), a large filter size is necessary to obtain decent results. With SSDD, however, the results are almost perfect with a small 3×3 filter, but much worse with larger filter sizes.

4.4 Experiments

Estimation of the correct pattern

We took the 16 images from the noise-free images dataset and demosaiced them in the $\begin{smallmatrix} R & G \\ G & B \end{smallmatrix}$ pattern with several algorithms. We then analysed the resulting images with our algorithm to check whether the correct pattern was detected, i.e. if the residual was lowest on the $\begin{smallmatrix} R & G \\ G & B \end{smallmatrix}$ pattern.

The image-level results are shown in Table 4.1. With most algorithms, the correct pattern was identified in almost all images. We can see that a larger filter size (5×5 or 7×7) usually improves the results, and is even necessary against Hamilton-Adams-demosaiced images. Surprisingly, however, SSDD-demosaiced images can only be analysed correctly with a small 3×3 filter; results sharply decrease when the filter is larger.

Next, we performed the same analysis in 32×32 blocks. The linear estimation was still performed image-wise, then the residuals were aggregated in 16×16 blocks (which correspond to 32×32 pixels in the original image, since each pixel in the residual represents 2×2 pixels in the original image). We then looked at the number of blocks in which the residual is lowest in the correct pattern. Results are shown in Table 4.2. The method does much better than a random baseline, however it makes mistakes on up to half the blocks depending on the algorithm and filter size. This

Demosaicing Algorithm \ Filter size	3×3	5×5	7×7
BIL	1.000	1.000	1.000
HA	0.403	0.654	0.661
GBTF	0.768	0.654	0.877
CS	0.755	0.749	0.873
LMMSE	0.535	0.634	0.834
AICC	0.504	0.686	0.746
SSDD	0.696	0.429	0.444
AP	0.490	0.784	0.861
RI	0.610	0.930	0.925
MLRI	0.647	0.929	0.929
ARI	0.610	0.841	0.866
CDM	0.782	0.832	0.890

Table 4.2: This table represents the proportion of 32×32 blocks whose pattern was detected correctly, depending on the filter size and the original demosaicing algorithms. The random baseline for this detection would be a score of 0.25. While the method does much better than the baseline, the high number of errors makes it impossible to confidently declare whether a single block is forged.

Camera \ Filter size	3×3	5×5	7×7
Canon 60D	0.020	0.017	0.017
Nikon D7000	0.143	0.147	0.172
Nikon D90	0.069	0.114	0.168
Sony α57	0.293	0.398	0.409
All	0.131	0.169	0.191

Table 4.3: Results of the method on the Korus [42], [43] datasets, using the Matthews Correlation Coefficient (MCC). The baseline of a random method is 0. The performance of the method strongly varies depending on the camera and its underlying demosaicing algorithm.

means that to make a confident detection, a large size is be needed.

Detection of forgeries

We applied the full method to the Korus [42], [43] dataset. As can be seen in Table **tab:korus**, the performance of the method depends a lot on the original demosaicing. On the Sony α57 camera, the bilinear demosaicing led to a very good score. On the two Nikon cameras, detections were more difficult to make, but decent res-

ults were still achieved. On the other hand, the score on the Canon 60D camera correspond to that of a random method. This is due to the fact that images from this camera do not seem to present any traces of demosaicing on this dataset, as will be evidenced by all the methods presented in this thesis as well as all tested SOTA demosaicing detection methods.

Overall, a larger filter size enables the method to make much better detections, even on the bilinear-demosaiced images of the Sony α57 camera. Indeed, even though bilinear demosaicing can be reproduced with a 3×3 convolution, the processing steps happening after demosaicing might slightly expand the footprint of the demosaicing.

Visual results can be seen in Figure 4.1. On images from the Nikon and Sony cameras, the forgeries are usually well-located. The detected forgery masks are often larger than the actual forgeries. This is due in part to the method used for NFA thresholding. Although this affects scoring, it is actually irrelevant to the detection itself, for as long as the forgery is detected, visual analysis of the detected patterns helps make the localization easy. However, when the demosaicing traces are harder to analyse, such as on the Nikon images, or when there are no demosaicing traces, such as on the Canon images, false detections can happen. The inconsistencies are already present in the grid detections; they can be caused by natural textures in the image or by traces which strongly deviate from the linear estimation.

4.5 Discussion

In this chapter, we constructed a method allowing us to correctly approximate demosaicing methods with several linear filters. An *a contrario* approach was then applied to detect regions where the grid detection is significantly wrong. Those regions are likely to be forged.

Linear estimation of the demosaicing algorithm seems to be enough to detect the original CFA pattern in most cases at the image level. However, this model is invalidated in regions where the original demosaicing is strongly non-linear, or where the image features naturally high-frequency-rich textures. This makes it difficult to use the method to find forgeries, as the detection can make localized mistakes, sometimes significant enough to be considered as forgeries after NFA thresholding.

Nevertheless, the possibility of estimating the demosaicing in different positions remains promising. The next chapter will explore this idea further, by replacing the linear estimation with the use of a collection of existing demosaicing algorithms. This will help us apprehend more precisely the common inter-channels transfers of demosaicing, as well as its non-linearities, using assumptions that stem from common demosaicing algorithms.

The proposed *a contrario* approach, on the other hand, seems satisfactory. The localization masks are not perfect and may require visual analysis of the block votes to precisely localize algorithms; while this can negatively impacts the scoring of the underlying method, it is of little practical importance as long as the proposed approach enables automatic detection of forged images while limiting the number of tests to perform, and thus the computational cost. This approach being good enough, we will not further extend work on *a contrario* analysis. The approach we

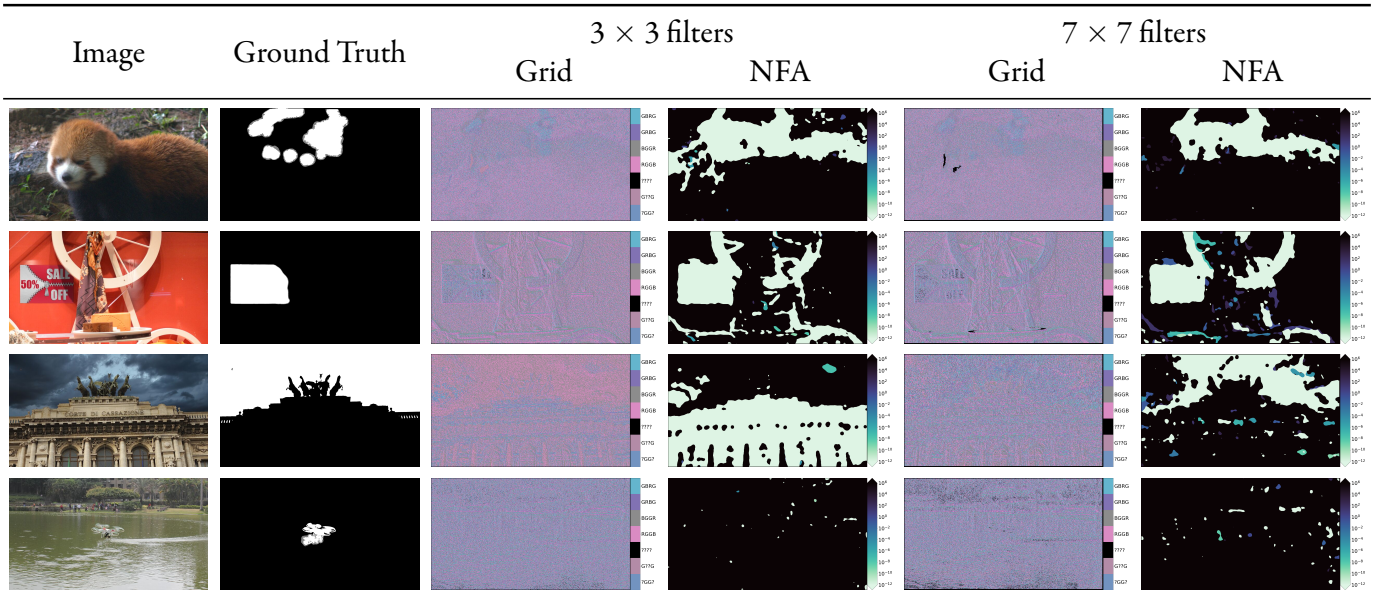


Figure 4.1: Results of the method on Korus images, from top to bottom from the Sony α57, Nikon D7000, Nikon D90, Canon 60D cameras. On the first image, the forgery is accurately detected. Even though the detected mask is too large, a visual analysis of the detected grid enables precise localization of the forgery. On the second image, the forgery is detected, but a second region of the image is mistakenly detected. On the third image, the forgery is inverted when using the smaller filter. Although this negatively affects the score, this is not actually a problem for the detection; as the method still shows the two regions are inconsistent with one another. Finally, on the fourth image, no traces of demosaicing are present. Some regions are incorrectly marked as inconsistent. Those inconsistencies are already present in the detected grids: those defects come from the linear estimation itself, not from the NFA thresholding.

followed here will be reused with our final method in Chapter 7 (Internal Learning to Improve Adaptability). Meanwhile, the next two chapters will focus on the identification of the CFA mosaic itself, with less focus on the subsequent analysis for forgery detection.

Chapter 5

Demosaicing to Detect Demosaicing

Abstract

The previous chapter proposed a simple reverse-engineering of the demosaicing algorithm with linear models. This simple estimation yields inconsistent results; however this does not invalidate the reverse-engineering approach itself. To avoid unrealistic assumptions about demosaicing such as that of linearity, we can instead use a collection of existing demosaicing algorithms, and accept the more natural assumptions that stem from these. Indeed, we can perform demosaicing using each of the four possible mosaics. The correct mosaic is more likely to yield an image closer to the original one.

In this chapter, we explore the possibilities of so-called double demosaicing to identify the CFA pattern. Local selection of the most suitable demosaicing algorithm enables one to detect the CFA pattern even when the original demosaicing algorithm is unknown. This method naturally avoids the two main issues of the previous chapters, namely their inability to take into account the inter-channel transfers of demosaicing and the need to go beyond overly simplistic linear assumptions. We simulate the two demosaicing steps to analyse in which conditions double demosaicing can be used to detect the pattern. Furthermore, introducing JPEG compression between the two demosaicing enables us to study the limits of CFA pattern detection on compressed images.

5.1 Introduction

To detect the CFA pattern, Kirchner et al. proposed to apply bilinear demosaicing in the four possible patterns. The result closest to the original image was more likely to correspond to the correct pattern. With the advent of better and more varied demosaicing algorithms, this is no longer always the case. However, it is possible to use a collection of algorithms, instead of just one.

Let $\mathcal{M}^{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}} S$ be an image of size $(2X, 2Y)$, mosaiced in the $^{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}}$ pattern[1]. We demosaic it with an algorithm A, yielding an image $I \triangleq \mathcal{D}^A \mathcal{M}^{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}} S$ of size $(2X, 2Y, 3)$. The goal is to estimate the original mosaicing pattern of the image, here $^{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}}$.

[1] Note that in general, the original signal S is not fully observed, as only the pixel colours within the mosaic are sampled by the camera: only its mosaiced version $\mathcal{M}^{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}} S$ is observed.

Demosaicing interpolates missing colour values from the image, but it does not modify the already-known values. This means that for all x, y, the following equalities apply:

- $\left(\mathcal{D}^A \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} S\right)_{2x,2y,R} = \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} S_{2x,2y}$,
- $\left(\mathcal{D}^A \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} S\right)_{2x,2y+1,G} = \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} S_{2x,2y+1}$,
- $\left(\mathcal{D}^A \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} S\right)_{2+1x,2y,G} = \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} S_{2x+1,2y}$,
- $\left(\mathcal{D}^A \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} S\right)_{2x+1,2y+1,B} = \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} S_{2x+1,2y+1}$,

where $I_{2x,2y,R}$ represents the red value of an image I at the position $2x$, $2y$.

What then happens if we artificially recreate a mosaic on $\mathcal{D}^A \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} S$, then demosaic it again? Formally, the mosaic function is defined as

$$\left(\mathcal{M}^P S\right)_{x,y} \triangleq \begin{cases} I_{x,y,R} & (x,y) \equiv (\delta_x, \delta_y) \mod (2,2) \\ I_{x,y,B} & (x,y) \equiv (1-\delta_x, 1-\delta_y) \mod (2,2) \\ I_{x,y,G} & \text{otherwise} \end{cases} \tag{5.1}$$

Where the offsets δ_x, δ_y are defined by P:

P	δ_x	δ_y
$\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$	0	0
$\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$	1	1
$\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$	1	0
$\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$	0	1

Mosaicing $\mathcal{D}^A \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} S$ in its original $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ position yields the original mosaic:

$$\mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} \mathcal{D}^A \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} S = \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} S. \tag{5.2}$$

As a consequence, when performing the remosaicing-demosaicing operation with the original CFA, the image is unchanged:

$$\mathcal{D}^A \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} \mathcal{D}^A \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} S = \mathcal{D}^A \mathcal{M}^{\begin{smallmatrix}R&G\\G&B\end{smallmatrix}} S = I. \tag{5.3}$$

On the other hand, if using the wrong CFA pattern, the originally-sampled values are lost, and the final demosaicing will reconstruct a slightly different image.

Assuming the demosaicing algorithm A is known, it is thus easy to find the original CFA pattern of an image, by remosaicing and demosaicing it again on the four possible patterns, and looking for the pattern with which the image was unchanged.

Based on this, Kirchner and Böhme [94] proposed to find the CFA pattern of an image by demosaicing in the four patterns with bilinear interpolation, and looking at the pattern with the lowest residual. Inspired by their method, we propose to

extend it by using multiple demosaicing algorithms instead of just bilinear interpolation. Indeed, although the assumption of bilinear demosaicing was reasonable when the article was published, most commonly-used demosaicing methods nowadays are much more advanced than bilinear demosaicing.

However, two issues arise:

- The original demosaicing algorithm is very rarely known,
- many images are post-processed after demosaicing. Notably, JPEG compression tends to destroy the high frequencies, in which demosaicing artefacts lay.

In this chapter, we study the robustness of double demosaicing for CFA pattern identification under two scenarios:

- when the original demosaicing algorithm is unknown,
- when the image has been compressed after the first demosaicing.

5.2 Methodology

The noise-free test images dataset [33] contains 18 high-quality images that were downsampled to remove noise. As those images are downsampled, they do not contain traces from a prior demosaicing, making them ideal for our experiments. Let S denote any of those images

We take several publicly available demosaicing algorithms: bilinear demosaicing, contour stencils (CS) [75], Hamilton-Adams (HA) [55], Linear Minimum Mean-Square-Error Estimation (LMMSE) [58], Alternating Projections (AP) [56], self-similarity-driven demosaicing (SSDD) [57], CDM-CNN (CDM) [62], gradient-based threshold-free (GBTF), residual interpolation (RI) [60], minimized-Laplacian residual interpolation (MLRI) [74], and adaptive residual interpolation (ARI) [61]. See Chapter 0.4 (Introduction) for a more detailed description of those algorithms.

We start by mosaicing S in the $\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}$ position and demosaicing it with algorithm A, in other words we compute $\mathcal{D}^A\mathcal{M}^{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}}S$. The goal is to study whether we can detect that $\mathcal{D}^A\mathcal{M}^{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}}S$ was indeed originally mosaiced in the $\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}$ pattern. Note that the four patterns are the same relative to a shift of one row and/or column. We can thus, without loss of generality, use $\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}$ as the initial pattern for the whole study.

To study the detection of the pattern, the core idea is to recreate the mosaic in all four patterns, and demosaic it with algorithm B. We then check whether the residual is lowest in the original $\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}$ pattern, in other words we check for

$$\underset{P\in\left\{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix},\begin{smallmatrix}\mathrm{G}&\mathrm{R}\\\mathrm{B}&\mathrm{G}\end{smallmatrix},\begin{smallmatrix}\mathrm{G}&\mathrm{B}\\\mathrm{R}&\mathrm{G}\end{smallmatrix}\right\}}{\arg\min} \left\|\left(\mathcal{D}^B\mathcal{M}^P - \mathrm{Id}\right)\mathcal{D}^A\mathcal{M}^{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}}S\right\| \overset{?}{=} \begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}, \tag{5.4}$$

where $\|\cdot\|$ is the L^2 norm. In case of equality for the arg min, a pattern is selected at random. This might be the case locally, for instance, in flat regions, where all patterns yield the same output.

In practice, the demosaicing algorithm initially used during image construction is rarely known. We thus propose to check the compatibility of different pairs of algorithms. In other words, we check Equation 5.4 for different pairs of algorithms. Compatibility is not necessarily commutative; it can be that an algorithm B is good to analyse A-demosaicked images (ie. Equation 5.4 is usually true) while A would fail to analyse B-demosaicked images (ie. Equation 5.4 is usually wrong with A and B swapped).

To truly identify the original mosaic pattern, however, the algorithm used for analysis needs to be selected automatically. To this end, we propose to simply run the second demosaicing with all the available algorithms, then locally select the one with the lowest residual. Equation 5.4 thus becomes

$$\underset{P\in\left\{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix},\begin{smallmatrix}\mathrm{B}&\mathrm{G}\\\mathrm{G}&\mathrm{R}\end{smallmatrix},\begin{smallmatrix}\mathrm{B}&\mathrm{G}\\\mathrm{R}&\mathrm{G}\end{smallmatrix}\right\}}{\arg\min} \min_{B} \left\| \left(\mathcal{D}^{B}\mathcal{M}^{P} - \mathrm{Id}\right) \mathcal{D}^{A}\mathcal{M}^{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}} S \right\|_2 \overset{?}{=} \begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}. \tag{5.5}$$

Even when pattern selection is performed on the whole image, the algorithm selection is always done locally, ie. at each pattern and for each pixel we take the output of the algorithm with the smallest residual. Indeed, demosaicing algorithms change behaviour in various ways depending on the region to demosaic, especially depending on the gradient and its direction. Algorithm A may thus behave more closely to B in one region, but to C in another.

In the current formulation, since the original demosaicing algorithm A belongs to the list used to select algorithm B, the original algorithm and the correct pattern are most likely to be selected; which is the case where the initial algorithm, although unknown, is by chance in the list of analysis algorithms. Since near-perfect results could be expected in this case, it is more interesting to study what happens when the original algorithm is excluded from the list, Equation 5.5 becomes

$$\underset{P\in\left\{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix},\begin{smallmatrix}\mathrm{B}&\mathrm{G}\\\mathrm{G}&\mathrm{R}\end{smallmatrix},\begin{smallmatrix}\mathrm{B}&\mathrm{G}\\\mathrm{R}&\mathrm{G}\end{smallmatrix}\right\}}{\arg\min} \min_{B\neq A} \left\| \left(\mathcal{D}^{B}\mathcal{M}^{P} - \mathrm{Id}\right) \mathcal{D}^{A}\mathcal{M}^{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}} S \right\|_2 \overset{?}{=} \begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}. \tag{5.6}$$

The two selection methods are called *inclusive* and *exclusive*, depending on whether they include or exclude the initial algorithm from the list of analysis algorithms. Note that, while the difference between the two methods is crucial to our study, they are not *per se* different methods. In practice, they only reflect the contrast depending on whether the – unknown – algorithm used in the image belongs to the analysis method's model.

The major issue of demosaicing detection methods is their low robustness to JPEG compression. This robustness is easy to check in our study; by simply introducing a compression step after the first demosaicing. In other words, Equation 5.4 is replaced by

$$\underset{P\in\left\{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix},\begin{smallmatrix}\mathrm{B}&\mathrm{G}\\\mathrm{G}&\mathrm{R}\end{smallmatrix},\begin{smallmatrix}\mathrm{B}&\mathrm{G}\\\mathrm{R}&\mathrm{G}\end{smallmatrix}\right\}}{\arg\min} \left\| \left(\mathcal{D}^{B}\mathcal{M}^{P} - \mathrm{Id}\right) \mathcal{J}^{Q}\mathcal{D}^{A}\mathcal{M}^{\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}} S \right\| \overset{?}{=} \begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}, \tag{5.7}$$

where $\mathcal{J}^{Q}$ represents JPEG compression with quality factor Q. The same change can be done for the exclusive and inclusive methods in Equations 5.5 and 5.6. The

JPEG compression introduces a loss of information. When this step is introduced after the first demosaicing, Equation 5.3 no longer holds; as a consequence neither same-algorithm analysis nor the inclusive method is no longer guaranteed to find the best pattern, neither is even analysis with the same algorithm.

5.3 Experiments

We first perform experiments on 2×2 blocks. In Table 5.1, we demosaic the 16 images from the dataset in the $\begin{smallmatrix}\mathrm{R} & \mathrm{G}\\ \mathrm{G} & \mathrm{B}\end{smallmatrix}$ pattern, remosaic them in the four possible patterns and subsequently demosaic them again. For each pair of algorithms, as well as with the automatic inclusive and exclusive methods, we show the proportion of 2×2 blocks that were detected in the correct position. Expectedly, results are best when the same algorithm is used for original demosaicing and analysis (grayed-out results), or with the Auto Inclusive method, since the original algorithm is included in the list.

Outside of these trivial cases, we can note that some algorithms are more suited for analysis than others. For instance, LMMSE demosaicing is able to analyse most images, and yields scores that are significantly above the baseline for all algorithms except bilinear interpolation.

Bilinear interpolation analysis yields decent scores against most images, however the analysis of images demosaiced by bilinear interpolation is much harder, and most algorithms yield scores that are below or barely above the random baseline. The same can be said for the AP algorithm.

On the other end of the spectrum, the most advanced demosaicing algorithms, such as ARI and CDM, can be decently analysed by simpler methods, however they cannot reliably be used for analysis, with ARI in particular yielding below-baseline scores on most methods. This may be due to the fact that such advanced algorithm succeed in removing most demosaicing artefacts that other methods would create, thus blurring the difference between patterns.

Automatic selection (exclusive of the original algorithm) consistently improves results of single methods. In all seen cases, its score is comparable to – and often slightly better than – the score of the best single algorithm. It still fails at being strongly above the baseline when analysing images demosaiced with AP or bilinear interpolation, and would thus require large images to be able to reliably detect the pattern.

We now perform the same experiment, this time adding JPEG compression between the two demosaicings. Tables 5.2, 5.3, 5.4, 5.5 show results at JPEG compression levels of 95, 90, 85, and 80. Figure 5.1 summarizes the results at all compression levels.

Contrarily to the uncompressed case, it is now interesting to look at results where the same algorithm is used for original demosaicing and analysis. Even if the demosaicing algorithm is known, detecting the pattern is no longer trivial due to the compression between the two steps. Unsurprisingly, using the same algorithms consistently yields better results than using other algorithms. However, the score gap between same-algorithm detection and automatic detections (both inclusive and exclusive) gets lower as the JPEG compression is stronger. Precisely knowing which

A \ B	BIL	HA	GBTF	CS	LMMSE	AICC	SSDD	AP	RI	MLRI	ARI	CDM	**Exclusive**	**Inclusive**
BIL	0.98	0.22	0.20	0.25	0.17	0.13	**0.32**	0.17	0.25	0.18	0.20	0.28	0.31	**0.94**
HA	0.34	0.96	0.46	0.40	0.47	0.36	0.38	0.35	0.42	0.43	0.28	0.34	**0.51**	**0.91**
GBTF	0.33	0.44	0.95	0.44	**0.62**	0.42	0.40	0.37	0.36	0.41	0.25	0.31	0.56	**0.89**
CS	0.38	0.34	0.43	0.95	0.43	0.36	0.41	0.31	0.29	0.31	0.20	0.31	**0.45**	**0.88**
LMMSE	0.24	0.41	**0.57**	0.39	0.92	0.42	0.33	0.38	0.28	0.34	0.18	0.22	0.51	**0.85**
AICC	0.28	0.37	0.44	0.40	**0.48**	0.92	0.33	0.36	0.30	0.34	0.21	0.22	0.44	**0.85**
SSDD	0.38	0.33	0.36	0.40	0.32	0.27	0.97	0.23	0.31	0.29	0.22	0.31	**0.42**	**0.92**
AP	0.23	0.36	0.36	0.29	**0.39**	0.30	0.25	0.91	0.20	0.25	0.18	0.14	0.32	**0.83**
RI	0.39	0.47	0.48	0.43	0.42	0.36	0.39	0.29	0.96	0.64	0.46	0.48	**0.65**	**0.91**
MLRI	0.34	0.47	0.52	0.43	0.47	0.39	0.37	0.32	**0.63**	0.96	0.39	0.41	0.61	**0.90**
ARI	0.38	0.38	0.39	0.36	0.35	0.30	0.32	0.27	0.51	0.47	0.96	0.42	**0.53**	**0.92**
CDM	0.43	0.41	0.44	0.46	0.38	0.32	0.42	0.27	0.47	0.42	0.36	0.96	**0.53**	**0.91**

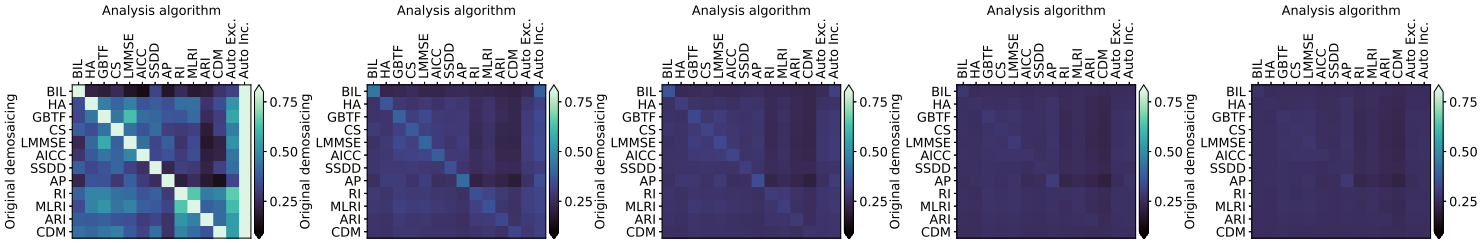
Table 5.1: Images are demosaiced with a first algorithm, then remosaiced in all four positions and demosaiced again with a second algorithm. For each pair of algorithm, we show the proportion of 2×2 blocks for which the second demosaicing had the lowest residual on the initial position, i.e. the proportion of blocks for which Equation 5.4 is verified, for different pairs of algorithms A and B. The last two columns, are obtained by taking the pattern with the lowest residual, across all analysing algorithms used. The inclusive method checks Equation 5.5 for various algorithms A, it can use residuals obtained with the same algorithm as the one used in the original image; the exclusive method instead checks Equation 5.6 and cannot use the same algorithm. As explained in Section 5.2, the pattern decision is trivial if the original algorithm is known. Entries are grayed out where both the original demosaicing and the analysing algorithms are the same. Each column represents the algorithm used for analysis, it shows how good this algorithm is to analyse images that were originally demosaiced with other algorithms. Each row represents the algorithm used in the original image, it shows how well images originally demosaiced by this algorithm can be analysed with other algorithms and the automatic method. In each row, results in bold represent the best analysing method, as well as where different the best results without knowledge of how the image was first demosaiced (i.e. without considering grayed-out entries) and the best results without using the original demosaicing algorithm in the models (i.e. without considering either grayed-out entries nor the Inclusive column). As a baseline, a random selection of the pattern on each block would be right 25% of the time.

demosaicing algorithm was originally used gives an important edge to detection on high-quality images. Against stronger compression, this is still useful but much less important.

Surprisingly, on compressed images, analysing bilinear-demosaiced images with the automatic method yields poor results, even if the bilinear algorithm itself is in the list of models (automatic inclusive). As can be seen in Figure 5.2, demosaicing artefacts from the initial bilinear demosaicing are partly removed by JPEG com-

A \ B	BIL	HA	GBTF	CS	LMMSE	AICC	SSDD	AP	RI	MLRI	ARI	CDM	**Exclusive**	**Inclusive**
BIL	0.44	0.23	0.24	**0.25**	0.23	0.23	0.24	0.25	0.23	0.23	0.22	0.22	0.24	**0.39**
HA	0.27	0.36	0.29	0.28	0.29	0.27	**0.29**	0.28	0.25	0.27	0.24	0.23	0.28	**0.31**
GBTF	0.28	0.28	0.39	0.31	**0.34**	0.29	0.29	0.29	0.25	0.28	0.25	0.23	0.30	0.34
CS	0.29	0.28	0.31	0.38	**0.32**	0.30	0.29	0.29	0.25	0.27	0.24	0.23	0.30	**0.33**
LMMSE	0.27	0.29	**0.34**	0.31	0.40	0.30	0.28	0.30	0.24	0.27	0.23	0.22	0.31	0.34
AICC	0.28	0.29	0.32	0.32	**0.32**	0.38	0.29	0.29	0.25	0.28	0.24	0.23	0.30	**0.34**
SSDD	0.28	0.29	0.29	**0.29**	0.29	0.28	0.35	0.28	0.25	0.27	0.25	0.24	0.29	**0.31**
AP	0.26	0.30	0.30	0.28	**0.30**	0.26	0.26	0.41	0.21	0.24	0.22	0.20	0.27	**0.35**
RI	0.29	0.28	0.30	0.29	0.29	0.28	0.28	0.27	0.35	**0.32**	0.29	0.27	0.31	**0.33**
MLRI	0.28	0.28	0.31	0.30	0.30	0.29	0.28	0.27	**0.31**	0.36	0.28	0.27	0.31	**0.33**
ARI	0.28	0.27	0.29	0.28	0.28	0.27	0.27	0.26	0.30	**0.30**	0.33	0.27	0.30	**0.31**
CDM	0.27	0.27	0.28	0.28	0.28	0.28	0.28	0.26	0.28	0.28	0.27	0.32	**0.29**	**0.31**

Table 5.2: Same results as in Table 5.1, but images are JPEG-compressed at quality 95 between the initial and analysing demosaicing operations. Contrarily to the uncompressed case, detection is no longer trivial when the demosaicing algorithm is known, since there is a loss of information between the two steps. Although inclusive results can be expected to be better than exclusive results, they rely on knowing the original algorithm, which is not always the case.



(a) Uncompressed (b) JPEG 95 (c) JPEG 90 (d) JPEG 85 (e) JPEG 80

Figure 5.1: Percentage of 2×2 blocks whose residual is lowest in the correct pattern depending on the original demosaicing and the algorithm used for analysis. On compressed data, `Auto Exc.` (resp.`Inclusive`) refers to selecting the pattern with the lowest residual across all analysis demosaicking algorithms, excluding (resp. including) the algorithm that was used in the original demosaicing process. On uncompressed images, only `Auto Exc.` results are presented.

pression. Performing a second bilinear demosaicing adds new artefacts, leading to a higher residual compared to the compressed image than what is obtained by using better demosaicing methods, that avoid creating new artefacts. As a consequence, bilinear demosaicing is not selected by the automatic method, leading to a wrong pattern identification since other demosaicing methods struggle to analyse bilinear-demosaiced images.

Both same-algorithm detection and the automatic methods yield scores that are consistently above the random baseline. However, the probability of finding the correct pattern gets closer to $\frac{1}{4}$ as the JPEG compression increases. As long as the

A \ B	BIL	HA	GBTF	CS	LMMSE	AICC	SSDD	AP	RI	MLRI	ARI	CDM	**Exclusive**	**Inclusive**
BIL	0.36	0.24	0.25	0.26	0.24	0.24	0.25	**0.26**	0.24	0.24	0.24	0.23	0.25	**0.31**
HA	0.27	0.30	0.27	0.27	0.27	0.26	**0.28**	0.27	0.25	0.26	0.24	0.23	0.26	0.27
GBTF	0.27	0.27	0.33	0.28	**0.30**	0.28	0.27	0.28	0.25	0.26	0.24	0.23	0.28	0.29
CS	0.28	0.27	0.29	0.32	**0.29**	0.28	0.28	0.28	0.24	0.26	0.24	0.23	0.28	0.29
LMMSE	0.27	0.27	**0.30**	0.29	0.34	0.28	0.27	0.29	0.24	0.26	0.23	0.23	0.28	0.29
AICC	0.27	0.27	0.29	0.30	**0.30**	0.32	0.28	0.28	0.25	0.27	0.24	0.23	0.28	0.30
SSDD	0.27	0.28	0.28	**0.28**	0.28	0.27	0.30	0.27	0.25	0.26	0.24	0.24	0.27	0.28
AP	0.26	0.28	0.29	0.27	**0.29**	0.26	0.26	0.35	0.22	0.25	0.23	0.21	0.26	**0.30**
RI	0.28	0.27	0.28	0.27	0.28	0.27	0.27	0.26	0.30	**0.29**	0.27	0.26	0.28	0.29
MLRI	0.27	0.27	**0.29**	0.28	0.28	0.27	0.27	0.27	0.28	0.30	0.26	0.25	0.28	**0.29**
ARI	0.27	0.26	0.28	0.27	0.27	0.27	0.26	0.26	0.28	**0.28**	0.29	0.26	0.28	**0.28**
CDM	0.26	0.26	0.27	0.27	0.27	0.27	0.26	0.26	0.26	0.27	0.26	0.28	**0.27**	**0.28**

Table 5.3: Same results as in Table 5.1, but images are JPEG-compressed at quality 90 between the initial and analysing demosaicing operations. Contrarily to the uncompressed case, detection is no longer trivial when the demosaicing algorithm is known, since there is a loss of information between the two steps. Although inclusive results can be expected to be better than exclusive results, they rely on knowing the original algorithm, which is not always the case.

probability of making a correct detection is above the baseline, making detections is still theoretically possible. However, if this probability is barely above the baseline, reliable detections require bigger images to be statistically significant.

Until now, the best grid was decided in 2×2 blocks. We now study whether double demosaicing can find the correct pattern in the full image. These results are shown in Figure 5.3.

When the original demosaicing algorithm is known or in the list of models, all 16 images are detected in the correct pattern on uncompressed and JPEG-95-compressed images. At a quality factor of 90, the detection is still almost perfect, with only one error on the GBTF-demosaiced images. Scores remain very good at a quality of 85, and are lower but still decent at a quality factor of 80. Even though the 2×2 block detections are barely above the random baseline, this is enough to provide mostly-accurate detections despite the images' small size of 704×469 pixels. As seen in the blockwise study, analysing bilinear-demosaiced images is difficult without knowing the image was demosaiced with bilinear demosaicing. Similarly, using ARI or CDM to analyse images yields inconsistent results on most original algorithms.

To conclude our study, we study whether double demosaicing could be used to detect forgeries. Indeed, if the demosaicing pattern could be detected locally in an image, it would be possible to find inconsistencies. However, one would need to be able to assess how confident the detection in a pattern is. To study this ability, instead of only looking at whether the detected position is correct, we check how the difference between the RMSE in the correct and the average RMSE across all

A \ B	BIL	HA	GBTF	CS	LMMSE	AICC	SSDD	AP	RI	MLRI	ARI	CDM	**Exclusive**	**Inclusive**
BIL	0.36	0.24	0.25	0.26	0.24	0.24	0.25	**0.26**	0.24	0.24	0.24	0.23	0.25	**0.31**
HA	0.27	0.30	0.27	0.27	0.27	0.26	**0.28**	0.27	0.25	0.26	0.24	0.23	0.26	0.27
GBTF	0.27	0.27	0.33	0.28	**0.30**	0.28	0.27	0.28	0.25	0.26	0.24	0.23	0.28	0.29
CS	0.28	0.27	0.29	0.32	**0.29**	0.28	0.28	0.28	0.24	0.26	0.24	0.23	0.28	0.29
LMMSE	0.27	0.27	**0.30**	0.29	0.34	0.28	0.27	0.29	0.24	0.26	0.23	0.23	0.28	0.29
AICC	0.27	0.27	0.29	0.30	**0.30**	0.32	0.28	0.28	0.25	0.27	0.24	0.23	0.28	0.30
SSDD	0.27	0.28	0.28	**0.28**	0.28	0.27	0.30	0.27	0.25	0.26	0.24	0.24	0.27	0.28
AP	0.26	0.28	0.29	0.27	**0.29**	0.26	0.26	0.35	0.22	0.25	0.23	0.21	0.26	**0.30**
RI	0.28	0.27	0.28	0.27	0.28	0.27	0.27	0.26	0.30	**0.29**	0.27	0.26	0.28	0.29
MLRI	0.27	0.27	**0.29**	0.28	0.28	0.27	0.27	0.27	0.28	0.30	0.26	0.25	0.28	**0.29**
ARI	0.27	0.26	0.28	0.27	0.27	0.27	0.26	0.26	0.28	**0.28**	0.29	0.26	0.28	**0.28**
CDM	0.26	0.26	0.27	0.27	0.27	0.27	0.26	0.26	0.26	0.27	0.26	0.28	**0.27**	**0.28**

Table 5.4: Same results as in Table 5.1, but images are JPEG-compressed at quality 85 between the initial and analysing demosaicing operations. Contrarily to the uncompressed case, detection is no longer trivial when the demosaicing algorithm is known, since there is a loss of information between the two steps. Although inclusive results can be expected to be better than exclusive results, they rely on knowing the original algorithm, which is not always the case.

patterns. We compare these results to the difference of RMSE in images that were not demosaiced. In this case, since there is no original demosaicing, there is no correct position either: we instead compute the difference between the average RMSE across all patterns, and the smallest of the four RMSE. Results are presented with the automatic inclusive and exclusive local selection of demosaicing algorithms. When working on images without demosaicing, there is no algorithms to exclude, and the automatic inclusive and exclusive methods are equivalent, thus we only show one number.

We first perform this analysis using the full images. Results are presented in Table 5.6. One can see that the difference between the four patterns' residuals is much larger on images that actually went through an initial demosaicing. Although the difference lowers on highly-compressed images, even at a quality factor of 80 it is still much higher than on images without a mosaic. As was seen before, bilinear-demosaiced images are not analysed correctly when the bilinear algorithm is not used for analysis. The pattern with the lowest residual is not the correct one, hence a negative difference with the exclusive automatic method. However, there is still a strong contrast in magnitude between the patterns. This means that even when the correct pattern cannot be detected, we can be confident that the image had been demosaiced prior to analysis. However, the downside to this is that we cannot be entirely confident that the detected demosaicing pattern is indeed correct.

In Table 5.7, we perform the same experiment at a local scale. Instead of computing the RMSE difference over each image, we compute it in each 32×32 block of the images and average the results. When the image has been demosaiced, the

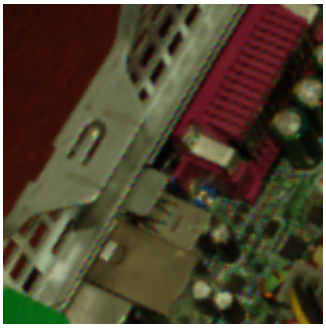
A \ B	BIL	HA	GBTF	CS	LMMSE	AICC	SSDD	AP	RI	MLRI	ARI	CDM	**Exclusive**	**Inclusive**
BIL	0.27	0.25	**0.26**	0.25	0.26	0.25	0.25	0.26	0.25	0.25	0.25	0.24	0.25	0.25
HA	0.26	0.26	0.26	0.26	0.26	0.25	0.26	**0.27**	0.25	0.25	0.24	0.24	0.26	0.26
GBTF	0.26	0.26	0.28	0.27	**0.27**	0.26	0.26	0.27	0.24	0.25	0.24	0.23	0.26	0.26
CS	0.26	0.26	0.27	0.28	**0.27**	0.26	0.26	0.27	0.24	0.25	0.24	0.23	0.26	0.26
LMMSE	0.26	0.26	**0.27**	0.27	0.28	0.26	0.26	0.27	0.24	0.25	0.24	0.23	0.26	0.27
AICC	0.26	0.26	0.27	0.27	**0.27**	0.27	0.26	0.27	0.25	0.25	0.24	0.23	0.26	0.27
SSDD	0.26	0.26	0.26	0.26	0.26	0.26	0.26	**0.27**	0.25	0.25	0.24	0.24	0.26	0.26
AP	0.26	0.26	0.27	0.26	**0.27**	0.25	0.26	0.29	0.23	0.24	0.23	0.22	0.25	0.26
RI	0.26	0.26	**0.27**	0.26	0.27	0.26	0.26	0.26	0.26	0.26	0.25	0.25	0.26	0.26
MLRI	0.26	0.26	0.27	0.26	**0.27**	0.26	0.26	0.26	0.26	0.26	0.25	0.24	0.26	0.26
ARI	0.26	0.25	0.26	0.26	**0.26**	0.26	0.25	0.26	0.26	0.26	0.26	0.25	0.26	0.26
CDM	0.25	0.25	**0.26**	0.26	0.26	0.26	0.26	0.26	0.25	0.25	0.25	0.25	0.26	0.26

Table 5.5: Same results as in Table 5.1, but images are JPEG-compressed at quality 80 between the initial and analysing demosaicing operations. Contrarily to the uncompressed case, detection is no longer trivial when the demosaicing algorithm is known, since there is a loss of information between the two steps. Although inclusive results can be expected to be better than exclusive results, they rely on knowing the original algorithm, which is not always the case.

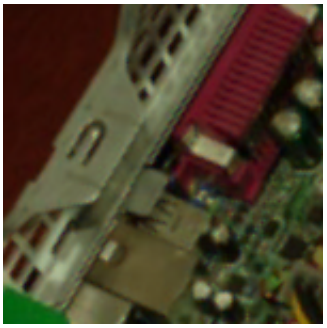
difference between the average of the pattern's RMSE and the correct pattern's is much lower than when computing the difference across the full image. On the other hand, when the original image features no mosaic, the difference is about ten times as high as on full images. This can be easily explained. When there are traces of demosaicing, a bias leads to the correct pattern's residual having a smaller amplitude. If smaller blocks are used, there are fewer samples for that bias to appear significantly, as a consequence the difference is smaller. On the other hand, in the absence of demosaicing traces, there is no bias that can alter one of the residuals, and the expectation for each pattern's RMSE is the same. In the absence of bias, smaller blocks are more affected by the natural variance of the residuals than the full image, leading to a higher difference than on full images. Despite this, the residual on demosaiced images still features significantly more contrast than non-demosaiced images on uncompressed images, and is still slightly above the contrast of non-demosaiced images after a compression at a quality factor of 90. However, against a stronger compression ($Q = 80$), the contrast between residuals cannot be used to distinguished images with and without mosaic.

Although double demosaicing can efficiently analyse demosaicing traces at the image level, as of now it is thus unable to make confident detections locally.

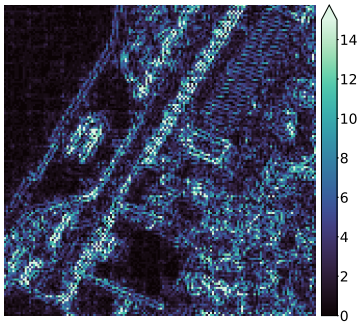
Several visual results are presented in Figure 5.4.



(a) Initial bilinear demosaicing in the $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ position.



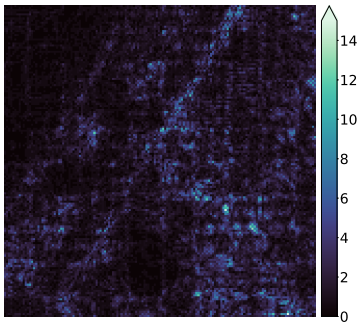
(b) Bilinear demosaicing of the compressed image, in the $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ position.



(c) Residual of the bilinear demosaicing

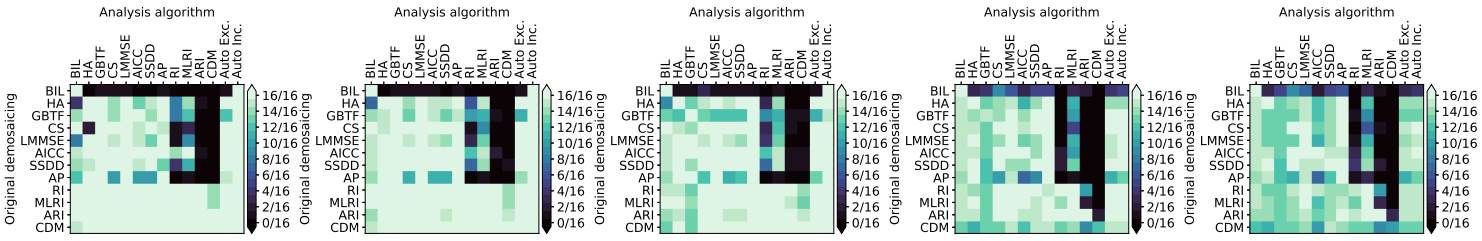
(d) After JPEG compression, $Q = 85$.

(e) CDM demosaicing of the compressed image, in the $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$ position.



(f) Residual of the CDM demosaicing

Figure 5.2: An image demosaiced with bilinear demosaicing in the $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ position was JPEG-compressed, and redemosaiced with several algorithms in different positions. JPEG compression remove most of the high-frequency demosaicing artefacts, however these are added again when performing a second demosaicing. As a consequence, performing a second bilinear demosaicing actually creates new artefacts and yields a higher residual than using another method, even in an incorrect pattern.



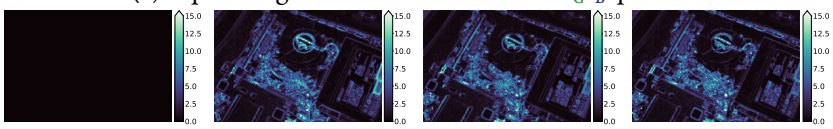
(a) Uncompressed (b) JPEG 95 (c) JPEG 90 (d) JPEG 85 (e) JPEG 80

Figure 5.3: Number of images that are detected in the correct pattern, for each pair of algorithms as well as with the automatic methods.

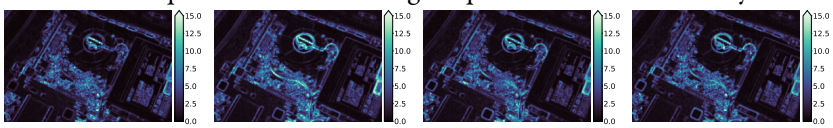
5.4 Discussion

In this chapter, we have studied the possibility of using double demosaicing to detect the pattern of an image. When an image has been through demosaicing in an unknown pattern, a second demosaicing can be applied in all four patterns. No information is lost when the initial pattern is used for the second demosaiced. As a consequence, the residual is lower or even zero in the correct pattern than in the

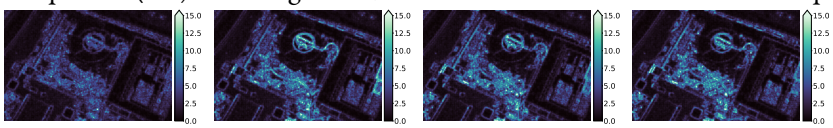
(a) Input image, HA-demosaiced in the $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ pattern.



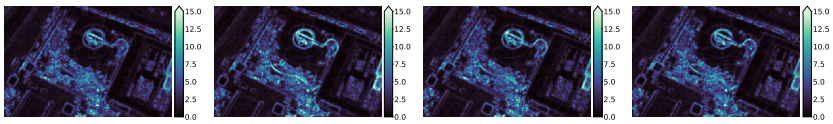
(b) Residuals when the input image is demosaiced again with the same algorithm (HA) in the four positions, from left to right $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ (correct pattern), $\begin{smallmatrix}B&G\\G&R\end{smallmatrix}$, $\begin{smallmatrix}G&R\\B&G\end{smallmatrix}$, $\begin{smallmatrix}G&B\\R&G\end{smallmatrix}$. The residual is zero when the correct pattern is used, making the pattern identification easy.



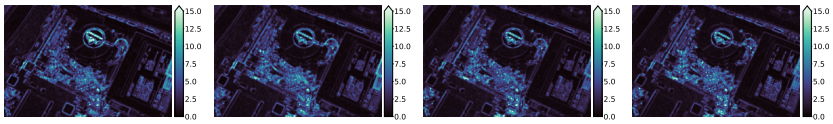
(c) Here, the input image is analysed with a different algorithm (LMMSE). The residual in the correct pattern (left) is no longer zero, but is still weaker than on the incorrect patterns.



(d) The input image is JPEG-compressed ($Q = 90$ before the second demosaicing, with the same (HA) algorithm. Again, although the correct pattern's residual is not zero, it is still weaker than in other positions.



(e) The input image is JPEG-compressed ($Q = 90$ before the second demosaicing, with a different (LMMSE) algorithm. The residual is still weaker with the correct pattern.



(f) Here, the (uncompressed) input image is analysed with yet another algorithm (ARI).

Figure 5.4: An image was demosaiced with HA and analysed with several algorithms, both compressed and uncompressed. When the image is not compressed between the two demosaicing operations, and if the algorithm used is the same, the residual will be zero in the correct mosaic pattern. When the image is compressed, the residual is no longer zero in the correct pattern, but is still weaker than in the other patterns as long as the compression is moderate. If a different algorithm is used for analysis, results vary: Analysing the image with LMMSE yields a lower residual in the correct pattern, but this is no longer the case when using ARI, which is not suited to analyse HA-demosaiced images. Residuals are blurred ($\sigma = 1.7$) for better visualization.

A \ B	Exclusive	Inclusive
BIL	1.76	-0.55
HA	0.94	0.42
GBTF	0.91	0.33
CS	0.83	0.43
LMMSE	0.82	0.48
AICC	0.82	0.46
SSDD	0.91	0.46
AP	0.80	0.23
RI	0.89	0.59
MLRI	0.82	0.57
ARI	0.83	0.46
CDM	0.70	0.41
No demosaicing	0.01	

(a) Uncompressed

A \ B	Exclusive	Inclusive
BIL	0.62	-0.27
HA	0.28	0.23
GBTF	0.33	0.26
CS	0.33	0.28
LMMSE	0.34	0.31
AICC	0.37	0.33
SSDD	0.28	0.27
AP	0.35	0.20
RI	0.33	0.30
MLRI	0.32	0.31
ARI	0.27	0.25
CDM	0.22	0.21
No demosaicing	0.01	

(b) JPEG $Q = 90$

A \ B	Exclusive	Inclusive
BIL	0.07	0.01
HA	0.11	0.10
GBTF	0.15	0.15
CS	0.12	0.10
LMMSE	0.16	0.14
AICC	0.17	0.16
SSDD	0.12	0.12
AP	0.16	0.08
RI	0.13	0.13
MLRI	0.13	0.13
ARI	0.11	0.11
CDM	0.08	0.08
No demosaicing	0.01	

(c) JPEG $Q = 80$

Table 5.6: Difference between the L_2 norm of the residual (RMSE) averaged over the four possible patterns and taken only on the correct position ($\begin{smallmatrix}R & G\\ G & B\end{smallmatrix}$). Results are shown with automatic local selection of algorithms, both allowing (Inclusive) and refusing (Exclusive) the use of the same algorithm for original demosaicing and analysis. On the last row, the original image presents no mosaic, and thus no correct pattern, the difference is thus computed between the average of the four residuals and the lowest residual, and only one number is presented since the two automatic methods are the same. The difference is shown for images in the $[0, 255]$ range. We can see that in the absence of original demosaicing, the residuals on the four patterns are very similar in magnitude, whereas the contrast is much stronger when the image was first demosaiced. JPEG compression reduces the contrast on demosaiced images, but even at a quality factor of 80 the residual difference is still about 10 times higher than without original demosaicing. Even when the correct position does not feature the lowest residual (such as when analysing bilinear-demosaiced images with the exclusive method), the contrast is high, enabling one to say with almost absolute certainty whether the image has been demosaiced.

other positions.

When the initial algorithm is not known, the pattern can be selected with a simple strategy. A list of algorithms can be used, and the algorithm that yields the lowest residual is chosen locally. Expectedly, this strategy yields excellent results when the initial demosaicing algorithm is in the list of tested algorithms. Even when this is not the case, however, this method usually improves on a comparison using a single algorithm.

Often, images are found in a compressed state. In this work, we studied the robustness of our method when JPEG compression is applied after the initial demosaicing. Although the detection is made harder by compression, the correct pattern can still be found at the image level even when the original demosaicing algorithm

A \ B	Exclusive	Inclusive
BIL	1.43	-0.36
HA	0.76	0.35
GBTF	0.74	0.37
CS	0.68	0.35
LMMSE	0.66	0.39
AICC	0.66	0.36
SSDD	0.77	0.38
AP	0.64	0.18
RI	0.75	0.50
MLRI	0.69	0.46
ARI	0.69	0.38
CDM	0.60	0.35
No demosaicing	0.10	

(a) Uncompressed

A \ B	Exclusive	Inclusive
BIL	0.43	-0.17
HA	0.19	0.15
GBTF	0.23	0.20
CS	0.23	0.19
LMMSE	0.24	0.21
AICC	0.26	0.23
SSDD	0.20	0.18
AP	0.24	0.14
RI	0.23	0.21
MLRI	0.22	0.21
ARI	0.18	0.17
CDM	0.15	0.15
No demosaicing	0.10	

(b) JPEG $Q = 90$

A \ B	Exclusive	Inclusive
BIL	0.04	0.01
HA	0.07	0.07
GBTF	0.10	0.09
CS	0.08	0.06
LMMSE	0.10	0.09
AICC	0.11	0.11
SSDD	0.08	0.08
AP	0.10	0.05
RI	0.08	0.08
MLRI	0.08	0.08
ARI	0.07	0.07
CDM	0.05	0.05
No demosaicing	0.07	

(c) JPEG $Q = 80$

Table 5.7: We perform the same experiment than in Table 5.6, but this time the difference of RMSE is computed over 32×32 blocks. The contrast between demosaiced and mosaiced images is much weaker than when analysing the full image, thus making it harder to detect whether a block has been demosaiced, and in which position.

is not known.

The main drawback of this method comes from the difficulty of making reliable detections at a local scale. While it is possible, with a very high confidence, to detect whether and in which pattern a whole image has been demosaiced, this decision becomes much harder to make locally. In small 32×32 blocks, the contrast of the residual across patterns is still biased towards detecting a lower residual in the correct position; however this contrast is not significantly higher than the contrast on images without a mosaic, in which no detection should be made. This is especially the case on highly-compressed images.

Overall, this method can be used to analyse the image at a global scale, helped by its surprisingly good robustness to JPEG compression. It is not yet possible, however, to use it locally. Local analysis would be needed to detect mosaic inconsistencies and thus potential forgeries.

This fourth chapter in the quest for reliable evidence of demosaicing traces again fails to yield a universal answer. Even the most natural assumption of closeness between different demosaicing algorithms is not always valid locally, especially under strong compression. In the next chapter, we will start from the even more natural hypothesis that demosaicing traces have a strong 2-periodic components. Positional learning will be introduced to train a convolutional neural network to mimic the phase of these traces, so as to expose its shifts and other inconsistencies.

Chapter 6

Positional Learning for Demosaicing Analysis:

Leveraging the Translation-Invariance of Convolutional Neural Networks

Abstract

In the previous four chapters, we explored various ways of identifying the mosaic. Direct detection, while able to identify the grid in many cases, is flawed against several demosaicing algorithms, and is hampered by the difficulty of designing features that reflect demosaicing algorithms' features. Reverse-engineering of the demosaicing algorithm of a specific image is not flawless either; even the all in all very natural assumption that the demosaicing algorithm will locally behave closely to at least one known algorithm is not always true, and fails to do miracles on highly-compressed images.

Faced with the difficulty of our problem, we decide to go back to the most bare and natural assumption about demosaicing traces: they feature a strong 2-periodic component. Without any other assumption, we propose to detect the phase of that component. In order to do this, we introduce positional learning. Leveraging the translation-inheritance and the high representative power of convolutional neural networks (CNN), we train one to detect the modulo-$(2, 2)$ position of each pixel. Implicitly, the CNN will rely on demosaicing traces to provide its output; said output will thus mimic the phase of the 2-periodic component.

When a forgery disrupts the image's mosaic, the network's output reflects this disruption, enabling the detection of the forgery as errors in the output. This method is fully self-supervised, requiring only authentic images for training. Furthermore, given several similar images whose authenticity is not clear, it is possible to fine-tune the model on the images to increase robustness to JPEG compression.

This chapter is a stepping stone towards the final method which will be introduced in Chapter 7 (Internal Learning to Improve Adaptability); as such many of its core concepts are explained here.

An interactive demo for this chapter is available at https://ipolcore.ipol.im/demo/clientApp/demo.html?id=77777000098.

6.1 Introduction

There are two concurrent paradigms for forgery detection techniques. The first way consists in developing many different methods, that address separately the varied forgeries and inconsistencies created by these forgeries. Error Level Analysis (ELA) [95] fits in this category and creates a heatmap by recompressing the image and visualising the difference. As we just mentioned, many methods look for inconsistencies in JPEG encoding; many other try to detect noise discrepancies [17], [26], [96]–[110] or attempt to directly detect internal copy-move operations [37], [73], [111]–[114]. The variety of setups before and after forgery makes exhaustiveness difficult, yet results obtained by such specific methods are self-explanatory. However, with few exceptions such as the recent development of Siamese Networks, which we briefly describe in Chapter 0.4 (Introduction), most of these methods are created manually, which can limit their performances, especially when forged images are created with a combination of methods rather than just one move.

Another possibility is to consider forgery detection as a unique learning problem and develop a structure – usually a neural network – to classify and/or localize forgeries independently of the setup and forgery type. For instance in [30] a heat map is computed, in [115] the network segments the image into forged and non-forged parts. See also [116] and [117]. While exhaustiveness is theoretically possible with these methods, it is actually limited by the database itself: they learn how to detect forgeries as seen in a training database, and can thus fail when confronted with images whose forgeries were made differently.

In this chapter, we try to bring the representative abilities of neural networks to the problem of demosaicing analysis. As seen in the previous chapters, most existing methods make assumptions of linearity of the interpolation and/or even assume the colour channels to be independently demosaiced. These assumptions are invalid with most commonly-used demosaicing methods. Still, taking into account these non-linearities and the interchannel transfers is difficult with manual methods, mainly due to the variety of ways with which those occur.

Indeed, many different demosaicing algorithms exist, furthermore most of those used in commercial cameras are undisclosed. Learning-based methods, while advantaged in this case against manual methods, must nevertheless take into account the impossibility to learn on all existing algorithms.

In this chapter, we overcome the above limitations by using an self-supervised convolutional neural network that learns to detect changes in the underlying pattern of mosaic artefacts. This network can be trained on unlabelled authentic images to detect forgeries in new images. It can also be fine-tuned directly on a database of potentially forged images to adapt to JPEG compression.

Our contributions here are two-fold. We create a new convolutional neural network (CNN) structure tailored to the specific task of mosaic artefacts detection, and that beats state-of-the-art mosaic detection methods. It can be trained in a fully unsupervised manner, and can even be directly retrained on a set of images to adapt to their specific conditions. To do that, we propose a new use for pixelwise convolutions in neural networks. Their main use in the literature has been to reduce the dimensionality of a network before performing heavier spatial operations,

such as in [118]. We argue that they can also be used stacked to each other, to process the causality relations between previously-computed spatial features as, for the same price as spatial convolutions, they can have more and bigger layers; furthermore, they do not add any more spatial dependency to the results.

6.2 Proposed method

A standard approach to finding copy-move forgeries through demosaicing artefacts would be to first detect the image's initial mosaic, and then detect if parts of the image actually have a different mosaic. Our manual attempts to detect the original mosaic were not successful. Indeed, criteria to do this heavily depend on the demosaicing method. Instead, we designed a convolutional neural network (CNN) to train on blocks of the image and directly predict their position in the image modulo $(2, 2)$. The only cue to this relative position are the periodic artefacts, such as CFA, resampling and JPEG artefacts. Hence, a change of the mosaic can lead to forged blocks being detected at incorrect positions modulo (2,2) and thus flagged as forged. Because the target output is only the relative position of blocks on the image, all that is required to train the network is a set of demosaiced images, without additional labels.

In a standard unsupervised scenario, the CNN can be trained with many authentic images and then used on new images to detect forgeries on them. However, if we have to detect forgeries on a large database, and if we can assume that the images in the database are similar in terms of demosaicing and post-processing – and in particular JPEG compression –, then we can retrain the CNN, performing unsupervised transfer learning directly on the test data. As the forged regions generally occupy a small part of the images, and only a small proportion of the images under study are forged, the risk that the CNN will overfit on the forged regions will be small.

The network consists of several parts, all of which serving different purposes. It only uses 31,504 trainable parameters. In the initial training phase, overfitting can occur both on the image contents and on the specific algorithms used for demosaicing. Although the former can easily be avoided by using more images for training, avoiding overfitting on the algorithms is harder. The small size of the network thus helps to avoid overfitting during training. It is even more useful when retraining on the same images to be studied, as overfitting on those images is much harder to avoid, and can make the network miss forgeries.

Spatial network

The first layers extract spatial features from the images. Due to the nature of demosaicing, we make use of two specific types of convolutions.

Most demosaicing algorithms try to avoid interpolating against a strong gradient [58], which would lead to visual artefacts. As a consequence, they often interpolate in one direction along edges. To mimic this, the first layers perform 10 horizontal, 10 vertical and 5 full convolutions, which are concatenated at the end of each layer.

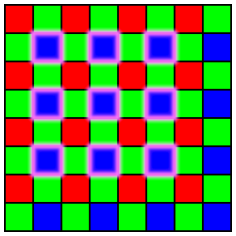


Figure 6.1: If we use a 3×3 convolution with a dilation of 2, the convolution at the central pixel sampled in blue only involves pixels sampled in the same colour. More generally, a 2-dilated convolution will look at pixels that all belong to the same colour channel.

In a mosaiced image, only one in four pixels is red and one in four is blue. As visualised in Fig. 6.1, this means that at the location of a sampled pixel, the closest sampled neighbours are all located at 2 pixels distance horizontally and/or vertically of the current position. We can take advantage of this by using dilated convolutions, which will only involve pixels belonging to the same mosaic.

We first use a sequence of two layers of 10 horizontal 1×3, 10 vertical 1×3 and 5 full 3×3 convolutions. In parallel, we perform 10 horizontal, 10 vertical and 5 full convolutions, which are all 2-dilated. The outputs of both parts are concatenated with a skip-connection from the input image. To this output is applied a similar sequence of two layers of 10 horizontal, 10 vertical and 5 full convolutions, in parallel with 10 horizontal, 10 vertical and 5 full convolutions with a dilation of 2. The spatial output is the concatenation of the output of the second and fourth non-dilated convolutions, and of the two dilated convolutions.

All layers in this part of the network are separated by a leaky rectified linear unit [119]. A diagram of this structure can be found in Fig. 6.2.

Pixelwise Causal network

Summarising, the network uses values that are up to four pixels away both horizontally and vertically from each pixel (the receptive field is thus 9×9). We consider this spatial span sufficient. Indeed, most demosaicing algorithms do not look farther to demosaic a given pixel. However, some algorithms still feature complex transfers between the different colour channels, especially in the high frequencies. As a consequence, the second part of our network consists of pixelwise (1×1) convolutions, which enable us to capture complex causal relations without adding more spatial dependencies to the convolutions. Although pixelwise convolutions are often used in the literature, their primary use is often to reduce data dimensionality. The Inception network [118], uses pixelwise convolutions before large convolutions to reduce dimensionality. Other networks use depthwise separable convolutions, where standard convolutions are replaced with one depthwise convolution followed by a pixelwise convolution [120], [121].

In our network, however, we do not stack them to reduce dimensionality, but to perform complex operations after the spatial features have been computed. Linking pointwise convolutions with each other enables us to represent complex relations

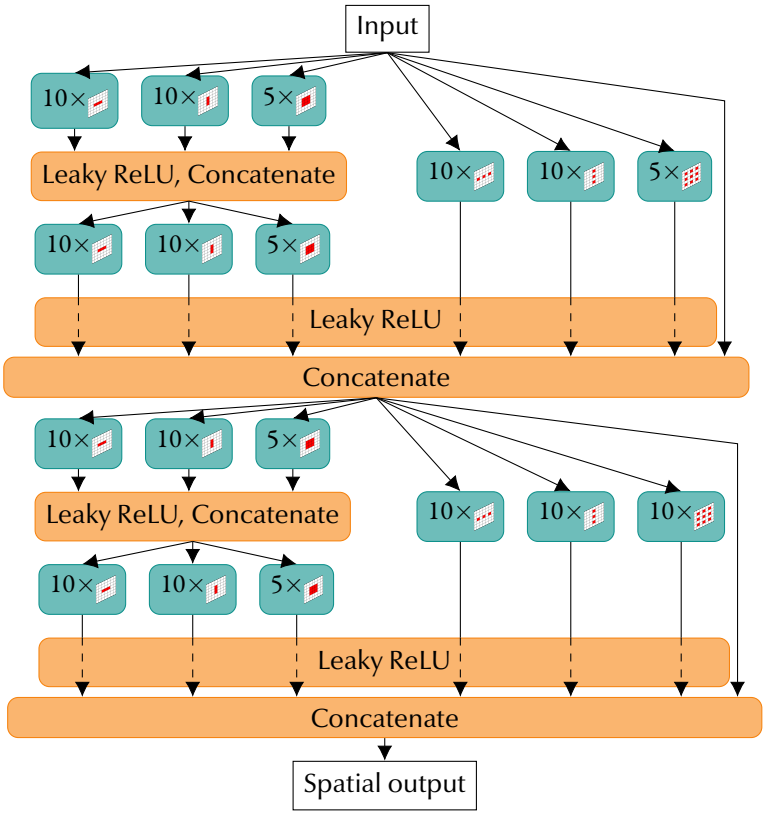


Figure 6.2: Spatial part of the network, containing 17,160 trainable parameters

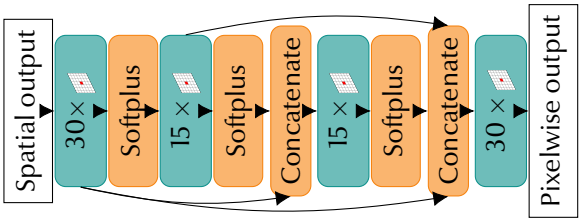


Figure 6.3: Pixelwise 1×1 convolutional part of the network, containing 6105 parameters

at a low computational cost, with few parameters and without incrementing spatial dependency.

This part of the network consists of four layers of respectively 30, 15, 15 and 30 1×1 convolutions. The output of the first convolution is skip-connected to the third and fourth convolutions, and the output of the second convolution is skip-connected to the fourth convolution. As a consequence, the last convolution takes the results of all previous pointwise layers into consideration to prepare features for the next step.

All the layers in this part of the network are separated by Softplus activation [122]. A diagram of the structure can be seen in Fig. 6.3.

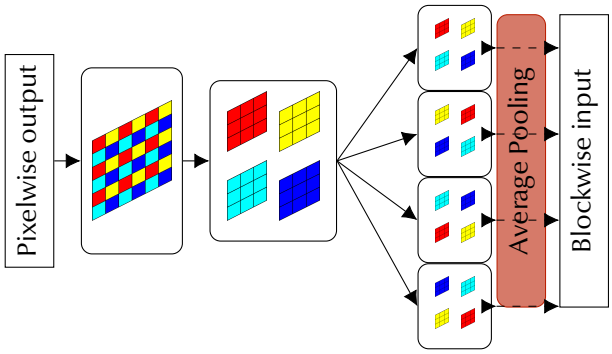


Figure 6.4: Processing the image into blocks

Blocks preparation

Although relative positions could be detected at the pixel level, grouping the pixels into blocks can lead to more reliable predictions. However, the blocks must be created carefully in order to avoid any bias.

Given an input image I of shape $(2Y, 2X, C)$, where C is the number of channels ($C = 30$ after our pixelwise network) and $2Y$ and $2X$ represent the spatial dimensions, we start by splitting the four modulo $(2, 2)$ positions of this image. We thus create four images I_{00}, I_{01}, I_{10} and I_{11}, each of shape (Y, X, C) and defined by

$$I_{\delta_x\delta_y}[y, x, c] = I[2y + \delta_y, 2x + \delta_x, c]. \tag{6.1}$$

We then concatenate these four images in different ways into four new images J_{00}, J_{01}, J_{10} and J_{11}, each of shape $(Y, X, 4C)$ and defined as follows:

$$\begin{aligned} J_{\delta_x\delta_y}[y, x, 4c] &= I_{\delta_x\delta_y}[y, x, c] \\ J_{\delta_x\delta_y}[y, x, 4c + 1] &= I_{(1-\delta_x)\delta_y}[y, x, c] \\ J_{\delta_x\delta_y}[y, x, 4c + 2] &= I_{\delta_x(1-\delta_y)}[y, x, c] \\ J_{\delta_x\delta_y}[y, x, 4c + 3] &= I_{(1-\delta_x)(1-\delta_y)}[y, x, c] \end{aligned} . \tag{6.2}$$

These four images are merely channel-wise permutations of one another, which enables the network to keep balance between the four patterns.

Finally, each of these images is decomposed in blocks. Because all spatial and pixelwise features have already been computed in the previous parts, we can directly view the decomposition in blocks as one big average pooling, so that each block is spatially represented by one pixel. We thus get four output images B_{00}, B_{01}, B_{10} and B_{11}, each of shape $(\frac{Y}{16}, \frac{X}{16}, 4C)$. Each image is thus spatially 32×32 times smaller than the original image.

Thanks to this permutation, the detection problem is slightly shifted: Pixels in $J_{\delta_x\delta_y}$ are shifted so that all blocks of $B_{\delta_x\delta_y}$ should be detected at the same relative position modulo $(2, 2)$, $\delta_x\delta_y$. This process is explained in Fig. 6.4.

Blockwise Causal network

Because blocks are represented through average pooling, each block is spatially represented by one pixel. As a consequence, creating new pointwise convolutions amounts to processing the data independently – but with shared weights – in each block.

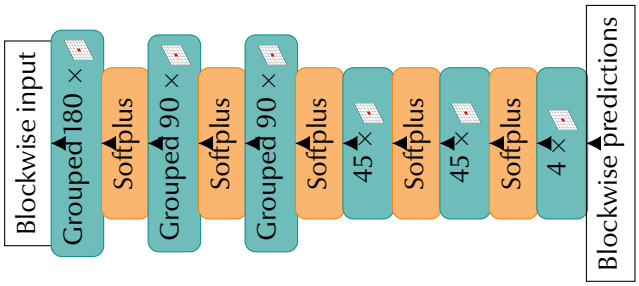


Figure 6.5: Blockwise part of the network, containing 8,239 trainable parameters

Furthermore, the four values $B_{\delta_x\delta_y}[y, x, 4c + i]$ for $i \in (0, 1, 2, 3)$ represent the same feature, averaged independently in each of the four possible mosaics $\delta_x\delta_y$. To compare these features separately before merging them, we start by stacking three layers of respectively 180, 90 and 90 grouped pixelwise convolution, where the output in one channel at one given block–position is made using only the values of the same feature, in the four mosaics and at the same position. Finally, we merge these features together with two additional layers, each of 45 full-depth pointwise convolutions. Like in the pixelwise network, the layers are separated by Softplus activation [122]. The structure of the blockwise causal network is shown in Fig. 6.5.

Decision and loss module

A final layer of four pointwise convolutions is placed to predict scores for each position. In an authentic image, all blocks from each image $B_{\delta_x\delta_y}$ would be expected to detect their own position as $\delta_x\delta_y$. If training on several images whose main mosaic may be different, we let the network permute the output of the four images either horizontally, vertically, or diagonally, in order to have the lowest of the four global losses before computing the local loss. This enables the loss to take into account the possibility of different images having different main positions.

Auxiliary prediction for training

Because the spatial and pixelwise networks are used at full resolution – whereas the resolution of images is reduced by a factor 32×32 in the blockwise network –, the first part of the network takes a higher computational toll than the rest. In order to speed up training, we work in a manner similar to [118] and start by training the spatial and pixelwise networks together. We add an additional layer of 4 pointwise convolutions at the end of the pixelwise network, and train it with the cross-entropy loss to detect the position of each pixel modulo $(2, 2)$.

Once the first part of the network is trained, we remove this auxiliary layer and process the output of the training images into blocks, as explained in 6.2. We then train the blockwise network, using the preprocessed output of the pixelwise network.

By training the first part of the network separately, and more importantly using a loss computed at full resolution, we can train it in fewer and faster iterations. Processing the images into blocks, which also requires a significant time, must only

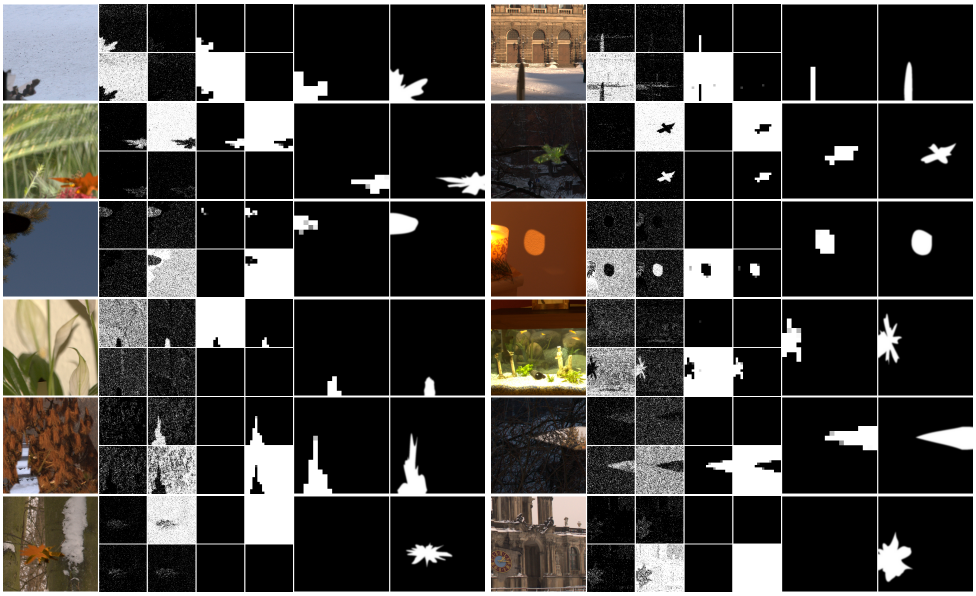


Figure 6.6: Network's results. Detections on the CFA Forgeries dataset, presented in Chapter 1 (Non-Semantic Evaluation of Image Forensics Tools). For each image, in order: Forged image, pixelwise predictions for each of the 4 grids (auxiliary network output), blockwise predictions for each of the 4 grids (full network output), detected forged blocks, ground truth. The mosaic of the image and the forgery is aligned for the two images in the last row, which explain why no detection can be made with our method.

be applied once between the two global training steps. Finally, the blockwise part of the network can be trained very quickly, because there is no need to propagate into and from the full-resolution network at each iteration, making each individual iteration quicker.

Training is done first on the spatial (Fig. 6.2) and pixelwise (Fig. 6.3) networks, using the aforementioned auxiliary layer. Then, the blockwise network (Fig. 6.5) is trained alone, using the results of the pixelwise network, processed into blocks as seen in Fig. 6.4. All training is done with the Cross-Entropy loss and the Adam optimiser [123], with a learning rate of 10^{-3}.

6.3 Experiments

For training, we took the images from the noise-free image dataset [33], which do not present any traces of demosaicing due to previous downsampling. We demosaiced them ourselves, each with three simple algorithms: bilinear interpolation, LMMSE [58] and Hamilton-Adams [55]. The size of all images is 704×469. We trained the first part of the network for 1500 iterations and the second part for 500 iterations with the Adam optimizer [123] at learning rate 10^{-3}.

Examples of detections on the CFA forgeries dataset presented in Chapter 1 (Non-Semantic Evaluation of Image Forensics Tools) can be seen in Fig. 6.6.

We also adapted the pretrained network to the database by retraining it directly on it for the 1000 iterations on the first part of the network and 500 on the second

part. This training was done without knowledge of which image is forged or authentic.

We compare our results with intermediate value mosaic detection[1] [14], variance of colour difference [15], as well as with ManTraNet [30], a state-of-the-art forgery detection method that directly trains a neural network to detect various forgeries on standard datasets. Results are measured with the ROC curve on the number of detected forged blocks.

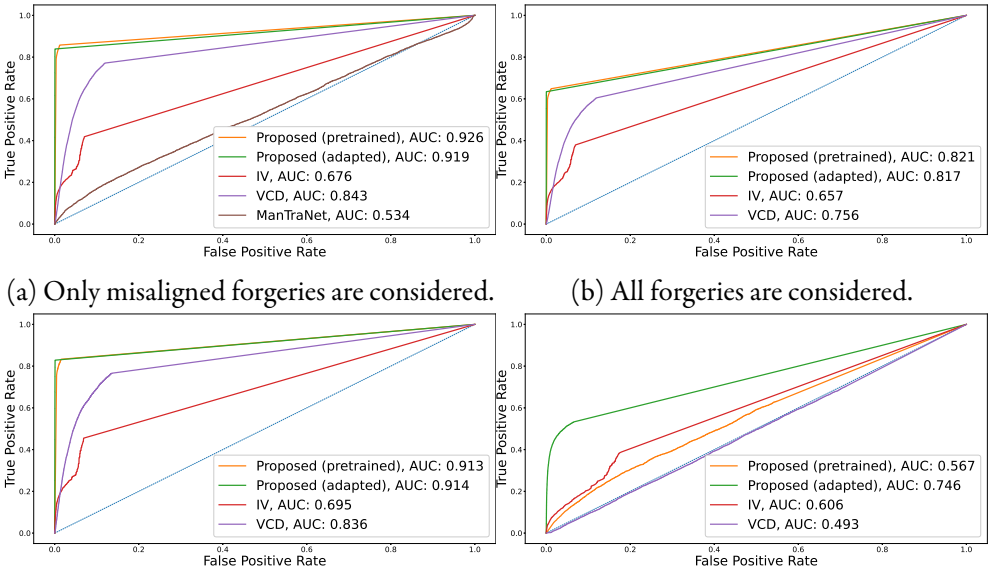
By nature of demosaicing, a region forged by copy-move has a $\frac{1}{4}$ probability of having its mosaic aligned with the main one, and in such case it cannot be detected by its CFA position. In our databases, aligned forgeries account for 26.7% of the total number of blocks. The results of our algorithms on the whole dataset is shown on Fig. 6.7b. Such results are closer to what could be detected in practical applications. However, because forgeries with aligned mosaic are not detectable by mosaic detection algorithms, we present other results with a modified ground truth, in which we consider a block as forged only if its mosaic is different than the position of the original image. These scores are thus given relative to what could theoretically be detected with perfect knowledge of the mosaics. Results under this definition of the ground truth can be seen in Fig. 6.7a.l The database features three algorithms that were also used for pretraining the network: bilinear interpolation, LMMSE [58] and Hamilton-Adams demosaicing [55]. In order to ensure fairness in the comparison, we remove all images demosaiced with, or containing a forged region demosaiced with, one of these three algorithms. The results are presented in Fig. 6.7c. We can see that the results are similar to those on the whole database, which shows that the network generalised well to new algorithms.

Finally, we test the robustness of our models to JPEG compression by compressing all the images at a quality of 95. The results are presented in Fig. 6.7d. [15] does no better than random guessing, with an AUC score of 0.52 in the global evaluation and 0.49 in the local evaluation, and both [14] and our pretrained network do little better. On the other hand, the adaptive network, by adapting directly to the database and thus learning to detect the CFA position over JPEG artefacts, was able to perform much better.

We also test our pretrained model on the Trace dataset, presented in Chapter 1 (Non-Semantic Evaluation of Image Forensics Tools), and on the Korus dataset [42], [43]. Quantitative results with the Matthews Correlation Coefficient can be seen in Tables 6.1a and 6.1b, and are compared with our implementation of Choi [14] presented in Chapter 3 (Intermediate Values Counting for CFA Pattern identification), demosaicing analysis methods Shin [15], Ferrara [9], Dirik [11] and Park [16], and state-of-the-art generic forgery detection method Noiseprint [27].

The proposed method yields excellent results on the Trace datasets, where the only cue to the forgery is the demosaicing itself. Results are much worse on the Korus dataset. Although the proposed method beats the state of the art on the Sony α57, detections are harder to make on the Nikon cameras, whose demosaicing traces are more subtle. One of the main reasons for this low score is that we try to

[1] This particular method was studied and extended in Chapter 3 (Intermediate Values Counting for CFA Pattern identification). However, the experiment in this chapter uses an earlier, more basic implementation.



(a) Only misaligned forgeries are considered.

(b) All forgeries are considered.

(c) Only misaligned forgeries, algorithms on which our network was pretrained are excluded.

(d) Only misaligned forgeries, images are ex-compressed at a JPEG quality of 95.

Figure 6.7: ROC curves comparing the detections of our methods to ManTraNet [30], Intermediate Values (IV) detection [14] and Variance of Colour Differance (VCD) detection [15].

find one single grid on each window. On the many forgeries of the Korus dataset that contain multiple small CFA patterns, this will lead to a very low confidence of detection. Furthermore, the method provides one output per 32×32 block, lowering the spatial precision of the detection.

In any case, the results on the Trace datasets validate the method and the ability of positional learning to help analyse demosaicing artefacts.

6.4 Discussion

We have shown that a small convolutional neural network could be used to accurately detect and interpret CFA artefacts, and subsequently use them to detect forgeries in images. Even without new training, this network can adapt well to images demosaiced with unseen and more complex algorithms than those studied during training. Our neural network is small and can process images almost as quickly as methods presented in the literature, while offering detections of superior quality.

We showed that state-of-the-art generic methods such as ManTraNet and Noiseprint are blind to shifts in the demosaicing pattern, even though Noiseprint is to some extent able to detect changes of demosaicing algorithm. This indicates that methods such as ours, which focus specifically on demosaicing artefacts, still have an important role to play, and are fully complementary to more generic methods.

Our network was trained on few images, which were not taken from the evaluation dataset, and with only three algorithms. This enabled us to show, in Fig. 6.7c, that we could get strong results even on images demosaiced with algorithms on

Method	Grid Exo	Grid Endo	Alg Exo	Alg Endo
Proposed	0.682	0.665	0.501	0.491
Shin[15]	0.104	0.099	0.085	0.084
Choi[14]	0.603	0.575	0.420	0.385
Ferrara[9]	0.071	0.095	0.218	0.238
Dirik[11]	-0.002	0.010	0.001	0.013
Park[16]	0.116	0.133	0.152	0.150
Noiseprint[27]	-0.001	0.002	0.066	0.060

(a) Results on the CFA Grid (Grid) and CFA Algorithm (Alg) datasets of the Trace database, with endomasks (Endo) and exomasks (Exo).

Method	Canon 60D	Nikon D7000	Nikon D90	Sony α57	All
Proposed	0.002	0.049	0.044	0.574	0.167
Noiseprint	0.153	0.322	0.236	0.148	0.202
Choi	0.004	0.176	0.251	0.251	0.238
Shin	0.021	0.003	0.012	0.511	0.143
Ferrara	-0.016	0.498	0.461	0.339	0.321
Dirik	0.036	0.241	0.275	0.062	0.153
Park	0.018	0.540	0.491	0.302	0.338

(b) Results on the Korus dataset

Table 6.1: Comparative results on the Trace CFA datasets (see Chapter 1 (Non-Semantic Evaluation of Image Forensics Tools)) and on the Korus dataset [42], [43].

which the network was not trained. In order to test and show its capacity to generalize to new images and algorithms, we only trained it with 18 images from a different dataset than the evaluation images, and three algorithms. A full instance of this network, trained with all known and available algorithm, would probably yield even better results.

Unfortunately, the pretrained model is not sufficient to process mosaic artefacts in compressed images. This is to be expected, as JPEG compression erases high frequencies even at a high quality, which is also where demosaicing algorithms leave artefacts. However, adapting the network to the new compressed data by retraining it directly on the studied data enabled it to retrieve demosaicing traces over JPEG compression.

This method is good enough to analyse uncompressed images. However, the focus of this chapter was the design of a learning-based method to analyse demosaicing artefacts. Little focus was put on the use of the network to automatically detect and localize forgeries. A necessary step, which will be explored in the next chapter, is to further process the results of the network, with automatic thresholding and better localization of the forgeries.

Another natural step concerns the possibility of fine-tuning. In this chapter, we saw that retraining the network on a study dataset of similar, potentially forged images, could help improve robustness to JPEG compression. In practice, however, it is rare to have a large quantity of similar images to analyse. Ideally, one should

be able to fine-tune the network on a single, potentially forged image. Adapting a pre-trained network to the testing data by retraining it on said data is of course something that must be done carefully. A network that is too big can easily overfit if too few samples are available, and see its quality lower in comparison to the pre-trained network. More experiments must thus be done to fully understand what can be done this way. Namely, two big questions arise: How much similar data is needed, and how similar does it need to be, in order to be able to improve the quality of a neural network by retraining it on this data? More importantly, can we prevent the network from overfitting when retrained on small amounts of data?

Since overfitting is the only reason why a network could worsen by trying to adapt to new data, it is likely that preventing or limiting it would make it possible to improve a network by retraining it on new data. We have shown that it was possible to drastically improve the performance of the network by retraining on the full database. Would it still be possible if we only have several images, or, in the most extreme but also the most frequent case, only one image?

This is a difficult challenge, however learning to make a network fit to new unlabelled data could greatly help make it more robust for practical applications; next chapter will thus focus on the internal fine-tuning of the network on a single image.

Chapter 7

Internal Learning to Improve Adaptability

Abstract

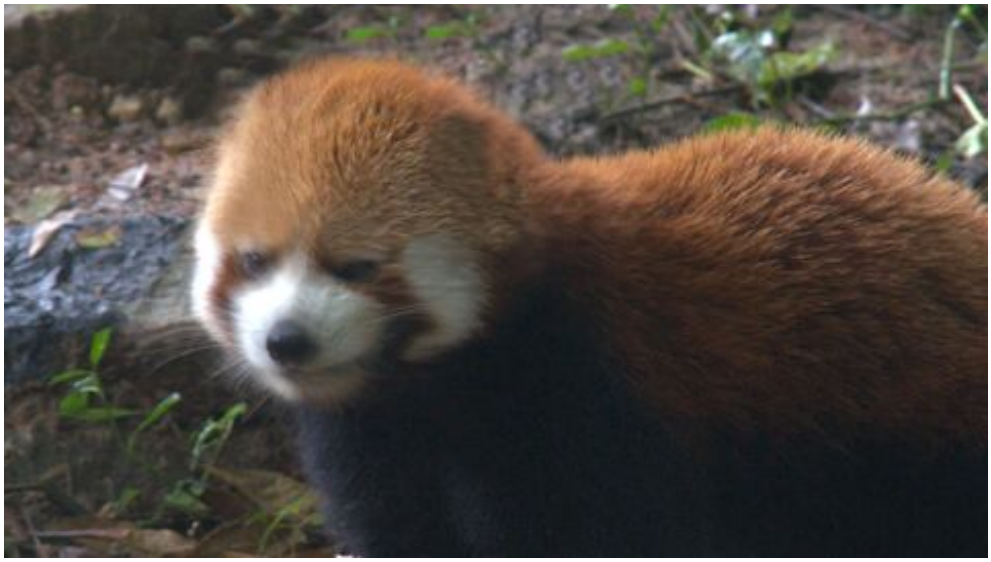
In Chapter 6, we used positional learning to make a network's output reflect the image's underlying CFA mosaic. We now show that such a network can be fine-tuned on a single, potentially forged image, to adapt to it. Doing so greatly increases robustness to JPEG compression and other post-processing. We further improve the method by adding a a contrario layer to automatically detect and localize significant inconsistencies in an image's mosaic. The method proposed in this final chapter beats state-of-the-art mosaic detection methods and generic forensic methods alike on uncompressed datasets. It remains relevant on slightly-compressed datasets, and provides a complementary insight to other methods, including generic learning-based models, which are blind to the mosaic traces we analyse.

7.1 Introduction

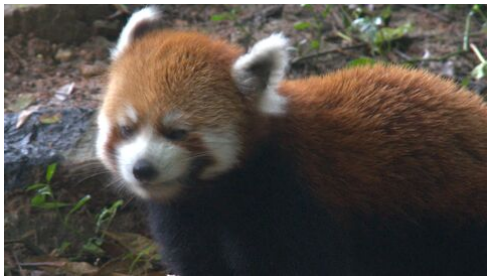
In the previous chapter, we proposed a self-supervised neural network that detects the relative position modulo 2 of pixels and blocks in the image. Because a CNN is invariant to translation, it implicitly needs to learn from mosaic artefacts in order to perform this classification. Shifts in the output of the network are thus evidence of forged regions. This network can be fine-tuned on a dataset of similar potentially forged images, and shows some resilience to JPEG compression.

However, fine-tuning on a full dataset of similar images is impractical. In most instances, only one image is given to analyse. Ideally, one should be able to fine-tune the network to a single, potentially forged image.

Single-image internal learning has indeed gained popularity in several domains of image processing in recent years. Using self-supervised information to retrain an over-parameterized network on the very image to process, such methods often provide better results and greater adaptability to uncontrolled cases than their supervised counterparts. For instance, in the close field of image denoising, Noise2Void [124] and the subsequent Self2Self [125] train a network to reconstruct a noisy image while hiding pixels from its input. The network provides the regularization necessary to actually denoise the image.

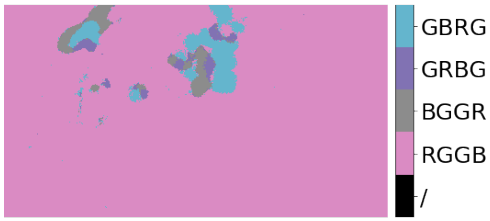


(a) Forged image



(b) Original image

(c) Forgery mask



(d) Detected demosaicing grids

(e) Detected forgeries

Figure 7.1: Results of the proposed method on an inpainted image from the Korus [42], [43] dataset. Local detection of the demosaicing pattern not only enables detection of the forgery, but also shows the patches used during inpainting.

Still, this learning scheme brings several challenges. Fewer sample makes overfitting more likely, yet if the network overfits it will correctly detect the position of the forgery's pixel, and the forgery will not be detected. More importantly in our case, JPEG compression poses difficulty to single-image fine-tuning. The proposed network uses demosaicing traces because those are the primary source of information on the modulo 2 position of the images. However, JPEG encoding compresses the image in 8×8 blocks, the traces it leaves behind thus have a strong 8-periodic component. Fine-tuning on a single compressed image may thus divert the network from looking at demosaicing traces to make it analyse JPEG compression traces instead. When fine-tuning on multiple images, this would be alleviated by the dif-

ferent alignments between the CFA and the JPEG grid on each image: a network trained on detecting JPEG traces on one alignment would fail on image aligned otherwise. With a single image, analysis of JPEG traces can directly lead to the correct modulo-2 position.

To avoid this problem, we propose to pretrain the network on manually-compressed images at different JPEG–CFA alignment. The network thus learns to detect CFA artefacts over JPEG compression. While this is not enough to produce good results on compressed images, it leads the model on a better track before internal learning: Previously prevented from using the JPEG grid position, starting to use this grid would have an immediate short-term cost for the network. Local optimization thus effectively incentivizes the network not to detect the JPEG grid position. To this method, we add the *a contrario* layer already introduced in Chapter 4. This enables our method to detect inconsistent regions in the image, even if a single mosaic cannot be detected locally. Statistically-insignificant results are then filtered out based on a NFA threshold.

This solution beats state-of-the-art methods on uncompressed datasets. Its robustness to various post-processing and counter-forensic artefacts such as JPEG compression makes it useful even on compressed images, where its automatic detections can be used alongside other specific or generic forensics methods. This complementarity is furthered by the blindness of state-of-the-art generic neural networks to shifts in periodic signals such as those we study.

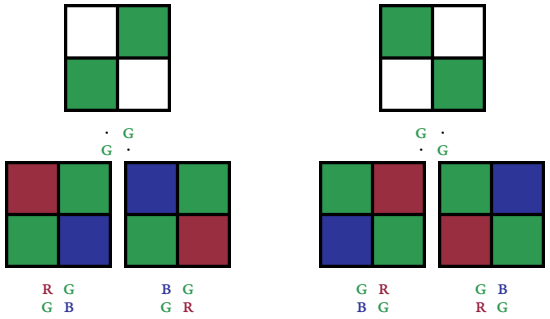


Figure 7.2: The four possible sampling patterns can be grouped by the diagonal on which the green channel was sampled: $\begin{smallmatrix} R & G \\ G & B \end{smallmatrix}$ and $\begin{smallmatrix} B & G \\ G & R \end{smallmatrix}$ share the $\begin{smallmatrix} \cdot & G \\ G & \cdot \end{smallmatrix}$ diagonal, whereas $\begin{smallmatrix} G & R \\ B & G \end{smallmatrix}$ and $\begin{smallmatrix} G & B \\ R & G \end{smallmatrix}$ share the $\begin{smallmatrix} G & \cdot \\ \cdot & G \end{smallmatrix}$ one.

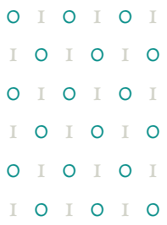
7.2 Method

We propose to train a fully-convolutionnal neural network (CNN) to detect the (modulo 2) positional information of each pixel of an image. Because a CNN is invariant to translation, it does not know these positions and has to infer them from camera traces, in particular from demosaicing traces. Inconsistencies in the output of the network reveal inconsistencies in the demosaicing pattern, and are thus traces of forgeries. In Sec. 7.2, we explain how the network is trained on positional data. In Sec. 7.2, we show how some level of robustness to JPEG compression can be achieved during training. In Sec. 7.2, we show that it is possible to retrain this network directly on a single suspicious image to adapt to the setup of this image, thus achieving a greater robustness to JPEG compression. Finally, in Sec. 7.2, we detail how the network's output can be used to detect forgeries.

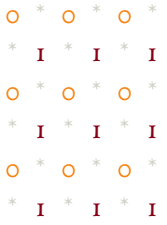
Positional learning

Following Chapter 6 (Positional Learning for Demosaicing Analysis), we propose to train a CNN to detect the modulo 2 position of each pixel, both horizontally and vertically. Coarser-scale position information is irrelevant to the very local demosaicing traces, and could lead the network to rely on unwanted cues. However, demosaicing detection algorithms [14], [15], [88] usually proceed in two steps: they start by detecting the diagonal pattern, ie. to find which pixels were sampled in green, then try to distinguish between the two patterns sharing the same diagonal. This two step method is preferable because detecting the diagonal is easier and more robust than directly making a decision on the full pattern. To adopt this behaviour, we trained a convolutional network to detect two features:

- For each pixel, the offset of its diagonal, representing whether the pixel is sampled in green or not (Fig. 7.3a);
- for pixels that are on the first diagonal, whether it is on an even or odd line. This step decides, for pixels which were not sampled in green, whether they were sampled in red or blue (Fig. 7.3b).



(a) The first detected feature is the diagonal offset. It decides whether the pixel was sampled in green.



(b) The second feature is the evenness of the line/column of the pixel on the main diagonal. It decides whether the pixel was originally sampled in red or in blue.

Figure 7.3: The CNN is trained to output these two patterns. Asterisks (*) correspond to values that are ignored.

This method is self-supervised, since the position of each pixel is known. The only requirement is that all images of the training set be demosaiced in the $\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}$ pattern, to enforce consistency of the output across images and correspondence between the detected position and the sampled colour[1]. If a training image has been demosaiced in another pattern, we align it to $\begin{smallmatrix}\mathrm{R}&\mathrm{G}\\\mathrm{G}&\mathrm{B}\end{smallmatrix}$ by cropping its first row and/or column.

We use the DnCNN [126] architecture, with 17 layers and 64 features per layer. DnCNN is more suited to our task than other standard structures, as it does not make use of any downsampling. The downsampling which is usually found in other CNN structures would remove the high-frequency information leading to the detection of potential demosaicing artefacts.

Training on JPEG-compressed images

JPEG compression is a major obstacle to demosaicing detection. The quantization induced during compression quickly removes the highest frequencies of an image, which contain demosaicing traces. As a consequence, strongly-compressed images keep no demosaicing traces and cannot be detected by our method. Nevertheless, it remains possible to find demosaicing traces on high-quality images, where the compression is minimal. This prompted us to train our network on JPEG-compressed images.

However, JPEG compression aggregates data in 8×8 blocks, thus creating strongly 8-periodic artefacts. Analysis of the JPEG artefacts would thus be enough for the CNN to correctly find the (mod. 2) position of all pixels without even looking at demosaicing traces. To prevent the network from doing this and to force it to analyse the demosaicing artefacts rather than JPEG compression, we start from uncompressed images and compress them ourselves in the four possible alignments between the demosaicing pattern and the JPEG grid, seen in Fig. 7.4. For each initial image, we thus have 4 compressed images with different alignments, and the network is trained simultaneously on them. Doing this prevents the network from

[1] If images whose grid is known were not available, it would still be possible to train on images of an unknown grid by applying the same method as during internal learning (see Sec. 7.2).

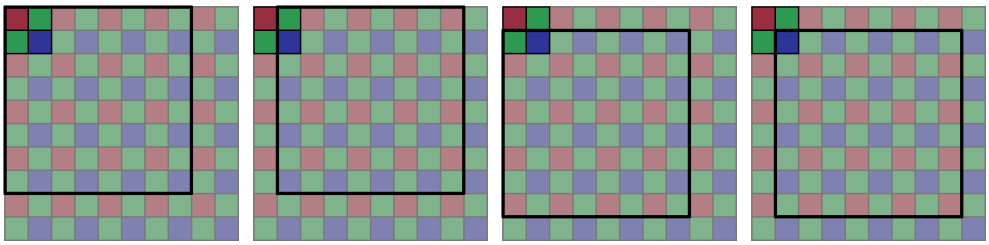


Figure 7.4: The JPEG grid and the Bayer pattern can be aligned in four different ways. By simultaneously training the network on the four possible alignments, we force it to analyse demosaicing traces over JPEG, instead of directly using the JPEG traces.

directly analysing and using JPEG compression to find the positions, as the positional cues provided by the compression are contradictory between the four compressed images.

This training scheme assumes that the JPEG grid is consistent across the image, ie. that the compression happened after the forgery. This assumption is reasonable; if the JPEG grid itself is inconsistent, the forgery will be apparent and much easier to detect through JPEG grid analysis [25].

Internal learning

Training on compressed images already provides some robustness to JPEG compression. However, there is a huge variety of post-processing algorithms, and an even bigger number of combinations thereof, as post-processing encompasses not only algorithms such as JPEG compression, but also image-enhancing filters often automatically applied by cameras, and specific counter-forensic measures such as added noise or median filtering. Furthermore, such algorithms change with time. We propose an alternative strategy. The robustness of our network can be further increased with internal learning, ie. by retraining the network on the specific image we want to study. This enables the network to adapt to the specific statistics of the image.

Given a potentially forged image to analyse, we assume the image is authentic and train the network repeatedly on it to detect the position of pixels, as explained in Sec. 7.2. Contrarily to the initial training, the image is not necessarily in the $\begin{smallmatrix}R&G\\G&B\end{smallmatrix}$ pattern. To train the network, we compute the loss not only on the initial target of Fig. 7.3, but we also shift the target by one row and/or column. We thus have 4 losses depending on the offset of the target, we use the one that is minimal across the global image. In other words, we train the network to be locally coherent with the globally-dominant pattern.

Of course, single-image training induces a risk of overfitting. However, even if the network overfits on the image, training is done on the hypothesis that the full image is authentic. In other words, fine-tuning incentivizes the network to conclude that everything is authentic. As a consequence, if the image is actually authentic, the risks of making a false detection are lower than with the pretrained-network; by design, even overfitting will not induce new false detections. In the more interest-

ing case where the image is indeed forged, the network will also adapt to the post-processing to learn demosaicing traces and detect the position of pixels. Forged regions in images are usually small compared to the total size. As a consequence, even though the forged regions would steer the network towards detecting their pixels' positions correctly, they would produce evidence contradicting that of much larger authentic regions, and the network should thus not learn too much from forged regions. The small size and locality of the network is particularly important here to prevent it from being able to adapt to both, the authentic and the forged regions. While some amount of overfitting is probably unavoidable, its impact can be expected to be limited.

During internal learning, we use the Ranger21 [127] optimiser with default parameters and a learning rate of 10^{-4}.

Forgery detection

The network does not directly detect forgeries in an image, it only detects pixel-wise, demosaicing-related positional information. This information must then be analysed to find forged regions of an image.

Block votes

The output of the network consists of two feature maps, following the targets of Fig. 7.3: the diagonal of the pixel O_d, and O_l the evenness of the line number of pixels on the main diagonal. Values are between 0 and 1, with values close to 0/1 signifying more confidence in the detection.

These results are aggregated in 2×2 blocks, corresponding to a Bayer CFA tile.Let B_d and B_l represent a block from O_d and O_l, three binary decisions are made on that block:

- $\delta_{\begin{smallmatrix}\cdot & \mathrm{G}\\ \mathrm{G} & \cdot\end{smallmatrix}\,\vert\,\begin{smallmatrix}\mathrm{G} & \cdot\\ \cdot & \mathrm{G}\end{smallmatrix}} \triangleq \left[\frac{1}{4}\left(B_{d_{0,0}} + B_{d_{1,1}} - B_{d_{0,1}} - B_{d_{1,0}}\right)\right]$,
- $\delta_{\begin{smallmatrix}\mathrm{R} & \mathrm{G}\\ \mathrm{G} & \mathrm{B}\end{smallmatrix}\,\vert\,\begin{smallmatrix}\mathrm{B} & \mathrm{G}\\ \mathrm{G} & \mathrm{R}\end{smallmatrix}} \triangleq \left[\frac{1}{2}\left(B_{l_{0,0}} - B_{l_{1,1}}\right)\right]$, and
- $\delta_{\begin{smallmatrix}\mathrm{G} & \mathrm{R}\\ \mathrm{B} & \mathrm{G}\end{smallmatrix}\,\vert\,\begin{smallmatrix}\mathrm{G} & \mathrm{B}\\ \mathrm{R} & \mathrm{G}\end{smallmatrix}} \triangleq \left[\frac{1}{2}\left(B_{l_{1,0}} - B_{l_{0,1}}\right)\right]$.

In these definitions, $[\cdot]$ represents rounding to 0 or 1. $\delta_{\begin{smallmatrix}\cdot & \mathrm{G}\\ \mathrm{G} & \cdot\end{smallmatrix}\,\vert\,\begin{smallmatrix}\mathrm{G} & \cdot\\ \cdot & \mathrm{G}\end{smallmatrix}}$ says in which diagonal the block – or rather its top-left pixel – is detected, from which the pattern diagonal can be inferred: a value of 0 (resp. 1) means the block is demosaiced in a $\begin{smallmatrix}\cdot & \mathrm{G}\\ \mathrm{G} & \cdot\end{smallmatrix}$ (resp. $\begin{smallmatrix}\mathrm{G} & \cdot\\ \cdot & \mathrm{G}\end{smallmatrix}$) pattern. Assuming we know the block is demosaiced in one of the two $\begin{smallmatrix}\cdot & \mathrm{G}\\ \mathrm{G} & \cdot\end{smallmatrix}$ patterns, $\delta_{\begin{smallmatrix}\mathrm{R} & \mathrm{G}\\ \mathrm{G} & \mathrm{B}\end{smallmatrix}\,\vert\,\begin{smallmatrix}\mathrm{B} & \mathrm{G}\\ \mathrm{G} & \mathrm{R}\end{smallmatrix}}$ then distinguishes them; a value of 0 (resp. 1) means the block is demosaiced in the $\begin{smallmatrix}\mathrm{R} & \mathrm{G}\\ \mathrm{G} & \mathrm{B}\end{smallmatrix}$ (resp. $\begin{smallmatrix}\mathrm{B} & \mathrm{G}\\ \mathrm{G} & \mathrm{R}\end{smallmatrix}$) pattern. Similarly, a value of 0 (resp. 1) for $\delta_{\begin{smallmatrix}\mathrm{G} & \mathrm{R}\\ \mathrm{B} & \mathrm{G}\end{smallmatrix}\,\vert\,\begin{smallmatrix}\mathrm{G} & \mathrm{B}\\ \mathrm{R} & \mathrm{G}\end{smallmatrix}}$ means the block is demosaiced in the $\begin{smallmatrix}\mathrm{G} & \mathrm{R}\\ \mathrm{B} & \mathrm{G}\end{smallmatrix}$ (resp. $\begin{smallmatrix}\mathrm{G} & \mathrm{B}\\ \mathrm{R} & \mathrm{G}\end{smallmatrix}$) pattern, assuming we already know the block is demosaiced in a $\begin{smallmatrix}\mathrm{G} & \cdot\\ \cdot & \mathrm{G}\end{smallmatrix}$ pattern.

We can now determine the diagonal and pattern on each block. The detected diagonal of a block is $D \triangleq \begin{smallmatrix}\cdot & \mathrm{G}\\ \mathrm{G} & \cdot\end{smallmatrix}$ if $\delta_{\begin{smallmatrix}\cdot & \mathrm{G}\\ \mathrm{G} & \cdot\end{smallmatrix}\,\vert\,\begin{smallmatrix}\mathrm{G} & \cdot\\ \cdot & \mathrm{G}\end{smallmatrix}} = 0$, and $D \triangleq \begin{smallmatrix}\mathrm{G} & \cdot\\ \cdot & \mathrm{G}\end{smallmatrix}$ if $\delta_{\begin{smallmatrix}\cdot & \mathrm{G}\\ \mathrm{G} & \cdot\end{smallmatrix}\,\vert\,\begin{smallmatrix}\mathrm{G} & \cdot\\ \cdot & \mathrm{G}\end{smallmatrix}} = 1$.

Similarly, the full pattern is determined based on the diagonal and the corresponding δ:

$$P \triangleq \begin{cases} \begin{smallmatrix} R & G \\ G & B \end{smallmatrix} & \text{if } D = \begin{smallmatrix} \cdot & G \\ G & \cdot \end{smallmatrix} \quad \text{and } \delta_{\begin{smallmatrix} R & G \\ G & B \end{smallmatrix} \mid \begin{smallmatrix} B & G \\ G & R \end{smallmatrix}} = 0 \\ \begin{smallmatrix} B & G \\ G & R \end{smallmatrix} & \text{if } D = \begin{smallmatrix} \cdot & G \\ G & \cdot \end{smallmatrix} \quad \text{and } \delta_{\begin{smallmatrix} R & G \\ G & B \end{smallmatrix} \mid \begin{smallmatrix} B & G \\ G & R \end{smallmatrix}} = 1 \\ \begin{smallmatrix} G & R \\ B & G \end{smallmatrix} & \text{if } D = \begin{smallmatrix} G & \cdot \\ \cdot & G \end{smallmatrix} \quad \text{and } \delta_{\begin{smallmatrix} G & R \\ B & G \end{smallmatrix} \mid \begin{smallmatrix} G & B \\ R & G \end{smallmatrix}} = 0 \\ \begin{smallmatrix} G & B \\ R & G \end{smallmatrix} & \text{if } D = \begin{smallmatrix} G & \cdot \\ \cdot & G \end{smallmatrix} \quad \text{and } \delta_{\begin{smallmatrix} G & R \\ B & G \end{smallmatrix} \mid \begin{smallmatrix} G & B \\ R & G \end{smallmatrix}} = 1 \end{cases} \tag{7.1}$$

We now know the detected diagonal and pattern of each 2×2 block in the image. The pseudo-code of their computation is detailed in Algorithm 9. The detected diagonal and pattern of the whole image is then defined as the mode of the blocks' diagonals and patterns. Let $D_g \in \left\{ \begin{smallmatrix} \cdot & G \\ G & \cdot \end{smallmatrix}, \begin{smallmatrix} G & \cdot \\ \cdot & G \end{smallmatrix} \right\}$ and $P_g \in \left\{ \begin{smallmatrix} R & G \\ G & B \end{smallmatrix}, \begin{smallmatrix} B & G \\ G & R \end{smallmatrix}, \begin{smallmatrix} G & R \\ B & G \end{smallmatrix}, \begin{smallmatrix} G & B \\ R & G \end{smallmatrix} \right\}$ denote the global diagonal and pattern of the image.

A contrario estimation

Beyond internal learning, the second goal of this chapter is to allow for automatic detection of the forgeries. As already done in Chapter 4 (Linear Estimation of the Demosaicing Algorithm), we want to be able to detect regions where the detection is significantly erroneous, ie. where the network makes more mistakes than in the rest of the image.

The previous subsection explained how the network's output could be analysed to make each 2×2 block vote for the most likely grid locally. From there, we can directly apply the method described in Chapter 4 (Linear Estimation of the Demosaicing Algorithm). See this chapter for more details. To account for the correlation between pixels, the detection methods simulates downsampling of the block votes' counts by a given factor. Here, we use $d^2 \triangleq 17^2$, the radius of the CNN, as a downsampling factor.

Algorithm 9: Block votes computation

```
function get_block_votes(O_d, O_l)
    Input O_d: Diagonal of each pixel (output of the neural network), size (2X, 2Y).
    Input O_l: Line evenness of pixels on the main diagonal (output of the neural network), size (2X, 2Y).
    Output δ(G ·; · G): Detected diagonal of each 2 × 2 block, size (X, Y).
    Output δ(RG BG; GB GR): Detected pattern of each 2 × 2 block, if their diagonal is (G ·; · G), size (X, Y).
    Output δ(GR BG; RG BG): Detected pattern of each 2 × 2 block, if their diagonal is (· G; G ·), size (X, Y).
    δ(G ·; · G), δ(RG BG; GB GR), δ(GR BG; RG BG) ≔ 0
    for x from 0 to X and y from 0 to Y do
        if (1/4)(O_{d_{2x,2y}} + O_{d_{2x+1,2y+1}} − O_{d_{2x+1,2y}} − O_{d_{2x,2y+1}}) < 1/2 then
            δ(G ·; · G)_{x,y} ≔ 0
        else
            δ(G ·; · G)_{x,y} ≔ 1
        if (1/2)(O_{d_{2x,2y}} − O_{d_{2x+1,2y+1}}) < 1/2 then
            δ(RG BG; GB GR)_{x,y} ≔ 0
        else
            δ(RG BG; GB GR)_{x,y} ≔ 1
        if (1/2)(O_{d_{2x+1,2y}} − O_{d_{2x,2y+1}}) < 1/2 then
            δ(GR BG; RG BG)_{x,y} ≔ 0
        else
            δ(GR BG; RG BG)_{x,y} ≔ 1
        return δ(g ·; · g), δ(rg bg; bg gr), δ(gr bg; gb rg)
```

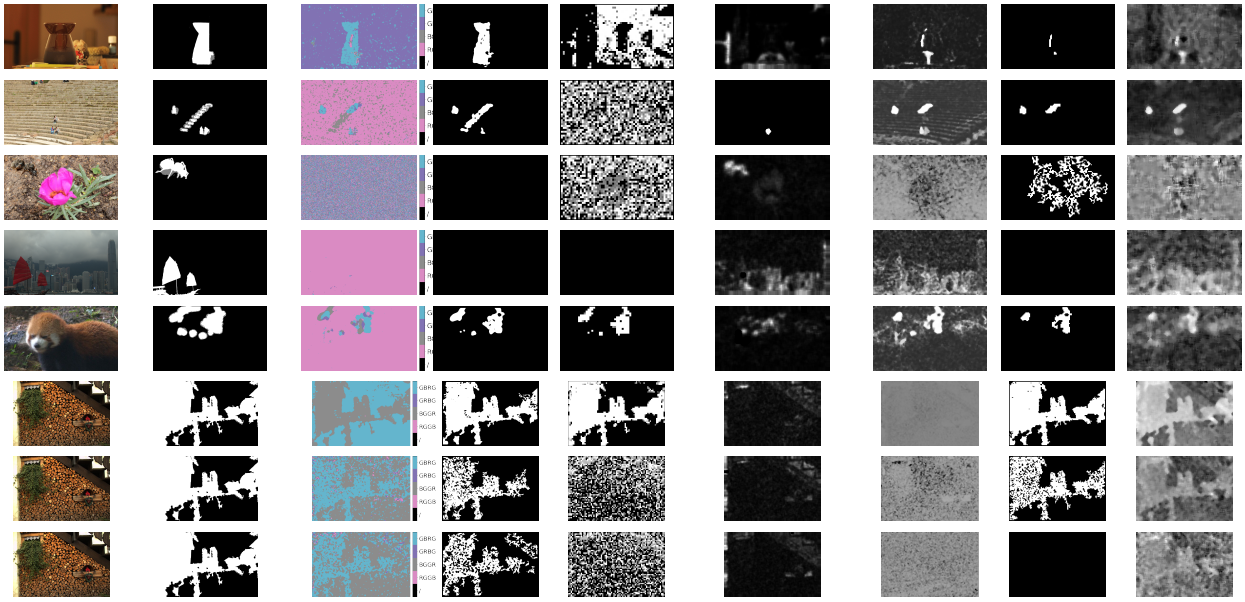


Figure 7.5: Comparative results on several images from the Korus [42], [43] (first 5 images) and Trace CFA Grid (last 3 images) databases. From left to right: Forged image, forgery mask, detected demosaicing grids with the proposed methods, detected forgeries with the proposed method, and results of Chapter 6 (Positional Learning for Demosaicing Analysis)'s method, Noiseprint [27], Ferrara [9], Park [16], and Chapter 3 (Intermediate Values Counting for CFA Pattern identification). As seen on the 1st, 2nd and 5th images, detecting the grid enables one not only to find forgeries, but also to precisely know how the forgery (here, an inpainting) was done, by localizing patches with different pattern alignments. On the 3rd image, no demosaicing traces are detected by any method. With our method, each block detects a different grid in a seemingly noise-like pattern. Still, the forgery can be detected by Noiseprint, which relies on different cues. On the 4th image, the same grid is detected on the whole image; the forged region cannot be detected by demosaicing analysis because the forged region's pattern is aligned. On the last three rows, the same image has been processed uncompressed and at JPEG compression qualities 95 and 90. Even after compression, we can still detect the forgery.

7.3 Experiments

To train our network, we used the 1488 raw images from the Dresden database [32], and demosaiced them with several demosaicing methods: AICC [76], RI [60], MLRI [74], ARI [61], CDMCNN [62], [128], CS [75], GBTF [59], Gunturk [56], HA [55], LMMSE [58] and bilinear demosaicing. Training was done with the Adam optimizer at an initial learning rate of 10^{-3} that was divided by 2 when the validation loss did not improve for 5 consecutive epochs. During internal training on each image, the network was retrained for 15 iterations.

Experiments were done on two datasets. The Korus [42], [43] database contains 220 splicing forgeries that are imperceptible to the naked eye. Those forgeries are done on images from 4 different cameras. From the Trace database, presented

in Chapter 1 (Non-Semantic Evaluation of Image Forensics Tools), we use the CFA grid and algorithm datasets, with the two different masks. As our method only detects pattern shifts, and not demosaicing algorithm changes, we cannot expect to make detections in regions where the demosaicing pattern is the same, ie. in one quarter of the images of the CFA algorithm dataset. However, since our method is not content-aware, results between the endomask and exomask should be similar. On the CFA grid dataset, we further studied the robustness of our method to JPEG compression, by compressing all images with quality factors of 95 and 90 before testing them with the forensic algorithms.

We compare our method with the state-of-the-art NN-based Noiseprint [27], with Chapter 6 (Positional Learning for Demosaicing Analysis)'s method, as well as demosaicing detection tools, Ferrara [9], Dirik [11], Park [16] Shin [15] and Chapter 3 (Intermediate Values Counting for CFA Pattern identification). Chapter 3 and Shin's methods operate on windows, we set their size to 32×32 pixels. For Park, the window size is set to 16×16 pixels, as suggested by the authors. For Choi, we use our implementation and extension, detailed in Chapter 3 (Intermediate Values Counting for CFA Pattern identification). For Shin, the original article only specifies the classification of the CFA pattern, but do not provide a way to turn that classification into a forgery detection. We use our own implementation for the grid detection, with the same scheme as Chapter 3 (Intermediate Values Counting for CFA Pattern identification).

Results are presented with the Matthews Correlation Coefficient (MCC), which is the cross-correlation coefficient between the ground truth and the detection. This metric, considered the most representative number for detection evaluation [81], [82], varies between -1 and 1, with 1 representing a perfect detection, -1 its complementary. Any random method has an expected score of 0. We computed the metric on each image and took the average of the mcc scores over the dataset.

As all the tested methods produce heatmaps and not binary outputs, the test was done using the best threshold over each dataset. Our method is designed to perform automatic detection: thresholding can be done directly with a NFA threshold of 1.

In Tab.7.1, we present compared results on the four dataset of the Trace database. Robustness of the methods to JPEG compression is tested in Tab. 7.3 on the Trace CFA Grid dataset, which is also used in Tab. 7.2 to show how our results vary depending on the algorithm used. Finally, in Tab. 7.4, we present compared results on the Korus [42], [43] dataset.

On the Trace datasets, the method beats the state-of-the-art at all the tested compression levels. The results are only slightly better than the previous chapter on the uncompressed images, but our method presents a stronger robustness to JPEG compression. Chapter 3 presents some robustness to compression as well. Both Bammey and Shin [15] are unable to make any detection even at a JPEG compression level of 95. Ferrara [9] and Park [16] do not provide good results on the grid datasets, but yield much better scores on the algorithm datasets. What these two methods detect is thus not as much the grid pattern as more generic demosaicing inconsistencies, thus suggesting some possible complementarity between our method and these two. Dirik [11] is unable to make any detection on the Trace datasets. The state-of-the-art forensic neural network Noiseprint [27] is entirely blind to demosa-

Method	Grid Exo	Grid Endo	Alg Exo	Alg Endo
Proposed	0.724	XXX	XXX	XXX
Chapter 6	0.682	0.665	0.501	0.491
Chapter 3	0.603	0.575	0.420	0.385
Shin[15]	0.104	0.099	0.085	0.084
Ferrara[9]	0.071	0.095	0.218	0.238
Dirik[11]	-0.002	0.010	0.001	0.013
Park[16]	0.116	0.133	0.152	0.150
Noiseprint[27]	-0.001	0.002	0.066	0.060

Table 7.1: Results on the CFA Grid (Grid) and CFA Algorithm (Alg) datasets of the Trace database, with endomasks (Endo) and exomasks (Exo)

	UNC	J95	J90
Bilinear	0.775	0.410	
AHD	0.741	0.287	0.150
AAHD	0.649	0.230	0.110
DCB	0.804	0.308	0.169
DHT	0.670	0.191	0.096
PPG	0.751	0.259	0.142
VNG	0.676	0.490	0.268

Table 7.2: Results of the proposed method on the CFA Grid dataset with exomasks of the Trace database, uncompressed (UNC) and at different compression levels (Jxx), depending on the demosaicing algorithm used in the image.

Method	UNC	J95	J90
Proposed	0.709	0.311	0.196
Chapter 6	0.692	0.005	0.003
Chapter 3	0.603	0.156	0.070
Shin[15]	0.104	0.001	0.001
Ferrara[9]	0.071	0.000	0.000
Dirik[11]	-0.002	0.000	0.001
Park[16]	0.116	0.001	0.000
Noiseprint[27]	-0.001	0.004	0.001

Table 7.3: Results on the CFA Grid dataset with exomasks of the Trace database, uncompressed and at two JPEG compression levels

icing pattern shifts, as seen in the Trace CFA Grid datasets. However, its results on the CFA Algorithm datasets mean that it is, to some extent, able to detect changes in the demosaicing algorithm used.

On the Korus dataset, our method presents the best results overall, however there is a high variability depending on the camera. Images from the Canon 60D dataset seem to present no demosaicing artefacts, as also evidenced by the other

Method	Canon 60D	Nikon D7000	Nikon D90	Sony α57	All
Proposed	0.000	**0.595**	**0.630**	**0.662**	**0.472**
Chapter 6	0.002	0.049	0.044	0.574	0.167
Chapter 3	0.004	0.176	0.251	0.251	0.238
Noiseprint	**0.153**	0.322	0.236	0.148	0.202
Shin	0.021	0.003	0.012	0.511	0.143
Ferrara	-0.016	0.498	0.461	0.339	0.321
Dirik	0.036	0.241	0.275	0.062	0.153
Park	0.018	0.540	0.491	0.302	0.338

Table 7.4: Results on the Korus dataset

demosaicing detection methods. This may be due to an absence of any demosaicing or to a downsampling of the images. These images indeed have a resolution lower than the camera's maximal resolution. On the two Nikon cameras, we get MCC scores of 0.412 (Nikon D7000) and 0.408 (Nikon D90). The best score is reached on the Sony α57 camera, where the MCC is 0.628. Most demosaicing detection methods perform best on that camera, which is probably due to the fact that images from this camera were demosaiced with a simple bilinear demosaicing. Ferrara and Park, on the other hand, yield better results on the two Nikon cameras. This is due to their methods being more suited to algorithms more modern than bilinear demosaicing. Noiseprint is the only method able to provide relevant detections on the Canon 60D cameras. Although its final score is lower than the proposed method's, we want to highlight that these two methods should be seen as complementary, not as competitors. As seen on the Trace database, Noiseprint is blind to shifts in the demosaicing pattern. In other words, its detections on the Korus datasets are based on other kinds of artefacts. On the other hand, the proposed method focuses solely on demosaicing pattern shifts. Both thus provide a different and complementary insight into potential forgeries.

Ferrara, Dirik and Park perform much better on the Korus dataset than on the Trace database. On the contrary, methods from Chapters 3 (Intermediate Values Counting for CFA Pattern identification) and 6 (Positional Learning for Demosaicing Analysis) yield better results on the Trace database. This can be easily explained: Ferrara, Dirik and Park look for demosaicing inconsistencies, but not necessarily for a specific CFA pattern, whereas Chapters 3 and 6 try to identify the CFA pattern locally. On the Trace datasets, focusing on finding the CFA pattern works best, because the pattern is consistent accross the forgery, furthermore it always lead to the detection in the Grid datasets, and 3 times out of 4 in the Algorithm datasets. On the Korus dataset, however, various means are used to create the forgery. In many images, inpainting is performed by pasting multiple small patches on an image. This leads to a forged region which is not consistent in its pattern, but rather made of many smaller subpatterns. Methods like Choi and Bammey, which try to find a single consistent pattern, are thus unable to do so. By trying to find regions where demosaicing pattern is less consistent than on the rest of the image, the proposed method is effectively able to perform well on both kind of forgeries.

However, specific detection of one single consistent pattern, like in our previous methods, may lead to more significant results on some forgeries.

Overall, this new method performs best in all Trace datasets, and on the Korus dataset, both globally and specifically on the three cameras from the Korus datasets that present demosaicing traces. This is despite the scoring method putting our method at a disadvantage, since the best threshold across the dataset is taken for other methods, whereas we use automatic thresholding for our method.

Visual results on different images can be seen on Fig. 7.5.

7.4 Discussion

In this chapter, we have shown that positional and internal learning could be coupled to detect image forgeries as shifts in the image's demosaicing pattern. The self-supervised nature of positional learning makes it fit for internal learning, as labels can be directly obtained from the tested image as if it were authentic. The main difficulty that could be expected with single-image fine-tuning comes from overfitting, especially on forged regions. While an overfitting network will correctly detect all pixels' locations, and will thus fail to detect the forgery, it will not lead to a higher number of forgery false positives. As we saw, the usually small size of forgeries, combined with the fact that forgeries create evidence that contradict the rest of the image, mean that overfitting is naturally limited. Positional learning may also be at play in preventing overfitting, by mapping similar outputs to opposite results. The role of positional learning against overfitting will be the subject of further studies.

Our experiments show that our method detects demosaicing pattern shifts better than other demosaicing detection methods, and more generally beats the state of the art on the Korus dataset of uncompressed forged images. *A contrario thresholding* enables us to automatically threshold the outputs, limiting the number of false detections and simplifying combination of this method with other methods.

The main limitation of our method, and of demosaicing detection in general, is that demosaicing artefacts are subtle and located on the high frequencies. As a consequence, a strong JPEG compression, or downsampling, will remove the artefacts and make demosaicing detection impossible. That being said, the proposed method provides enough robustness to yield decent results at a compression quality level of 95, and is still able to find a few forgeries with a compression factor of 90. This is not enough to perform detection on low-quality images such as those found on social medias. Nevertheless, the method works on the usually high JPEG quality provided by cameras. This makes it relevant in fields such as photographic contests, criminal investigations, scientific misconduct investigations, or journalism, tasks where image authentication is often needed.

Internal learning was performed with 15 iterations on each tested image. This number has been set heuristically, as results do not seem to improve much after these iterations. In future work, obtaining an automatic stopping criterion would be desirable, to increase both the speed and results of our method.

We observed that alterations of demosaicing patterns remain mostly undetected by most generic SOTA forensic algorithms, whereas our method specifically focuses

on those. This means that when trying to find forgeries, these methods yield complementary results and can thus work in parallel, not in competition.

Chapter 8

Conclusion

This short chapter concludes a thesis dedicated to image forgery detection through demosaicing analysis. In Chapter 1, we introduced a novel way to evaluate forgery detection methods. By locally modifying the formation pipeline of an image, we were able to create 'non-semantic forgeries', that contain changes in the underlying traces of the image without changing any of its semantic content. This methodology enables trace-aware evaluation of forensics tools, as it can highlight exactly to which traces each method is sensitive.

Chapters 2 (CFA Identification with Differential Operators) and 3 (Intermediate Values Counting for CFA Pattern identification) explored the use of direct numerical cues to distinguish sampled from interpolated pixels, and *in fine* identify the correct mosaic. Despite decent results with Chapter 3's method, the method systematically fails on several demosaicing algorithms. This shows the difficulty of devising one single method to analyse the large variety of demosaicing algorithms, as well as the need to break free from the unrealistic – but often difficult-to-avoid – assumption of a linear, channel-independent demosaicing.

Chapter 4 tried to increase adaptability to various algorithms by estimating the demosaicing directly on the image. An *a contrario* validation of the results was then used to filter out statistically-insignificant results and control the number of false alarms due to the acceptance of actually-insignificant detections. However, the linear estimation of demosaicing was not accurate enough in most cases.

In Chapter 5, we therefore explored a more natural hypothesis. Simply assuming that an image's demosaicing locally behaves closely to at least one of a set of other demosaicing algorithms, double demosaicing performed in the four positions with various algorithms was used to identify the most likely pattern. The theoretical foundations for this method are no longer valid when some post-processing, such as JPEG compression, takes place after the original demosaicing and alters the spatial correlation of the pixels. Still, the method is able to detect the correct pattern at the image level under relatively strong compression, even though local identification is no longer as reliable.

Chapter 6 introduces the novel concept of positional learning. We leveraged the translation-invariance of a CNN by training it to detect the modulo-2 position of each pixel. To do this, the CNN has to rely on the strong 2-periodic component of demosaicing traces; as such its output replicates the CFA mosaic, and more

importantly its shifts, that are important forgery cues. This self-supervised learning was further improved in Chapter 7, by using internal learning to adapt to each image and increase its robustness to JPEG compression. It is possible to retrain the CNN on a single potentially-forged image as if it were authentic; indeed potential forgeries send a contradictory response to the model: overfitting on those would usually imply going against the – usually much bigger – rest of the image. *A contrario* analysis then automatically detects significant inconsistencies in the CNN's output.

All in all, the main contributions of this thesis are four-fold:

- The *Trace methodology and database*, introduced in Chapter 1, enables asemantic analysis of forensic tools, to ensure they respond to traces in the image and not only to purely semantic incongruities; and to deduce specifically which traces each method is sensitive to. We obtain that way an easy estimation of each method's strong points, weaknesses and complementarities.

- *Positional learning*, introduced and used in Chapters 6 (Positional Learning for Demosaicing Analysis) and 7 (Internal Learning to Improve Adaptability), leverages the translation-invariance of CNN, replicates the behaviour of an underlying trace with a periodic component (in our case demosaicing), so as to detect its shifts.

- We also showed in Chapter 7 that *internal learning* can be used on a single, potentially forged image, to adapt to its specific processing, as potential forgeries are overpowered by the usually larger authentic region, which pulls the model in the opposite way.

- As part of the Envisu4 project, the method presented in Chapter 3 has also been integrated in the forensics browser plugin InVID & WeVerify[1], an online tool and plugin for journalists and fact-checkers to verify the authenticity of images and check for traces of tampering. More methods will be integrated as well.

Further works on the Trace database would include a systematic analysis of more camera traces and combinations thereof, in particular when further post-processing is applied to the whole image. In addition to evaluation, the proposed methodology could also be used to train images.

The ideas behind the final method in Chapter 7 seem stable enough. The achieved JPEG robustness makes the method applicable to a variety of high-quality images such as those usually found in criminal investigations, photographic contests, etc..... It is unlikely that robustness to a much stronger compression can be achieved. Results could be improved by a better localization of the exact forgery mask, an automatic stopping criterion during internal learning, and an optimization of the network structure. The latter could also improve the speed of the method, although speed is not a critical issue since typical use-cases for demosaicing analysis are offline.

[1] Beta version for Chrome at https://chrome.google.com/webstore/detail/fake-news-debunker-by-inv/mhccpoafgdgbhnjfhkcmgknndkeenfhe.

More generally, however, the notion of positional learning might find other applications. One can think, for instance, of JPEG analysis or texture synthesis. For better versatility, future work will also focus on the possibility of extending the concept to detect an unknown periodicity.

These few lines concludes this thesis. I want to thank one last time those who helped me through it (see page 3 for a non-exhaustive list), as well as the reader for your interest in my work.

And this gray spirit yearning in desire
To follow knowledge like a sinking star,
Beyond the utmost bound of human thought.

Ulysses, Alfred, Lord Tennyson

Bibliography

[1] M. A. Qureshi and M. Deriche, 'A review on copy move image forgery detection techniques', in *2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14)*, IEEE, 2014, pp. 1–5.

[2] C. Aguerrebere, J. Delon, Y. Gousseau and P. Musé, 'Study of the digital camera acquisition process and statistical modeling of the sensor raw data', Tech. Rep., Aug. 2013. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00733538.

[3] (), [Online]. Available: https://web.archive.org/web/20210414195050/https://news.samsung.com/global/samsungs-108mp-isocell-bright-hm1-delivers-brighter-ultra-high-res-images-with-industry-first-nonacell-technology (visited on 14/04/2021).

[4] O. Losson and E. Dinet, 'From the Sensor to Color Images', in *Digital Color - Acquisition, Perception, Coding and Rendering*, ser. Digital Image and Signal Processing series, C. Fernandez-Maloigne, F. Robert-Inacio and L. Macaire, Eds., Wiley, Mar. 2012, pp. 149–185. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00705825.

[5] G. Fechner, 'Elemente der psychophysik, breitkopf und härtel', *Leipzig: Breitkopf und Härtel*, 1860.

[6] A. C. Popescu and H. Farid, 'Exposing digital forgeries in color filter array interpolated images', *IEEE Transactions on Signal Processing*, vol. 53, no. 10, pp. 3948–3959, Oct. 2005, ISSN: 1053-587X. DOI: 10.1109/TSP.2005.855406.

[7] L. Liu, Y. Zhao, R. Ni and Q. Tian, 'Copy-move forgery localization using convolutional neural networks and cfa features', *Int. J. Digit. Crime For.*, vol. 10, no. 4, pp. 140–155, Oct. 2018, ISSN: 1941-6210. DOI: 10.4018/IJDCF.2018100110. [Online]. Available: https://doi.org/10.4018/IJDCF.2018100110.

[8] E. González Fernández, A. Sandoval Orozco, L. Garcia Villalba and J. Hernandez-Castro, 'Digital image tamper detection technique based on spectrum analysis of cfa artifacts', *Sensors*, vol. 18, no. 9, p. 2804, Aug. 2018, ISSN: 1424-8220. DOI: 10.3390/s18092804. [Online]. Available: http://dx.doi.org/10.3390/s18092804.

[9] P. Ferrara, T. Bianchi, A. De Rosa and A. Piva, 'Image forgery localization via fine-grained analysis of cfa artifacts', *IEEE TIFS*, vol. 7, no. 5, pp. 1566–1577, 2012.

[10] M. Kirchner, 'Efficient estimation of cfa pattern configuration in digital camera images.', in *Media Forensics and Security*, 2010, p. 754 111.

[11] A. E. Dirik and N. Memon, 'Image tamper detection based on demosaicing artifacts', in *ICIP*, IEEE, 2009, pp. 1497–1500.

[12] A. Swaminathan, M. Wu and K. R. Liu, 'Nonintrusive component forensics of visual sensors using output images', *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 1, pp. 91–106, 2007. DOI: 10.1109/TIFS.2006.890307.

[13] N. Le and F. Retraint, 'An improved algorithm for digital image authentication and forgery localization using demosaicing artifacts', *IEEE Access*, vol. 7, pp. 125 038–125 053, 2019. DOI: 10.1109/ACCESS.2019.2938467.

[14] C.-H. Choi, J.-H. Choi and H.-K. Lee, 'CFA pattern identification of digital cameras using intermediate value counting', in *MM&Sec*, ser. MM&Sec '11, Buffalo, New York, USA: ACM, 2011, pp. 21–26, ISBN: 978-1-4503-0806-9. DOI: 10.1145/2037252.2037258.

[15] H. J. Shin, J. J. Jeon and I. K. Eom, 'Color filter array pattern identification using variance of color difference image', *Journal of Electronic Imaging*, vol. 26, no. 4, pp. 1–12, 2017. DOI: 10.1117/1.JEI.26.4.043015. [Online]. Available: https://doi.org/10.1117/1.JEI.26.4.043015.

[16] C. W. Park, Y. H. Moon and I. K. Eom, 'Image tampering localization using demosaicing patterns and singular value based prediction residue', *IEEE Access*, vol. 9, pp. 91 921–91 933, 2021.

[17] B. Mahdian and S. Saic, 'Using noise inconsistencies for blind image forensics', *Image and Vision Computing*, vol. 27, pp. 1497–1503, Sep. 2009. DOI: 10.1016/j.imavis.2009.02.001.

[18] S. Lyu, X. Pan and X. Zhang, 'Exposing region splicing forgeries with blind local noise estimation', *International Journal of Computer Vision*, vol. 110, pp. 202–221, Nov. 2013. DOI: 10.1007/s11263-013-0688-y.

[19] M. Gardella, P. Musé, J.-M. Morel and M. Colom, 'Noisesniffer: A fully automatic image forgery detector based on noise analysis', in *2021 IEEE International Workshop on Biometrics and Forensics (IWBF)*, IEEE, 2021, pp. 1–6.

[20] W. Li, Y. Yuan and N. Yu, 'Passive detection of doctored jpeg image via block artifact grid extraction', *Signal Processing*, vol. 89, no. 9, pp. 1821–1829, 2009.

[21] C. Iakovidou, M. Zampoglou, S. Papadopoulos and Y. Kompatsiaris, 'Content-aware detection of jpeg grid inconsistencies for intuitive image forensics', *Journal of Visual Communication and Image Representation*, vol. 54, pp. 155–170, 2018.

[22] Z. Lin, J. He, X. Tang and C.-K. Tang, 'Fast, automatic and fine-grained tampered jpeg image detection via dct coefficient analysis', *Pattern Recognition*, vol. 42, no. 11, pp. 2492–2501, 2009.

[23] T. Bianchi, A. De Rosa and A. Piva, 'Improved dct coefficient analysis for forgery localization in jpeg images', in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2011, pp. 2444–2447.

[24] I. Amerini, R. Becarelli, R. Caldelli and A. Del Mastio, 'Splicing forgeries localization through the use of first digit features', in *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, 2014, pp. 143–148.

[25] T. Nikoukhah, J. Anger, T. Ehret, M. Colom, J.-M. Morel and R. Grompone von Gioi, 'Jpeg grid detection based on the number of dct zeros and its application to automatic and localized forgery detection', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 110–118.

[26] D. Cozzolino, G. Poggi and L. Verdoliva, 'Splicebuster: A new blind image splicing detector', in *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, IEEE, Nov. 2015, pp. 1–6. DOI: 10.1109/WIFS.2015.7368565.

[27] D. Cozzolino and L. Verdoliva, 'Noiseprint: A cnn-based camera model fingerprint', *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 144–159, 2020. DOI: 10.1109/TIFS.2019.2916364.

[28] M. Huh, A. Liu, A. Owens and A. A. Efros, 'Fighting fake news: Image splice detection via learned self-consistency', in *ECCV*, Sep. 2018, pp. 101–117.

[29] Jianbo Shi and J. Malik, 'Normalized cuts and image segmentation', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000. DOI: 10.1109/34.868688.

[30] Y. Wu, W. AbdAlmageed and P. Natarajan, 'Mantra-net: Manipulation tracing network for detection and localization of image forgeries with anomalous features', in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[31] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter and G. Boato, 'Raise: A raw images dataset for digital image forensics', in *Proceedings of the 6th ACM Multimedia Systems Conference*, 2015, pp. 219–224.

[32] T. Gloe and R. Böhme, 'The 'Dresden Image Database' for benchmarking digital image forensics', in *Proceedings of the 25th Symposium On Applied Computing (ACM SAC 2010)*, vol. 2, 2010, pp. 1585–1591.

[33] M. Colom, *Noise-free test images dataset*. [Online]. Available: http://mcolom.info/pages/no_noise_images/.

[34] G. Qian, Y. Wang, C. Dong *et al.*, 'Rethinking the pipeline of demosaicing, denoising and super-resolution', *arXiv preprint arXiv:1905.02538*, 2019.

[35] T.-T. Ng and S.-F. Chang, 'A data set of authentic and spliced image blocks', Columbia University, Tech. Rep., Jun. 2004.

[36] J. Dong, W. Wang and T. Tan, 'Casia image tampering detection evaluation database', in *2013 IEEE China Summit and International Conference on Signal and Information Processing*, Jul. 2013, pp. 422–426. DOI: 10.1109/ChinaSIP.2013.6625374.

[37] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo and G. Serra, 'A SIFT-based forensic method for copy-move attack detection and transformation recovery', *IEEE Trans. on Information Forensics and Security*, vol. 6, no. 3, pp. 1099–1110, Sep. 2011. DOI: 10.1109/TIFS.2011.2129512. [Online]. Available: http://www.lambertoballan.net/downloads/2011%5C%5Ftifs%5C%5Fpreprint.pdf.

[38] V. Christlein, C. Riess, J. Jordan, C. Riess and E. Angelopoulou, 'An evaluation of popular copy-move forgery detection approaches', *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1841–1854, Dec. 2012. DOI: 10.1109/TIFS.2012.2218597.

[39] D. Tralic, I. Zupancic, S. Grgic and M. Grgic, 'Comofod – new database for copy-move forgery detection', in *Proceedings ELMAR-2013*, Sep. 2013, pp. 49–54.

[40] B. Wen, Y. Zhu, R. Subramanian, T. Ng, X. Shen and S. Winkler, 'Coverage – a novel database for copy-move forgery detection', in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 161–165. DOI: 10.1109/ICIP.2016.7532339.

[41] H. Guan, M. Kozak, E. Robertson *et al.*, 'Mfc datasets: Large-scale benchmark datasets for media forensic challenge evaluation', in *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, 2019, pp. 63–72. DOI: 10.1109/WACVW.2019.00018.

[42] P. Korus and J. Huang, 'Evaluation of random field models in multi-modal unsupervised tampering localization', in *Proc. of IEEE Int. Workshop on Inf. Forensics and Security*, 2016.

[43] P. Korus and J. Huang, 'Multi-scale analysis strategies in prnu-based tampering localization', *IEEE Trans. on Information Forensics and Security*, 2017.

[44] G. Mahfoudi, B. Tajini, F. Retraint, F. Morain-Nicolier, J.-L. Dugelay and M. Pic, 'Defacto: Image and face manipulation dataset', in *27th European Signal Processing Conference (EUSIPCO 2019)*, A Coruña, Spain, Sep. 2019.

[45] T.-Y. Lin, M. Maire, S. Belongie *et al.*, *Microsoft coco: Common objects in context*, 2015. arXiv: 1405.0312 [cs.CV].

[46] A. Desolneux, L. Moisan and J.-M. Morel, 'Meaningful alignments', *International journal of computer vision*, vol. 40, no. 1, pp. 7–23, 2000.

[47] B. Brecht, *Life of Galileo*. Bloomsbury, 2015, Translated by John Willet.

[48] Q. **Bammey**, R. Grompone von Gioi and J.-M. Morel, 'Automatic detection of demosaicing image artifacts and its use in tampering detection', in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, See Chapter 4, 2018, pp. 424–429. DOI: 10.1109/MIPR.2018.00091.

[49] Q. **Bammey**, R. Grompone von Gioi and J.-M. Morel, 'Reliable demosaicing detection for image forensics', in *2019 27th European Signal Processing Conference (EUSIPCO)*, See Chapter 2, 2019, pp. 1–5. DOI: 10.23919/EUSIPCO.2019.8903152.

[50] Q. **Bammey**, R. G. v. Gioi and J.-M. Morel, 'An adaptive neural network for unsupervised mosaic consistency analysis in image forensics', in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, See Chapter 6, Jun. 2020.

[51] Q. **Bammey**, R. Grompone von Gioi and J.-M. Morel, 'Image Forgeries Detection through Mosaic Analysis: the Intermediate Values Algorithm', *Image Processing On Line*, vol. 11, pp. 317–343, 2021. DOI: 10.5201/ipol.2021.355.

[52] Q. **Bammey**, R. G. von Gioi and J.-M. Morel, 'Forgery detection by internal positional learning of demosaicing traces', in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2022, pp. 328–338.

[53] Q. **Bammey**, T. Nikoukhah, M. Gardella, R. G. von Gioi, M. Colom and J.-M. Morel, 'Non-semantic evaluation of image forensics tools: Methodology and database', in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2022, pp. 3751–3760.

[54] Q. **Bammey**, M. Colom, M. Gardella *et al.*, 'Sécurité multimédia', in ISTE, Jul. 2021, vol. 1, ch. Comment reconstruire l'histoire d'une image digitale, et de ses altérations ?, pp. 9–50, Directed by William Puech. English version in preparation. DOI: 10.51926/ISTE.9026.ch1.

[55] J. F. Hamilton Jr and J. E. Adams Jr, *Adaptive color plan interpolation in single sensor color electronic camera*, US Patent 5,629,734, May 1997.

[56] B. K. Gunturk, Y. Altunbasak and R. M. Mersereau, 'Color plane interpolation using alternating projections', *IEEE TIP*, vol. 11, no. 9, pp. 997–1013, Sep. 2002, ISSN: 1057-7149. DOI: 10.1109/TIP.2002.801121.

[57] A. Buades, B. Coll, J. M. Morel and C. Sbert, 'Self-similarity Driven Demosaicking', *Image Processing On Line*, vol. 1, pp. 51–56, 2011. DOI: 10.5201/ipol.2011.bcms-ssdd.

[58] P. Getreuer, 'Zhang-Wu Directional LMMSE Image Demosaicking', *Image Processing On Line*, vol. 1, pp. 117–126, 2011. DOI: 10.5201/ipol.2011.g_zwld.

[59] I. Pekkucuksen and Y. Altunbasak, 'Gradient based threshold free color filter array interpolation', in *2010 IEEE International Conference on Image Processing*, IEEE, 2010, pp. 137–140.

[60] D. Kiku, Y. Monno, M. Tanaka and M. Okutomi, 'Residual interpolation for color image demosaicking', in *2013 IEEE International Conference on Image Processing*, IEEE, 2013, pp. 2304–2308.

[61] Y. Monno, D. Kiku, M. Tanaka and M. Okutomi, 'Adaptive residual interpolation for color image demosaicking', in *2015 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2015, pp. 3861–3865.

[62] R. Tan, K. Zhang, W. Zuo and L. Zhang, 'Color image demosaicking via deep residual learning', in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2017, pp. 793–798.

[63] M. Gharbi, G. Chaurasia, S. Paris and F. Durand, 'Deep joint demosaicking and denoising', *ACM Trans. Graph.*, vol. 35, no. 6, 191:1–191:12, Nov. 2016, ISSN: 0730-0301. DOI: 10.1145/2980179.2982399.

[64] C.-H. Choi, H.-Y. Lee and H.-K. Lee, 'Estimation of color modification in digital images by cfa pattern change', *Forensic Science International*, vol. 226, no. 1, pp. 94–105, 2013.

[65] H. Farid, *Photo Forensics*. The MIT Press, 2016.

[66] L. Zheng, Y. Zhang and V. Thing, 'A survey on image tampering and its detection in real-world photos', *Journal of Visual Communication and Image Representation*, Dec. 2018. DOI: 10.1016/j.jvcir.2018.12.022.

[67] C.-C. Hsu, T.-Y. Hung, C.-W. Lin and C.-T. Hsu, 'Video forgery detection using correlation of noise residue', in *2008 IEEE 10th workshop on multimedia signal processing*, IEEE, 2008, pp. 170–174.

[68] M. Castro, D. M. Ballesteros and D. Renza, 'A dataset of 1050-tampered color and grayscale images (cg-1050)', *Data in Brief*, vol. 28, p. 104 864, 2020, ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2019.104864. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352340919312193.

[69] P. Zhou, X. Han, V. I. Morariu and L. S. Davis, 'Two-stream neural networks for tampered face detection', in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017, pp. 1831–1839. DOI: 10.1109/CVPRW.2017.229.

[70] M. Delbracio, D. Kelly, M. S. Brown and P. Milanfar, *Mobile computational photography: A tour*, 2021. arXiv: 2102.09000 [cs.CV].

[71] A. Foi, M. Trimeche, V. Katkovnik and K. Egiazarian, 'Practical poissonian-gaussian noise modeling and fitting for single-image raw-data', *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, vol. 17, pp. 1737–54, Nov. 2008. DOI: 10.1109/TIP.2008.2001399.

[72] G. K. Wallace, 'The jpeg still picture compression standard', *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[73] B. Wen, Y. Zhu, R. Subramanian, T.-T. Ng, X. Shen and S. Winkler, 'Coverage – a novel database for copy-move forgery detection', in *2016 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2016, pp. 161–165.

[74] D. Kiku, Y. Monno, M. Tanaka and M. Okutomi, 'Minimized-laplacian residual interpolation for color image demosaicking', in *Digital Photography X*, International Society for Optics and Photonics, vol. 9023, 2014, p. 90230L.

[75] P. Getreuer, 'Image Demosaicking with Contour Stencils', *IPOL*, vol. 2, pp. 22–34, 2012, https://doi.org/10.5201/ipol.2012.g-dwcs. DOI: 10.5201/ipol.2012.g-dwcs.

[76] J. Duran and A. Buades, 'A Demosaicking Algorithm with Adaptive Inter-Channel Correlation', *IPOL*, vol. 5, no. 9, pp. 311–327, Sep. 2015, ISSN: 1057-7149. DOI: 10.1109/TIP.2014.2341928.

[77] H. Zhang, K. Dana, J. Shi *et al.*, 'Context encoding for semantic segmentation', in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[78] *Libraw library, copyright © 2008-2019 libraw llc, https://www.libraw.org.*

[79] M. Zampoglou, S. Papadopoulos and Y. Kompatsiaris, 'Large-scale evaluation of splicing localization algorithms for web images', *Multimedia Tools and Applications*, vol. 76, no. 4, pp. 4801–4834, 2017.

[80] B. Matthews, 'Comparison of the predicted and observed secondary structure of t4 phage lysozyme', *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975, ISSN: 0005-2795. DOI: https://doi.org/10.1016/0005-2795(75)90109-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005279575901099.

[81] D. Chicco, 'Ten quick tips for machine learning in computational biology', eng, *BioData mining*, vol. 10, pp. 35–35, Dec. 2017, ISSN: 1756-0381. DOI: 10.1186/s13040-017-0155-3. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/29234465.

[82] D. Chicco and G. Jurman, 'The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation', eng, *BMC genomics*, vol. 21, no. 1, pp. 6–6, Jan. 2020, ISSN: 1471-2164. DOI: 10.1186/s12864-019-6413-7. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/31898477.

[83] A. Desolneux, L. Moisan and J. Morel, *From gestalt theory to image analysis. interdisciplinary applied mathematics, vol. 35*, Oct. 2007. DOI: 10.1023/A:1026593302236. [Online]. Available: https://doi.org/10.1023/A:1026593302236.

[84] A. Desolneux, L. Moisan and J.-M. Morel, *From Gestalt Theory to Image Analysis: A Probabilistic Approach*. Springer, 2008.

[85] A. Desolneux, L. Moisan and J.-M. Morel, *From Gestalt Theory to Image Analysis*. Springer, 2008.

[86] A. Gordon, G. Glazko, X. Qiu and A. Yakovlev, 'Control of the mean number of false discoveries, Bonferroni and stability of multiple testing', *The Annals of Applied Statistics*, vol. 1, no. 1, pp. 179–190, 2007.

[87] H. B. Mann and D. R. Whitney, 'On a test of whether one of two random variables is stochastically larger than the other', *The annals of mathematical statistics*, pp. 50–60, 1947.

[88] M. Kirchner, 'Efficient estimation of CFA pattern configuration in digital camera images', in *Media Forensics and Security*, 2010.

[89] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias *et al.*, 'Scikit-image: Image processing in Python', *PeerJ*, vol. 2, e453, Jun. 2014, ISSN: 2167-8359. DOI: 10.7717/peerj.453. [Online]. Available: https://doi.org/10.7717/peerj.453.

[90] A. Clark, *Pillow (pil fork) documentation*, 2015. [Online]. Available: https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf.

[91] M. Kirchner and J. Fridrich, 'On detection of median filtering in digital images', in *Media forensics and security II*, International Society for Optics and Photonics, vol. 7541, 2010, p. 754 110.

[92] K. Hirakawa and T. W. Parks, 'Adaptive homogeneity-directed demosaicing algorithm', *IEEE Transactions on Image Processing*, vol. 14, no. 3, pp. 360–369, 2005.

[93] C.-k. Lin, Archived from the original page on 23rd September 2016, 2010. [Online]. Available: https://web.archive.org/web/20160923211135/https://sites.google.com/site/chklin/demosaic/.

[94] M. Kirchner and R. Böhme, 'Synthesis of color filter array pattern in digital images.', *Media Forensics and Security*, vol. 7254, p. 72 540, 2009.

[95] D. C. Jeronymo, Y. C. C. Borges and L. dos Santos Coelho, 'Image forgery detection by semi-automatic wavelet soft-thresholding with error level analysis', *Expert Systems with Applications*, vol. 85, pp. 348–356, 2017.

[96] Y. Ke, Q. Zhang, W. Min and S. Zhang, 'Detecting image forgery based on noise estimation', *International Journal of Multimedia and Ubiquitous Engineering*, vol. 9, no. 1, pp. 325–336, 2014.

[97] A. C. Popescu and H. Farid, 'Statistical tools for digital forensics', in *Information Hiding*, Springer, 2004, pp. 128–147.

[98] A. Ghosh, Z. Zhong, T. E Boult and M. Singh, 'Spliceradar: A learned method for blind image forensics', in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Jun. 2019.

[99] O. Mayer, B. Bayar and M. C. Stamm, 'Learning unified deep-features for multiple forensic tasks', in *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, ACM, 2018, pp. 79–84.

[100] O. Mayer and M. C. Stamm, 'Learned forensic source similarity for unknown camera models', in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 2012–2016.

[101] B. Bayar and M. C. Stamm, 'Constrained convolutional neural networks: A new approach towards general purpose image manipulation detection', *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2691–2706, 2018.

[102] O. Mayer and M. C. Stamm, 'Forensic similarity for digital images', *IEEE Transactions on Information Forensics and Security*, 2019, ISSN: 1556-6013. DOI: 10.1109/TIFS.2019.2924552.

[103] P. Korus, 'Digital image integrity–a survey of protection and verification techniques', *Digital Signal Processing*, vol. 71, pp. 1–26, 2017.

[104] S. Walia and M. Kaur, 'Forgery detection using noise inconsistency: A review', *International Journal of Computer Science and Information Technologies*, vol. 5, no. 6, pp. 7618–7622, 2014.

[105] T. Julliand, V. Nozick and H. Talbot, 'Automated image splicing detection from noise estimation in raw images', 2015.

[106] C. Destruel, V. Itier, O. Strauss and W. Puech, 'Color noise-based feature for splicing detection and localization', in *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, IEEE, 2018, pp. 1–6.

[107] X. Pan, X. Zhang and S. Lyu, 'Exposing image splicing with inconsistent local noise variances', in *2012 IEEE International Conference on Computational Photography (ICCP)*, IEEE, 2012, pp. 1–10.

[108] B. Liu and C.-M. Pun, 'Splicing forgery exposure in digital image by detecting noise discrepancies', *International Journal of Computer and Communication Engineering*, vol. 4, no. 1, p. 33, 2015.

[109] H. Zeng, Y. Zhan, X. Kang and X. Lin, 'Image splicing localization using pca-based noise level estimation', *Multimedia Tools and Applications*, vol. 76, no. 4, pp. 4783–4799, 2017.

[110] H. Yao, S. Wang, X. Zhang, C. Qin and J. Wang, 'Detecting image splicing based on noise level inconsistency', *Multimedia Tools and Applications*, vol. 76, no. 10, pp. 12 457–12 479, 2017.

[111] N. D. Wandji, S. Xingming and M. F. Kue, 'Detection of copy-move forgery in digital images based on dct', *arXiv preprint arXiv:1308.5661*, 2013.

[112] Y. Wu, W. Abd-Almageed and P. Natarajan, 'Busternet: Detecting copy-move image forgery with source/target localization', in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 168–184.

[113] E. Silva, T. Carvalho, A. Ferreira and A. Rocha, 'Going deeper into copy-move forgery detection: Exploring image telltales via multi-scale analysis and voting processes', *Journal of Visual Communication and Image Representation*, vol. 29, no. 0, pp. 16–32, 2015.

[114] A. Ferreira, S. C. Felipussi, C. Alfaro *et al.*, 'Behavior knowledge space-based fusion for copy–move forgery detection', *IEEE Transactions on Image Processing*, vol. 25, no. 10, pp. 4729–4742, 2016.

[115] J. H. Bappy, A. K. Roy-Chowdhury, J. Bunk, L. Nataraj and B. Manjunath, 'Exploiting spatial structure for localizing manipulated image regions', in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4970–4979.

[116] J. Bunk, J. H. Bappy, T. M. Mohammed *et al.*, 'Detection and localization of image forgeries using resampling features and deep learning', in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, 2017, pp. 1881–1889.

[117] J. H. Bappy, C. Simons, L. Nataraj, B. Manjunath and A. K. Roy-Chowdhury, 'Hybrid lstm and encoder-decoder architecture for detection of image forgeries', *IEEE Transactions on Image Processing*, 2019.

[118] C. Szegedy, W. Liu, Y. Jia *et al.*, 'Going deeper with convolutions', in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[119] A. L. Maas, 'Rectifier nonlinearities improve neural network acoustic models', 2013.

[120] F. Chollet, 'Xception: Deep learning with depthwise separable convolutions', in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 1800–1807. DOI: 10.1109/CVPR.2017.195.

[121] A. G. Howard, M. Zhu, B. Chen *et al.*, 'Mobilenets: Efficient convolutional neural networks for mobile vision applications', *CoRR*, vol. abs/1704.04861, 2017. arXiv: 1704.04861. [Online]. Available: http://arxiv.org/abs/1704.04861.

[122] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau and R. Garcia, 'Incorporating second-order functional knowledge for better option pricing.', Jan. 2000, pp. 472–478.

[123] D. P. Kingma and J. Ba, 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980*, 2014.

[124] A. Krull, T.-O. Buchholz and F. Jug, 'Noise2void-learning denoising from single noisy images', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2129–2137.

[125] Y. Quan, M. Chen, T. Pang and H. Ji, 'Self2self with dropout: Learning self-supervised denoising from single image', in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[126] K. Zhang, W. Zuo, Y. Chen, D. Meng and L. Zhang, 'Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising', *IEEE transactions on image processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[127] L. Wright and N. Demeure, 'Ranger21: A synergistic deep learning optimizer', *arXiv preprint arXiv:2106.13731*, 2021.

[128] T. Ehret and G. Facciolo, 'A Study of Two CNN Demosaicking Algorithms', *Image Processing On Line*, vol. 9, pp. 220–230, 2019. DOI: 10.5201/ipol.2019.274.

université PARIS-SACLAY | ÉCOLE DOCTORALE de mathématiques Hadamard (EDMH)

Titre: Détection de falsification d'images via l'analyse du démosaïquage : dévoilement d'une signature

Mots clés: Forensiques d'image, apprentissage positionnel, apprentissage auto-supervisé, apprentissage interne, démosaïquage

Résumé: Autrefois considérées comme des preuves fiables, les images photographiques ne dépeignent plus toujours la pure vérité. Avec l'avènement de la photographie numérique et les progrès des outils de retouche photo, il n'a jamais été aussi facile de modifier une image. Si la plupart de ces modifications visent uniquement à améliorer l'image, elles peuvent potentiellement en altérer la sémantique même. Dissimuler, modifier ou ajouter un objet étranger, tout cela peut donner à une image un sens nouveau et trompeur. Bien que ces falsifications puissent facilement être rendues visuellement réalistes, ils n'en altèrent pas moins le tissu même de l'image. La formation d'une image numérique, depuis les capteurs de la caméra jusqu'au stockage, laisse des traces, qui agissent comme une signature de l'image. La modification d'une image déforme ces traces, créant des incohérences détectables.

Les images brutes sont initialement une mosaïque de pixels rouges, bleus et verts. Les valeurs de couleur manquantes doivent être interpolées dans un processus connu sous le nom de démosaïquage. Dans cette thèse, nous étudions les traces laissées par ce processus. La nature 2-périodique du motif de la mosaïque laisse son empreinte sur l'image. Les falsifications peuvent déphaser ces traces, voire les supprimer entièrement ; l'identification du motif de mosaïque est donc utile pour localiser les régions falsifiées.

Les méthodes non spécifiques de détection des falsifications peuvent déjà analyser de nombreuses traces dans une image ; elles restent néanmoins aveugles aux déplacements de la mosaïque, en raison de l'invariance par translation des réseaux de neurones convolutifs sur lesquels la plupart sont basés. Les méthodes spécifiques au démosaïquage peuvent donc fournir des résultats complémentaires pour la détection des falsifications. Cependant, elles ont historiquement reçu peu d'attention. L'analyse des artefacts de démosaïquage est rendue plus difficile par la vaste gamme d'algorithmes de démosaïquage, souvent non divulgués, et surtout par la compression JPEG. Ces artefacts, créés tôt dans le pipeline de formation de l'image et situés aux fréquences les plus élevées de l'image, s'estompent rapidement pendant la compression.

Pourtant, ces artefacts peuvent encore être détectés sous une compression légère. Pour canaliser la puissance représentative des réseaux neuronaux convolutifs dans l'analyse des artefacts de démosaïquage, nous introduisons la notion d'apprentissage positionnel. Ce schéma auto-supervisé entraîne le réseau à détecter la position modulo 2 de chaque pixel, en tirant parti de l'invariance de translation de la convolution pour que le réseau analyse implicitement les artefacts de démosaïquage, son seul indice de la position modulo 2 d'un pixel. De plus, l'entraînement interne sur une seule image potentiellement falsifiée peut renforcer la robustesse de la méthode à la compression JPEG de ladite image. Les erreurs dans la sortie du réseau neuronal sont alors des indices d'incohérences de la mosaïque. Un paradigme *a contrario* nous permet alors de prendre des décisions automatiques sur l'authenticité d'une image. En utilisant uniquement les artefacts de démosaïquage, la méthode proposée dépasse l'état de l'art sur plusieurs jeux de données non compressés. Sur les images compressées, elle fournit encore des résultats décents qui sont tout à fait complémentaires avec les méthodes qui ne sont pas spécifiques à la mosaïque.

Enfin, nous explorons l'évaluation même des méthodes de détection de falsification. Nous proposons une méthodologie et un jeu de données pour étudier la sensibilité des outils de détection à des traces spécifiques, ainsi que leur capacité à effectuer des détections sans indices sémantiques sur l'image. Plus qu'un simple outil d'évaluation, cette méthodologie peut être utilisée pour évaluer les forces et faiblesses de chaque méthode, ainsi que leurs complémentarités.

Title: Image Forgery Detection through Demosaicing Analysis: Unconcealment of a Signature

Keywords: Image Forensics, Positional Learning, Self-Supervised Learning, Internal Learning, Demosaicing

Abstract: Once considered reliable evidence, photographic images can no longer be assumed to depict the naked truth. With the advent of digital photography and the progress of photo editing tools, altering a picture has never been easier. While most of these modifications solely seek to enhance the image, they can potentially alter its very semantics. Concealing, modifying or adding a foreign object, all those can give an image a new and false meaning. Although these forgeries can easily be made visually realistic, they still distort the very fabric of the image. The formation of a digital image, from the camera sensors to storage, leaves traces, which act like a signature for the image. Modifying an image distorts these traces, creating detectable inconsistencies.

Raw images are initially a mosaic of red, blue and green pixels. Missing colour values must be interpolated in a process known as demosaicing. In this thesis, we study the traces left by this process. The 2-periodic nature of the mosaic pattern leaves its imprint onto the image. Forgeries may dephase these traces, or even remove them entirely; mosaic pattern identification is consequently helpful in localizing tampered regions.

Non-specific forgery detection methods can already analyse many traces in an image; nevertheless they remain blind to shifts in the mosaic, due to the translation-invariance of the convolutional neural networks on which most are based. Demosaicing-specific methods can thus provide complementary results for forgery detections. However, these have historically received little attention. Analysis of demosaicing artefacts is made harder by the vast array of often-undisclosed demosaicing algorithms, and above all by JPEG compression. Those artefacts, created early in the image formation pipeline and lying at the highest frequencies of the image, are quick to wane during compression.

Yet, those artefacts can still be detected under mild compression. To channel the representative power of convolutional neural networks into the analysis of demosaicing artefacts, we introduce the notion of positional training. This self-supervised scheme trains the network to detect the modulo-2 position of each pixel, leveraging the translation invariance of convolution to make the network implicitly analyse demosaicing artefacts, its only clue to the modulo-2 position of a pixel. On top of that, internal training on a single potentially forged image can bolster the method's robustness to JPEG compression on said image. Errors in the output of the neural network are then clues of mosaic inconsistencies. An *a contrario* paradigm then enables us to make automatic decisions on the authenticity of an image. Using only demosaicing artefacts, the proposed method beats the state of the art on several uncompressed datasets. On compressed images, it still provides decent results that are fully complementary with methods that are not mosaic-specific.

Finally, we explore the very evaluation of forgery detection methods. We propose a methodology and dataset to study the sensitivity of forensic tools to specific traces, as well as their ability to make detections without semantic cues on the image. More than a simple evaluation tool, this methodology can be used to assess the strength and weaknesses of each method, as well as their complementarities.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France