PLUG-AND-PLAY ADMM FOR IMAGE RESTORATION

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Xiran Wang

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2019

Purdue University

West Lafayette, Indiana

ACKNOWLEDGMENTS

I thank my advisor, Prof. Stanley Chan. I worked closely with him over the past few years over a range of interesting research topics and he has been a wonderful mentor. Discussion with Prof. Chan has been very beneficial and crucial for my research. I am grateful to Prof. Allebach, Prof. Bouman and Prof. Reibman for serving on my committee. Prof. Allebach has been always supportive on all the matters along my PhD journey, and I am truly grateful for the opportunity to work with him for part of the journey. My research has largely been inspired by Prof. Bouman and Prof. Buzzard's work to whom I am sincerely grateful. I thank Prof. Reibman for being supportive and her feedback on my research which is important in finalizing my research.

I want to thank my friends and labmates who have been supportive along the journey. Finally, I want to thank my parents. This journey would not have been possible without the support from my family both financially and emotionally. Thank you for believing in me.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

P&P	plug-and-play
AMP	approximate message passing
ADMM	alternating direction method of multipliers
PAMP	parameter free approximate message passing
MACE	multi-agent consensus equilibrium
VR	virtual reality
SpaRSA	sparse reconstruction by separable approximation
FISTA	fast iterative shrinkage thresholding algorithm
TwIST	two-step iterative shrinkage thresholding
GAMP	generalized approximate message passing

ABSTRACT

Wang, Xiran Ph.D., Purdue University, August 2019. Plug-and-Play ADMM for Image Restoration. Major Professor: Stanley H. Chan.

The alternating direction method of multiplier (ADMM) is one of the most widely used optimization algorithms in image restoration. Among many features, e.g., provably convergent under mild conditions, its modular structure is particularly appealing to model based image reconstruction problems. In particular, one can separate the log-likelihood and the log-prior in a maximum-a-posteriori formulation using the ADMM algorithm. However, such approach does not allow us to incorporate likelihood or priors that are not expressible as proximal maps (a particular type of optimization involving a convex function regularized by a quadratic penalty). Deep neural network based image denoisers are some of the better known examples. They have demonstrated very promising image restoration results, yet they cannot be expressed as proximal maps. The question to pursue in this thesis is how to integrate non-optimization likelihood or priors using the ADMM algorithm.

The Plug-and-Play (P&P) ADMM is a generalization of the ADMM algorithm that allows non-optimization based models to be used. Since its introduction in 2013, the method has demonstrated promising performance for various imaging problems, e.g., tomography. Its convergence has also been proven under some restrictive conditions where the denoiser is non-expansive and has a symmetric Jacobian. However, many problems remain unsolved. First, existing work has been focusing on a range of medical imaging problems in tomography while the applications of the Plug-and-Play framework for more general problems have not been explored. Second, even though study on the global convergence analysis has been done, it restricts the compatible denoisers to a class of symmetric smoothing filters that are rarely adopted in state-of-the-art competitive denoisers. Third, due to its ad-hoc nature, the performance of Plug-and-Play ADMM is sensitive to the choice of internal parameters of the algorithm. A more robust version is desirable in order for the framework to be applicable for a wide range of problems. Fourth, the current Plug-and-Play framework still relies on a well-defined forward model of the imaging system, which means it is still an optimization based approach. However, as well-defined models are sometimes unavailable for more complicated imaging problems, a non-optimization based version is desired.

In this thesis, we address the above issues by studying both the theoretical and practical aspects of the algorithm. First, we study the applications of the Plug-and-Play framework for a wide range of general image restoration problems, such as super-resolution, deblurring, inpainting, single-photon imaging and even a video segmentation problem used for virtual reality content creation. Efficient implementations of the Plug-and-Play ADMM for these applications are introduced and outperforms state-of-the-art existing algorithms for every task. For superresolution specifically, we derived a closed-form solution for the inversion step that is previously unavailable and potentially applicable for other optimization frameworks. Second, to tackle the Plug-and-Play ADMM's sensitivity on internal parameters, we draw insights from the generalized approximate message passing to design an automatic update scheme for the internal parameters achieving robust performance across different tasks. Third, a new convergence analysis is presented proving a fixed-point convergence for a much wider range of denoisers compared to previous work. With the recent introduction of Multi-agent consensus equilibrium (MACE), a generalized Plug-and-Play framework that can work with an arbitrary number of operators to solve a common problem, this work also introduces the application and design of a MACE algorithm for solving video segmentation which is outside the scope of classical image restoration problems. The proposed MACE algorithm, unlike Plug-and-Play ADMM, does not rely on a well-defined forward model and is capable of including an arbitrary number of operators instead of just two.

The Plug-and-Play ADMM algorithms studied in this thesis have significantly advanced our image restoration capability by allowing non-optimization procedures to be used in the framework. We demonstrated applications in super-resolution, inpainting, deblurring, and single-photon reconstruction, with superior performance than the previous state-of-the-art. The algorithm has also enabled a new line of applications in segmenting foreground masks for virtual reality content creation that is fully automatic and does not require human interactions.

PUBLICATIONS

- **Xiran Wang**, Jason Juang and Stanley H. Chan, "Foreground Extraction for Virtual Reality Filming using Multi-Agent Consensus Equilibrium", submitted to IEEE Trans. Image Processing.

- **Xiran Wang**, Stanley H. Chan, "Parameter-free Plug-and-Play ADMM for image restoration", IEEE ICASSP, pp.1323-1327, New Orleans, Louisiana, Mar. 2017.

- Stanley H. Chan, **Xiran Wang** and Omar Elgendy, "Plug-and-Play ADMM for image restoration: Fixed point convergence and applications", IEEE Trans. Comp. Imaging, vol. 3, no. 5, pp.84-98, Mar. 2017.

- Stanley H. Chan, Omar Elgendy and **Xiran Wang**, "Images from bits: Non-iterative image reconstruction for quanta image sensors", MDPI Sensors Special Issue on Photon-Counting Image Sensors, vol. 16, no. 11, paper 1961, pp.1-21, Nov. 2016.

1. INTRODUCTION

1.1 Image Restoration

Image restoration is one of the fundamental problems in image processing, involving a wide range of topics that aims to recover an original image from a corrupted or altered observation. In most cases, some original image serves as an input to an imaging system, either physically defined like MRI devices or conceptually defined by some process on the software side. The system, described by a degradation model, then produces an altered or corrupted output image which will be observed. The process of retrieving the original clean image is called image restoration. There is a wide range of practical problems within image restoration such as: image deblurring where the observed image is blurry caused by motions or optics, image inpainting where parts of the input image is corrupted or removed, image super resolution where the input image is downsampled to a smaller resolution, etc. For all these applications in image restoration, we are only given an altered version of the original latent image and want to recover the original image from the observed image and knowledge about the forward imaging system.

Often time a mathematical model can be found for the imaging system. The model that captures the nature of the corruption or alteration is often called the forward model, which is determined by the specific problem and imaging system at hand. Specifically, we are interested in the following linear model for image restoration problems.

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\epsilon}, \tag{1.1}$$

where $\boldsymbol{x}$ is the input or latent image that serves as the input to the imaging system, $\boldsymbol{A}$ is an operator describing the operation done by the forward system, $\boldsymbol{y}$ is the observed output of the system, and $\boldsymbol{\epsilon}$ is a noise term present inside the system often time modeled as Gaussian. For image restoration problems, we are focusing on a linear system where $\boldsymbol{A}$ can be expressed by a single matrix.

1.2 Examples of Image Restoration

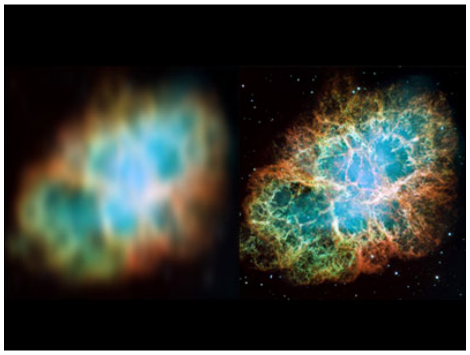
In this section, we will introduce several image restoration problems and discuss the forward models associated with them.

1.2.1 Image Deblurring

Image deblurring is a process of recovering the original clear and noise-free image when a blurry and noisy observation is available. Image deblurring is widely used in many applications including medical imaging, microscopy, astronomy, etc. Figure 1.1 shows its application for astronomy and spiral CT images. In the case of deblurring,we assume the forward model $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ to be a circular convolution matrix that represents a global blur kernel which could be caused by either motions or optics in the physical world. The blur kernel often corresponds to a filter that cause information loss especially in the high frequency part of the spectrum when applied to an image. At the same time, noise is also induced during the forward process. In our case, the goal of image deblurring algorithm is then to recover the lost information under noisy system conditions when the blur kernel itself is known.

1.2.2 Image Inpainting/Interpolation

Image inpainting/interpolation is the process of reconstructing lost/unsampled parts of the original image, and in most practical cases noise is also induced during the forward process. In this case, the $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ matrix is a binary sampling matrix


(a) Deblurring of astronomical photo from Hubble [1]

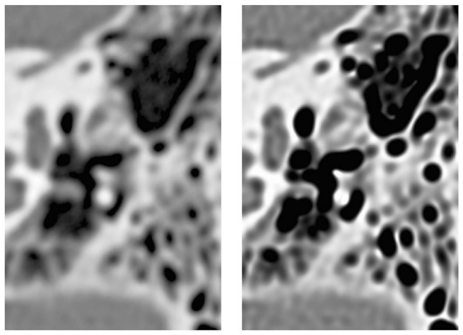
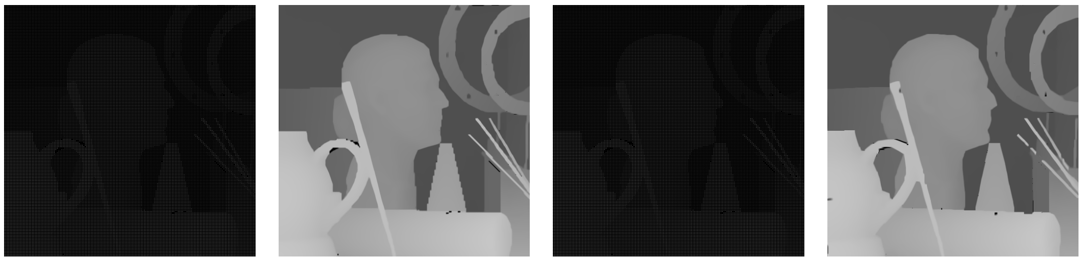

(b) Deblurring of CT images [2]

Fig. 1.1.: Application of image deblurring.

of which each row has only one nonzero element being one. More specifically, $\boldsymbol{S} = squeeze(\tilde{\boldsymbol{S}})$, where $\tilde{\boldsymbol{S}}$ is an $n \times n$ matrix with $S_{ii} = 1$ if pixel i is sampled, the $S_{ij} = 0$ for all other (i, j). The function $squeeze(\cdot) : \mathbb{R}^{n \times n} \to \mathbb{R}^{m \times n}$ discards rows with all zeros. $\boldsymbol{A}$ is also a fat matrix with more columns than rows representing the 'loss' present in the forward model. The most classical application of this process is in the photography and cinema where the film or picture are deteriorated. Inpainting and interpolation also widely used in many modern applications such as HDR imaging with multiple exposure and video compression. It is also recently used in other applications that aims to recover unsampled data. In depth detection, depth detectors such as lidar can only obtain a very sparse and noisy depth measurement of the scene, which is not directly usable. This type of problems fits the model of image interpolation and with the assumption of local smoothness of depth field, a reasonable estimate of the real depth information can be obtained by a model based method such as P&P ADMM. Different from image deblurring, the information loss during the forward process is in the time/spatial domain as directly reflected in the resolution change in the observed image. An example of depth reconstruction using image inpainting/interpolation can be seen in Figure 1.2.



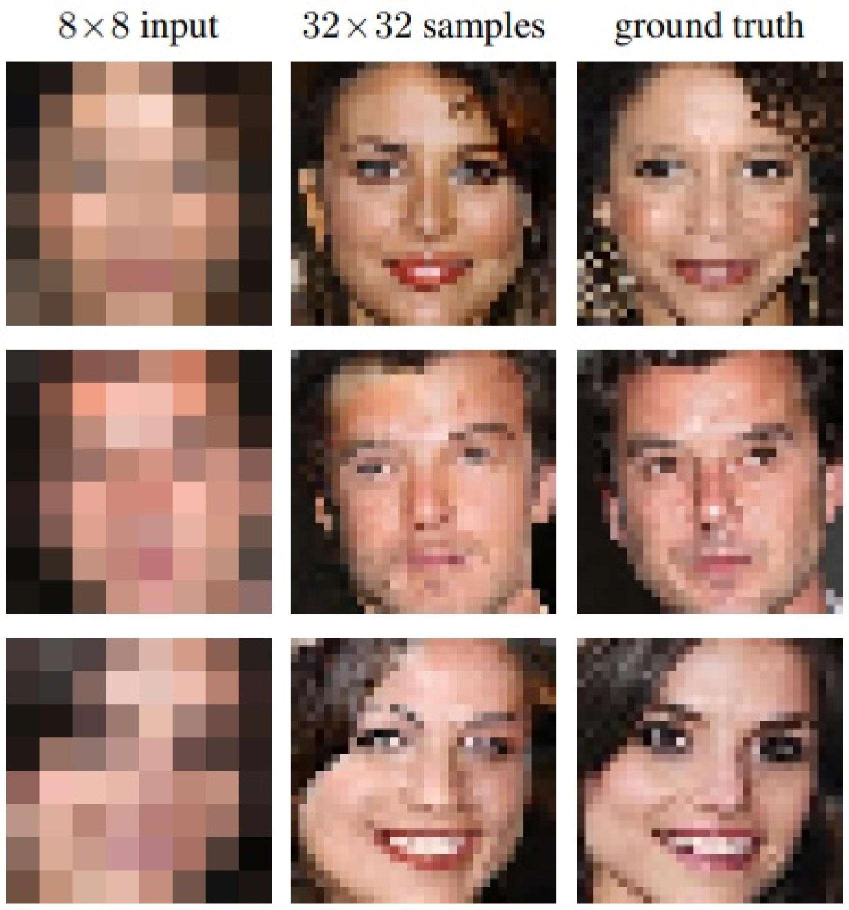
(a) input uniform (b) output (c) input random (d) output

Fig. 1.2.: (a) Uniformly sampled depth data, (b) reconstructed depth map from (a) using the Plug-and-Play framework, (c) randomly sampled depth data, (d) reconstructed depth map from (c) using the Plug-and-Play framework.

1.2.3 Single image super resolution

Super resolution is a process of enhancing the resolution of an input image. In this case, the forward model contains two parts $\boldsymbol{A} = \boldsymbol{SH}$, where $\boldsymbol{S} \in \mathbb{R}^{m \times n}$ and $m < n$, represents a K-fold sampling matrix just like the one in the case of interpolation, and $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ is an anti-aliasing kernel which is the same as the forward model in the deblurring case. We assume both $\boldsymbol{S}$ and $\boldsymbol{H}$ are known, and in this formulation a single-image super-resolution approach can be developed. Super-resolution techniques are often used in image quality enhancement, medical imaging and hyper-spectral imaging. The problem can be thought of as a combination of image deblurring(anti-aliasing kernel) and image inpainting/interpolation(downsampling). To handle the loss of information, multiple low resolution observations are often obtained in order to recover a single frame of high resolution image. However, we focus on the single image approach in our work.

The difficulty for this super-resolution problem lies in the fact that the $\boldsymbol{A}$ operator is neither a diagonal matrix as in the inpainting/interpolation case nor a diagonalizable matrix as in the case of the deblurring problem. As a result, traditional model-based iterative methods usually adopt an approximation approach such as conjugate gradient for solving the inversion step since a direct closed form solution is not available. This makes the inversion step more time consuming and less accurate at the same time. In this work, a closed-form solution for the inversion step is proposed assuming the anti-aliasing filter is applied circularly followed by a standard K-fold downsampling. An example of super-resolution is shown in Figure 1.3.



}

Fig. 1.3.: Illustration of probabilistic pixel recursive super-resolution model by [3]

1.2.4 Single photon imaging

In single-photon imaging, each pixel of the latent image would corresponds to multiple jots on a quanta image sensor, the pixel value of the latent image would determine the photon arrival rate for these jots. The photon count can be be modeled with a Poisson distribution, and a conditional distribution of photon count given the latent pixel value can be obtained. Then the goal would be to find the latent pixel value that best explains these photon counts of these corresponding jots. For this problem, even though the forward model cannot be represented by single matrix, a closed form MAP solution can still be obtained. Since, each latent pixel is independent from each other under our formulation, the pixel value of the latent image can be optimized independently. The optimization problem turns out to be a one-dimensional root finding problem. Then during the second denoising step, local smoothness/correlation will be taken into consideration. An simulation of single-photon imaging is shown in Figure 1.4.

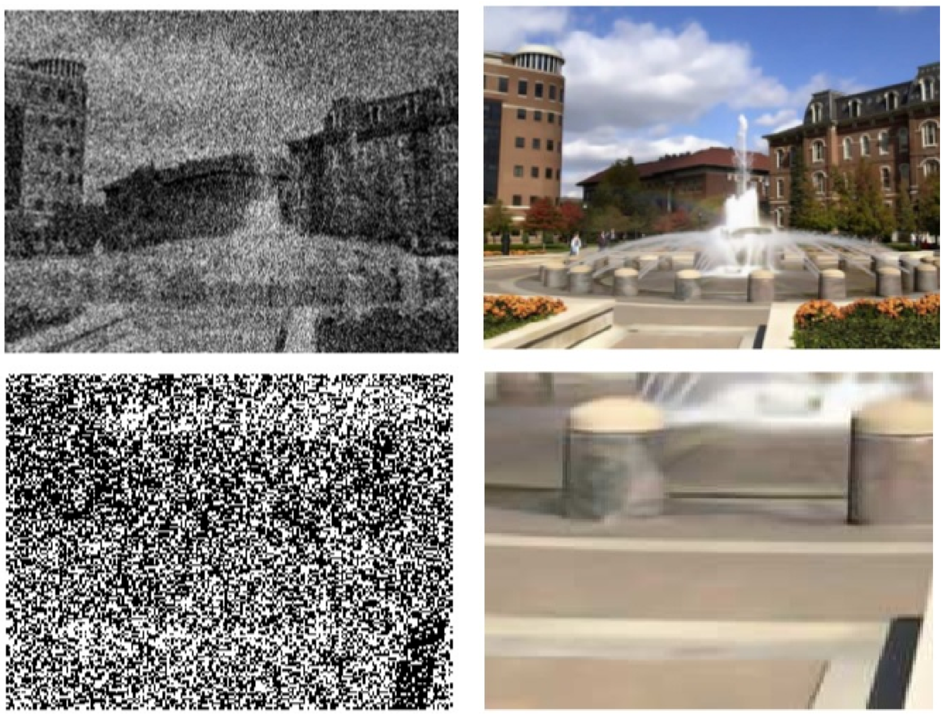
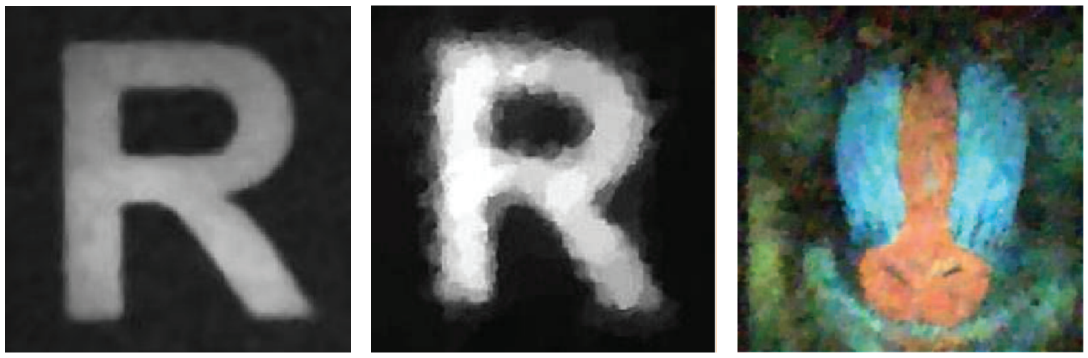

(a) Binary uniform (b) Output

Fig. 1.4.: (a) Binary input image from quanta image sensor (b) Reconstructed color image using Plug-and-Play framework [4]

1.2.5 Single pixel camera

Another interesting application within image restoration is the single pixel camera using compressed sensing [5]. Different from traditional cameras with mega-pixel camera sensors, the single pixel camera has only a single pixel as indicated by its name. The camera works by using an array of N mirrors to reflect light from the scene to the single pixel sensor. These mirrors are either 'On' or 'Off'. As a result, each configuration corresponds to a binary sampling matrix we introduced in previous sections. By using M random mirror configurations that corresponds to M samples of the scene, the forward model $\boldsymbol{A}$ can be described as a randomly generated, fat, binary sampling matrix. The number of samples $M = O(K\log(N/K)) \ll N$ when the scene being imaged is compressible by a compression algorithm. An example is shown.



(a) (a) (b) (b) (c) (c)

Fig. 1.5.: (a) A image being captured by the single pixel camera. (b) Reconstructed image from the camera. (c)Color reconstruction of printout of Mandrill test image.

1.3 Contribution

Efficient implementation: We studied the applications of the Plug-and-Play framework for multiple image restoration problems, such as image deblurring, image inpainting/interpolation, image super-resolution and single photon imaging. Efficeint implementations for each of the problems are also proposed. For the case of image super-resolution, we derived a fast closed form solution for the x sub-problem of the Plug-and-Play ADMM, and compared its performance with commonly used conjugate gradients methods which only output an approximation of the real solution. Our closed form solution can not only find the exactly solution of the x sub-problem but also executes faster than even a single iteration of the conjugate gradient method.

Fixed-point convergence: We also studied the convergence properties of the Plug-and-Play ADMM. We proposed a modified version of the Plug-and-Play algorithm with a continuation scheme that guarantees a fixed point convergence for a wider variety of denoisers compared to existing work. Our fixed point convergence with a wider range of denoisers complements the existing work on a global convergence with a relatively smaller range of denoisers. With the modifications, we also show the algorithm is more robust compared to the original Plug-and-Play ADMM for certain applications.

Internal parameter update: To make the Plug-and-Play algorithm more practical and robust for a wide range of different applications, we introduce the parameter-free version of the Plug-and-Play ADMM. This algorithm is derived from the perspective of the generalized approximate message passing method where a weighted norm is used instead of the standard norm due to the introduction of the variance of each independent random variable. Similarly to generalized approximate message passing, we use the gradient information of the proximal maps of the optimization process, but instead of using the gradient vector, we use the divergence of the proximal maps which is a scalar value that enables the use of any arbitrary off-the-shelf denoisers. To calculate the divergence of the denoiser, denoising is done twice for each iteration, and the divergence is found through the Monte Carlo scheme. Although the parameter-free version has many unknowns about its convergence properties, it does provide a practical and robust alternative for many applications in image restoration problems.

MACE on foreground extraction: Our study then extends to a more generalized framework called multi-agent consensus equilibrium. Each agent of the algorithm is like a single step of the Plug-and-Play ADMM, except that MACE can have an arbitrary number of agents while Plug-and-Play ADMM only have two steps. Different from Plug-and-Play ADMM, MACE also does not rely on setting up an optimization function which means it is not within the same category of the standard optimization methods. The algorithm can simply guarantee a fixed-point convergence as long as each agent is non-expansive, while if an agent does correspond to some proximal map then the MACE result would be the solution to the original optimization problem defined by those proximal maps. We developed a three-agent MACE algorithm for solving the foreground extraction problem involved in the virtual reality content creation pipeline. Utilizing the power of MACE, we are able to achieve state-of-the-art performance combining different agents' horsepower together.

2. PLUG-AND-PLAY ADMM

In this section we will introduce the Plug-and-Play framework for ADMM which is a model-based iterative approach for optimization problems. Although there is a wide range of applications for optimization methods, we will focus on image restoration problems, and thus most variables in the following sections are used to represent images or operators that operates on images.

2.1 ADMM

ADMM was first introduced by [6,7] around 1975, which was originally about using Augmented Lagrangian functional to decouple cost functions with possibly ill-posed linear operator. A number of other papers analyzed the properties of the algorithm: [8] studied convergence for several proposed algorithms, [9] proposed different algorithms for the numerical solution that can be thought as descent methods on the dual formulation, [10–12] presented decomposition algorithms for solving convex problem with a separable structure. The convergence properties of ADMM has also been analyzed, where [13] generalized the Augmented Lagrangian method to the method of multipliers which does not generally involve a Lagrangian and demonstrated equivalence between these methods and some well-known algorithm in nonlinear analysis. [14] showed the Douglas-Rachord splitting method, such as method of multipliers, is a special case of the proximal point algorithm and derived a generalized ADMM for convex programming. Various studies have been focused on applying ADMM on a number of statistical problems such as [15] that proposed a fast iterative convergent algorithm using forward-backward and Douglas-Rachford splitting for sparse signal recovery in linear inverse problems, [16–18] which discussed general optimization frameworks for solving image restoration problems with the help of some priors, and [19, 20] showed applications in some signal processing tasks such as distributed estimation and smoothing of random signals.

In our work, we are focused on applying optimization on a set of image restoration problems, where a hidden/nonobservable image $\boldsymbol{x} \in \mathbb{R}^n$ is fed into a system either physical or algorithmic such that an observable image $\boldsymbol{y} \in \mathbb{R}^m$ is obtained as the output of the system. The goal is then to find the best estimate $\boldsymbol{x}$ that best explains the observation $\boldsymbol{y}$. Using maximum-a-posteriori (MAP), we can formulate the problem as a optimization of a conditional probability:

$$\begin{aligned} \hat{\boldsymbol{x}} &= \underset{\boldsymbol{x}}{\operatorname{argmax}}\, p(\boldsymbol{x}|\boldsymbol{y}) && (2.1) \\ &= \underset{\boldsymbol{x}}{\operatorname{argmin}} - \log p(\boldsymbol{y}|\boldsymbol{x}) - \log p(\boldsymbol{x}) && (2.2) \end{aligned}$$

where $p(\boldsymbol{x}|\boldsymbol{y})$ is a conditional probability that defines the forward model of the imaging system, $p(\boldsymbol{x})$ is a prior distribution defined based on some prior knowledge of the latent image $\boldsymbol{x}$. Often time, as the forward model of a specific problem is well defined, most of the design effort of the optimization problem is put into the prior formulation. Many prior distribution have been well studied in the literature such as L1 norm that prefers a sparse solution for the optimization, L2 norm that is similar to L1 but does not zero out values in the solution and also total variation which is a slightly more complicated prior that prefers a smooth solution. The choice of prior should depend on the specific problem or the preference and understanding of the latent image $\boldsymbol{x}$.

The solution to (2.2) can be found through a variety of optimization approaches such as the sparse reconstruction by separable approximation (SpaRSA) [21], the fast iterative shrinkage thresholding algorithm (FISTA) [22], the two-step iterative shrinkage thresholding (TwIST) [23] or the alternating direction method of multipliers (ADMM) [24], which recently has become the power horse of many applications using the optimization approach. In this work, we mainly focus on the ADMM framework as it is the most versatile method that allows decoupling and shows competitive convergence properties.

The general procedure of applying ADMM starts by turning (2.2) into a constrained optimization problem:

$$\begin{aligned} &\underset{\boldsymbol{x},\boldsymbol{v}}{\operatorname{argmin}} \quad f(\boldsymbol{x}) + \lambda g(\boldsymbol{v}) \\ &\text{subject to} \quad \boldsymbol{x} = \boldsymbol{v} \end{aligned} \tag{2.3}$$

where $f(\boldsymbol{x})$ corresponds to the forward imaging model, $g(\boldsymbol{v})$ corresponds to the prior term, and with the constraint $\boldsymbol{x} = \boldsymbol{v}$ such that the solution to the constrained problem should be the same as that of the original problem. Based on the constrained problem formulation, we can write out the corresponding augmented Lagrangian:

$$L(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{u}) = f(\boldsymbol{x}) + \lambda g(\boldsymbol{v}) + \boldsymbol{u}^T(\boldsymbol{x} - \boldsymbol{v}) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2 \tag{2.4}$$

where $\boldsymbol{u}$ is called the Lagrangian multiplier and ρ is an internal parameter of the augmented Lagrangian. ADMM then proceeds by decoupling the augmented Lagrangian into three parts with each part corresponding to a step of the iterative algorithm as the following:

$$\begin{aligned} \boldsymbol{x}^{k+1} &= \underset{\boldsymbol{x}}{\operatorname{argmin}}\, L(\boldsymbol{x}, \boldsymbol{v}^k, \boldsymbol{u}^k) \\ \boldsymbol{v}^{k+1} &= \underset{\boldsymbol{v}}{\operatorname{argmin}}\, L(\boldsymbol{x}^{k+1}, \boldsymbol{v}, \boldsymbol{u}^k) \\ \boldsymbol{u}^{k+1} &= \boldsymbol{u}^k + \rho(\boldsymbol{x} - \boldsymbol{v}) \end{aligned} \tag{2.5}$$

Under certain conditions, e.g., when both the forward model term $f(\boldsymbol{x})$ and the prior term $g(\boldsymbol{v})$ are closed, proper and convex, and there exists a saddle point for L, then it's proven that these iterative steps will reach a convergence which is the solution of the original optimization problem stated in (2.2). The ADMM method has good robustness of the more general method of multipliers and also supports decomposition of the optimization problem.

2.2 Plug-and-Play framework

The Plug-and-Play framework was originally introduced in [25] as a framework that allows both the forward models of general imaging systems and state-of-the-art denoisers. Following that, an initial convergence analysis was done by [26] as the convergence properties of the Plug-and-Play framework was largely unknown. This work demonstrated the global convergence of the Plug-and-Play framework, and just like the classical ADMM algorithms when the denoiser can be represented using a doubly stochastic matrix or in other words a symmetric smoothing filter. Although global convergence is perhaps the strongest form of convergence possible, restricting the denoiser to be a simple symmetric smoothing filter weakens the key concept of the Plug-and-Play framework, which is being flexible enough to be compatible with a wide range of state-of-the-art denoisers. To tackle this issue, the work by [4] is done through introducing a modified version of the Plug-and-Play framework with a continuation scheme such that a wider range of denoisers can be used however with a trade-off being that only a fixed convergence is guaranteed.

Some variations of the Plug-and-Play framework are also proposed such as [27] that embedded a class-adapted denoiser using GMM so that simultaneous restoration and semantic segmentation can be done. [28] proposed a regularization by denoising (RED) algorithm claiming it is capable of incorporating any denoisers and guarantees a global convergence. However, this work is rebutted by [29] stating the proof for RED is only true when the denoiser has a symmetric Jacobian which is not the case for most common state-of-the-art denoisers, and RED algorithms seek a consensus equilibrium solution. At the same time they proposed a different variation called Score-Matching by Denoising (SMD) which aims to match the gradient of a log-prior. Tight connection between SMD and kernel density estimation can be shown. Another variation is proposed by [30] based the fast iterative shrinkage/threshold algorithm (FISTA) for Fourier ptychographic microscopy.

Although many work has been focusing the application of the Plug-and-Play framework, few have addressed why the framework works so well. For this purpose, a recent work by [31] provides an analysis on the Plug-and-Play framework by using the concept of consensus equilibrium and shows an equivalent MAP optimization exists for a Plug-and-Play algorithm when the denoiser is a symmetric smoothing filter.

For the following section, we will provide a brief introduction of the key concept behind the Plug-and-Play framework. We start by expressing (2.5) more explicitly as below:

$$\boldsymbol{x}^{k+1} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\, f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{x} - \tilde{\boldsymbol{x}}^k\|^2 \tag{2.6}$$

$$\boldsymbol{v}^{k+1} = \underset{\boldsymbol{v}}{\operatorname{argmin}}\, \lambda g(\boldsymbol{v}) + \frac{\rho}{2}\|\boldsymbol{v} - \tilde{\boldsymbol{v}}^k\|^2 \tag{2.7}$$

$$\hat{\boldsymbol{u}}^{k+1} = \hat{\boldsymbol{u}}^k + (\boldsymbol{x}^{k+1} + \boldsymbol{v}^{k+1}) \tag{2.8}$$

where $\tilde{\boldsymbol{x}}^k = \boldsymbol{v}^k - \hat{\boldsymbol{u}}^k$, $\tilde{\boldsymbol{v}}^k = \boldsymbol{x}^k + \hat{\boldsymbol{u}}^k$ and $\hat{\boldsymbol{u}}^k = \frac{1}{\rho}\boldsymbol{u}^k$. A key feature of the ADMM algorithm is its decoupling and modular structure. (2.6) can be seen as an inversion step trying to find the latent image that fits the forward imaging model $f(\boldsymbol{x})$ the best. Let $\sigma = \sqrt{\frac{\lambda}{\rho}}$, we can rewrite (2.7) into:

$$\boldsymbol{v}^{k+1} = \underset{\boldsymbol{v}}{\operatorname{argmin}}\, g(\boldsymbol{v}) + \frac{1}{2\sigma^2}\|\boldsymbol{v} - \tilde{\boldsymbol{v}}^k\|^2 \tag{2.9}$$

If we think $\tilde{\boldsymbol{v}}^k$ as a noisy image, (2.9) is essentially trying to minimize the difference between $\boldsymbol{v}$ and $\tilde{\boldsymbol{v}}^k$ while enforcing a prior term $g(\boldsymbol{v})$. If we have $g(\boldsymbol{v}) = \|\boldsymbol{v}\|_{TV}$, then (2.9) becomes a standard total variation denoising problem aiming to remove the noise in $\tilde{\boldsymbol{v}}^k$.

Seeing (2.7) as a denoising module, [25] propose a variation of ADMM where any off-the-shelf denoiser can be used to replace this denoising step without the need to specify the prior term $g(\boldsymbol{v})$ in the original optimization problem. As a result, (2.7) will become:

$$\boldsymbol{v}^{k+1} = \mathcal{D}_\sigma(\tilde{\boldsymbol{v}}^k) \tag{2.10}$$

where $\mathcal{D}_\sigma$ is some off-the-shelf denoiser with noise level specified as σ. This is why this newly proposed framework is coined P&P ADMM. Numerous studies have found its performance to be very promising for a wide range of applications [4, 25, 26]. In this work, we address the applications of P&P ADMM in several image restoration problems, provide convergence analysis for the algorithm and propose an automatic update scheme for internal parameters.

To ensure global convergence of the ADMM algorithm, classical results require $g(\boldsymbol{v})$ to be closed, proper and convex. Although new results have demonstrate the performance of ADMM for nonconvex problems [32], little work is done considering an implicitly defined $g(\boldsymbol{v})$ through $\mathcal{D}_\sigma$. The only work containing convergence analysis for the P&P ADMM is done by [26] for $\mathcal{D}_\sigma$ that is a symmetric smoothing filter [33]. In our work, by adopting a continuation scheme we are able to show the algorithm is guaranteed to converge for a much broader range of denoisers called bounded denoisers that approaches the identity operator as the denoising parameter (noise level) vanishes. Bounded denoisers are weaker than the non-expansive denoisers in [26], as a result a weaker form of fixed point convergence can be achieved instead of a global one.

Although assurance of convergence is nice, tuning internal parameter of the algorithm can be difficult in the practice as not only the performance can be quite sensitive to these parameters but also the rate of convergence can be largely dependent on them. As a result, it would be undesirable in the practice for users to tune these parameters for different optimization problems. In this work, we also offer an parameter-free version of the P&P ADMM in exchange of the guarantee of convergence. In this variation of the P&P ADMM, the internal parameters are updated as part of the iterative optimization process. This algorithm is derived from the perspective of the generalized approximate message passing [34] with multiple modifications. Although the vanilla P&P ADMM with fine tuned parameters can still out performs the PAMP, we find the proposed PAMP to achieve solutions along a reliable and fast convergence path without users having to worry about tuning the internal parameters.

2.3 Parameter-free Plug-and-Play Framework

Although we discuss the promising performance of the Plug-and-Play framework for various applications, the behavior of the algorithm is largely unknown especially in terms of the internal parameters which controls the rate of convergence of the algorithm and also the quality of the final output. In practice, it is also not desirable for users to hand tune parameters from time to time on different applications. As a result, we propose a parameter-free variation of the P&P ADMM derived from the perspective of the generalized approximate message passing (GAMP) [34].

Just like ADMM, GAMP considers the same constrained optimization problem in (2.3). GAMP then treats $f(\boldsymbol{x})$ and $g(\boldsymbol{v})$ as two different nodes, the output node and the input node, and passes intermediate parameters (messages) between the two nodes both forward and backward. However, different from ADMM, the intermediate parameters are vectors used to calculate weighted norms of data and prior terms. These intermediate parameters are calculated using the gradient of the proximal maps of $f(\boldsymbol{x})$ and $(\boldsymbol{v})$.

In this work, in deriving the parameter-free version of the P&P ADMM we make several modifications: we change the vectorized parameters into a scalar parameter so that the algorithm would still work for an arbitrary denoiser with non-separable cost; we replace the second step of the GAMP with an off-the-shelf denoiser so that the obtained new algorithm fits the Plug-and-Play framework. In order to replace vectorized parameter with scalar parameter, we also adopt divergence which is a scalar instead of the vector of gradients as in GAMP.

The divergence of the inversion step is relatively straight forward, but a general closed-form for the divergence of some arbitrary denoiser is impossible. To alleviate the issue, we use the Monte Carlo scheme to numerically approximate the divergence of the off-the-shelf denoiser. The denoising step has to be performed twice as a trade off: one for $\mathcal{D}(\tilde{\boldsymbol{v}})$ and one for $\mathcal{D}(\tilde{\boldsymbol{v}} + \epsilon \boldsymbol{b})$, where $\epsilon \boldsymbol{b}$ is a small random noise.

Intuitively, the divergence of a function measures the sensitivity of the function with respect to the input. Essentially, PAMP attenuates the influence of the denoiser when solving the inversion step if the denoiser has a high divergence and is sensitive to $\boldsymbol{v}$. And if the denoiser is quite stable for some $\boldsymbol{v}$ with low divergence, PAMP puts more emphasis on the denoised image when solving subsequent inversion step.

Although the convergence of PAMP remains an open question, empirical studies in our work have shown relatively stable performance for multiple image restoration problems. The goal of PAMP is to update the internal parameters of P&P ADMM so that the algorithm is more practical and robust for different problems.

2.4 MACE

Multi-agent consensus equilibrium (MACE) [35] is a generalization of the ADMM framework. The goal of the algorithm is to minimize a cost function that contains an arbitrary number of terms instead of just two in the case of a standard ADMM. In this section, we will discuss the application of MACE on foreground extraction for videos.

2.4.1 Introduction on MACE

General optimization algorithms focuses on finding a minimizer for a sum of auxiliary functions defined perhaps by a multi-modal system:

$$\underset{f}{\text{minimize}}(\boldsymbol{x}) = \sum_{i=1}^{N} f_i(\boldsymbol{x}) \tag{2.11}$$

where $\boldsymbol{x} \in \mathbb{R}^n$. In consensus optimization, a set of separate variables are introduced with the constraint that these separate variables must be equal. This type of problem is solvable through frameworks such as ADMM. However, due to the emergence of deep neural networks and other non-optimization based algorithms that are not captured by traditional optimization, there exists a substantial gap between the classical optimization based approach and the state-of-the-art operators such as a deep learning based denoiser. The goal of MACE is to generalize traditional optimization framework to emcompass models and algorithms not associated with an optimization problem. Free from optimization cost functions, the solution of MACE is defined as follows:

$$\begin{aligned} F_i(\boldsymbol{x}^* + \boldsymbol{u}_i^*) &= \boldsymbol{x}^* \quad , \quad i = 1, ..., N \\ \hat{\boldsymbol{u}}_\mu^* &= 0 \end{aligned} \tag{2.12}$$

where F_i's are N vector-valued maps. To show MACE can generalize the consensus optimization problem, [35] presents a proof showing when the F_i's are proximal maps with corresponding f_i's being proper, closed and convex functions, then the solution as defined in (2.12) is the same as the solution of the optimization problem stated by (2.11). Certainly, the goal of MACE is not to just generalize the traditional consensus optimization problem, but to be able to include non-optimization based operators.

The solution of MACE can be found by finding the fixed point of the map:

$$(2\boldsymbol{G}_\mu - \boldsymbol{I})(2\boldsymbol{F} - \boldsymbol{I})\boldsymbol{v}^* = \boldsymbol{v}^* \tag{2.13}$$

where

$$\boldsymbol{F}(\boldsymbol{v}) = \begin{bmatrix} F_1(v_1) \\ F_2(v_2) \\ \vdots \\ F_N(v_N) \end{bmatrix} \quad and \quad \boldsymbol{G}_\mu(\boldsymbol{v}) = \begin{bmatrix} \hat{\boldsymbol{v}}_\mu \\ \hat{\boldsymbol{v}}_\mu \\ \vdots \\ \hat{\boldsymbol{v}}_\mu \end{bmatrix} \tag{2.14}$$

with $\hat{\boldsymbol{v}}_\mu$ being the average of the set of vectors $\boldsymbol{v}_i$'s in $\boldsymbol{v}$. When $\boldsymbol{T} = (2\boldsymbol{G}_\mu - \boldsymbol{I})(2\boldsymbol{F} - \boldsymbol{I})$ is non expansive the the fixed point of the map $\boldsymbol{T}$ can be found through anistropic preconditioned Mann iterations:

$$\boldsymbol{v}^{k+1} = (1-\rho)\boldsymbol{v}^k + \rho\boldsymbol{T}(\boldsymbol{v}^k) \tag{2.15}$$

By treating CE as a root finding problem, Newton's method can also be used to find the MACE solution. However, this is beyond the scope of this work.

Intuitively, MACE can be thought of as a system containing two major parts: one consists of multiple agents each could correspond to a term in the objective function; the other one is a consensus unit that gathers outputs from each agent in the first part to calculate a 'consensus' and then distributes the 'consensus' back to each agent again. Similar to constrained optimization, each agent can be run in parallel which is an advantage over standard ADMM algorithms. As a result, each MACE algorithm is only bottle-necked by the slowest agent in the framework. Another major advantage of MACE is that it is not only a framework for standard optimization problems. Although MACE can have a well-defined global convergence when each agent corresponds to a convex and proper function, MACE is able to achieve a fixed point convergence even when arbitrary agents such as a neural network denoiser are used as long as each agent is non-expansive. This feature liberates the algorithm and enables a much broader range of applications. In our work, we design a MACE algorithm for foreground extraction which is not a nicely formulated problem as the more common image restoration problems. But with the flexibility of the MACE framework, we are able to achieve promising performance even compared to state-of-the-art competing algorithms.

2.4.2 Foreground extraction

Foreground extraction is one of the most studied topics in computer vision. In this work, we focus on foreground extraction in the image processing pipeline of a virtual reality system. The standard mathematical formulation of the problem is as below:

$$\boldsymbol{I} = \boldsymbol{\alpha}\boldsymbol{F} + (1 - \boldsymbol{\alpha})\boldsymbol{B} \tag{2.16}$$

where $\boldsymbol{I}$ is the image we observe, $\boldsymbol{\alpha}$ is a grey scale mask, $\boldsymbol{F}$ is the latent foreground image, and $\boldsymbol{B}$ is the latent background image. Under this formulation, we are essentially saying the observed image is a convex combination of a foreground image and a background image. The goal of standard foreground extraction algorithm is to find $\boldsymbol{\alpha}$ given the observed image $\boldsymbol{I}$. The problem is obviously under-determined. To tackle this problem, a standard approach called alpha matting [36–39] is developed, where user need to provide additional information in the form of a trimap or scribbles indicating foreground and background areas in $\boldsymbol{I}$.

Although alpha matting alleviates the problem, large amount of human efforts are needed in order to process a huge volume of image data, which is the common situation of virtual reality content creation. On the other hand, we tackle the problem from a different perspective with different assumptions about the problem that are more friendly for the VR content creation setting. First, instead of having a trimap for each frame of a video, we assume a single plate image containing pure background is available. This is very feasible in most filming shooting environment where camera can simply capture a single frame before the objects enter the scene. This approach is almost effortless compared to drawing trimaps for all frames in a video. Second, we only focus on the major boundaries as fine details can always be improved using state-of-the-art alpha matting algorithms as a post-processing.

2.4.3 Agents for foreground extraction

We develop three agents for the MACE algorithm for foreground extraction. The first agent is inspired by closed-form matting [36] that uses a linear model for $\boldsymbol{\alpha}$, which is a generalization of the problem formulation in (2.16). Under this linear model, closed-form-matting is able to derive a Laplacian matrix that contains inter-pixel information which can used used for solving $\boldsymbol{\alpha}$. The original closed-form matting requires a trimap as input like most alpha matting algorithms. The proposed agent takes the plate image as input instead and does not require any user input. A new dual-layer Laplacian matrix is derived to accommodate the additional plate image. The original objective function of closed-form-matting also has a term that enforces the trimap input on the result. This is replaced by a softer enforcement based on thresholding an operation on the input to the agent.

The second agent draws both the color and texture cues from the plate and input frames. In using the color cue, the frame difference is calculated between the plate and the image and is then smoothed with a bilateral filter guided by the input image. As a result, when plate and input images are similar in color for a region, the result using color cue will have very small values for this region. However, color cue only is not enough when the foreground and background have similar colors. To tackle this issue, texture information also needs to be explored. The input image is segmented using super-pixel algorithm [40], and the same segmentation is placed on the plate image. Texture information is then calculated for each super-pixel on both plate and input image. Then super-pixels on plate and input images are compared to make a soft decision on whether a super-pixel on the input image belongs to foreground or background. The final estimate is simply the pixel-wise product between the color cue and texture cue results. And this final estimate is then fed into an objective function so that it can be used iteratively within the MACE framework. An additional term that prefers binary pixel values is also added to the objective function of the second agent.

Perhaps not surprisingly, the third agent is an off-the-shelf denoiser just like Plug-and-Play ADMM. Using different denoisers for this agent will produce different final results for MACE. We tested on several denoisers such as Total variation [41], BM3D [42] and a deep learning denoiser trained for iterative algorithms. Overall, total variation denoiser offers the best balance among performance, robustness and computational complexity.

Because of the flexibility of MACE, other agent designs for foreground extraction could also be explored, but it would be beyond the scope of this work. In our work, we show the proposed three agents achieve promising performance. In ablation study, the performance degrades largely when any one of them is absent in MACE with the rest unchanged.

3. PLUG-AND-PLAY ADMM FOR IMAGE RESTORATION: FIXED POINT CONVERGENCE AND APPLICATIONS

3.1 Introduction

3.1.1 MAP and ADMM

Many image restoration tasks can be posted as the following inverse problem: Given an observed image $\boldsymbol{y} \in \mathbb{R}^n$ corrupted according to some forward model and noise, find the underlying image $\boldsymbol{x} \in \mathbb{R}^n$ which "best explains" the observation. In estimation, we often formulate this problem as a maximum-a-posteriori (MAP) estimation [43], where the goal is to maximize the posterior probability:

$$\begin{aligned} \widehat{\boldsymbol{x}} &= \underset{\boldsymbol{x}}{\operatorname{argmax}} \quad p(\boldsymbol{x} \mid \boldsymbol{y}) \\ &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \quad -\log p(\boldsymbol{y} \mid \boldsymbol{x}) - \log p(\boldsymbol{x}), \end{aligned} \tag{3.1}$$

for some conditional probability $p(\boldsymbol{y} \mid \boldsymbol{x})$ defining the forward imaging model, and a prior distribution $p(\boldsymbol{x})$ defining the probability distribution of the latent image. Because of the explicit use of the forward and the prior models, MAP estimation is also a model-based image reconstruction (MBIR) method [44] which has many important applications in deblurring [18,45,46], interpolation [41,47,48], super-resolution [49–52] and computed tomography [26], to name a few.

It is not difficult to see that solving the MAP problem in (3.1) is equivalent to solving an optimization problem

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \quad f(\boldsymbol{x}) + \lambda g(\boldsymbol{x}), \tag{3.2}$$

with $f(\boldsymbol{x}) \overset{\text{def}}{=} -\log p(\boldsymbol{y}\,|\,\boldsymbol{x})$ and $g(\boldsymbol{x}) \overset{\text{def}}{=} -(1/\lambda)\log p(\boldsymbol{x})$. The optimization in (3.2) is a generic unconstrained optimization. Thus, standard optimization algorithms can be used to solve the problem. In this work, we focus on the alternating direction method of multiplier (ADMM) [24], which has become the workhorse for a variety of problems in the form of (3.2).

The idea of ADMM is to convert (3.2), an unconstrained optimization, into a constrained problem

$$(\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{v}}) = \underset{\boldsymbol{x},\boldsymbol{v}}{\text{argmin}} \quad f(\boldsymbol{x}) + \lambda g(\boldsymbol{v}), \quad \text{subject to} \quad \boldsymbol{x} = \boldsymbol{v}, \tag{3.3}$$

and consider its augmented Lagrangian function:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{u}) = f(\boldsymbol{x}) + \lambda g(\boldsymbol{v}) + \boldsymbol{u}^T(\boldsymbol{x} - \boldsymbol{v}) + \frac{\rho}{2}\|\boldsymbol{x} - \boldsymbol{v}\|^2. \tag{3.4}$$

The minimizer of (3.3) is then the saddle point of $\mathcal{L}$, which can be found by solving a sequence of subproblems

$$\boldsymbol{x}^{(k+1)} = \underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{argmin}} \quad f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{x} - \widetilde{\boldsymbol{x}}^{(k)}\|^2, \tag{3.5}$$

$$\boldsymbol{v}^{(k+1)} = \underset{\boldsymbol{v}\in\mathbb{R}^n}{\text{argmin}} \quad \lambda g(\boldsymbol{v}) + \frac{\rho}{2}\|\boldsymbol{v} - \widetilde{\boldsymbol{v}}^{(k)}\|^2, \tag{3.6}$$

$$\bar{\boldsymbol{u}}^{(k+1)} = \bar{\boldsymbol{u}}^{(k)} + (\boldsymbol{x}^{(k+1)} - \boldsymbol{v}^{(k+1)}), \tag{3.7}$$

where $\bar{\boldsymbol{u}}^{(k)} \overset{\text{def}}{=} (1/\rho)\boldsymbol{u}^{(k)}$ is the scaled Lagrange multiplier, $\widetilde{\boldsymbol{x}}^{(k)} \overset{\text{def}}{=} \boldsymbol{v}^{(k)} - \bar{\boldsymbol{u}}^{(k)}$ and $\widetilde{\boldsymbol{v}}^{(k)} \overset{\text{def}}{=} \boldsymbol{x}^{(k+1)} + \bar{\boldsymbol{u}}^{(k)}$. Under mild conditions, e.g., when both f and g are closed, proper and convex, and if a saddle point of $\mathcal{L}$ exists, one can show that the iterates (5.2a)-(5.2c) converge to the solution of (3.3) (See [24] for details).

3.1.2 Plug-and-Play ADMM

An important feature of the ADMM iterations (5.2a)-(5.2c) is its modular structure. In particular, (5.2a) can be regarded as an inversion step as it involves the forward imaging model $f(\boldsymbol{x})$, whereas (5.2b) can be regarded as a denoising step as it involves the prior $g(\boldsymbol{v})$. To see the latter, if we define $\sigma = \sqrt{\lambda/\rho}$, it is not difficult to show that (5.2b) is

$$\boldsymbol{v}^{(k+1)} = \underset{\boldsymbol{v}\in\mathbb{R}^n}{\operatorname{argmin}} \quad g(\boldsymbol{v}) + \frac{1}{2\sigma^2}\|\boldsymbol{v} - \widetilde{\boldsymbol{v}}^{(k)}\|^2. \tag{3.8}$$

Treating $\widetilde{\boldsymbol{v}}^{(k)}$ as the "noisy" image, (3.8) minimizes the residue between $\widetilde{\boldsymbol{v}}^{(k)}$ and the "clean" image $\boldsymbol{v}$ using the prior $g(\boldsymbol{v})$. For example, if $g(\boldsymbol{x}) = \|\boldsymbol{x}\|_{TV}$ (the total variation norm), then (3.8) is the standard total variation denoising problem.

Building upon this intuition, Venkatakrishnan et al. [25] proposed a variant of the ADMM algorithm by suggesting that one does not need to specify g before running the ADMM. Instead, they replace (5.2b) by using an off-the-shelf image denoising algorithm, denoted by $\mathcal{D}_\sigma$, to yield

$$\boldsymbol{v}^{(k+1)} = \mathcal{D}_\sigma\left(\widetilde{\boldsymbol{v}}^{(k)}\right). \tag{3.9}$$

Because of the heuristic nature of the method, they called the resulting algorithm as the Plug-and-Play ADMM. An interesting observation they found in [25] is that although Plug-and-Play ADMM appears ad-hoc, for a number of image reconstruction problems the algorithm indeed performs better than some state-of-the-art methods. A few recent reports have concurred similar observations [26, 53–55].

3.1.3 Challenges of Plug-and-Play ADMM

From a theoretical point of view, the main challenge of analyzing Plug-and-Play ADMM is the denoiser $\mathcal{D}_\sigma$. Since $\mathcal{D}_\sigma$ is often nonlinear and does not have closed form expressions, the analysis has been very difficult. Specifically, the following three questions remain open:

1. Convergence of the Algorithm. Classical results of ADMM require g to be closed, proper and convex in order to ensure convergence [24]. While newer results have extended ADMM for nonconvex problems [32], there is little work addressing the case when g is defined implicitly through $\mathcal{D}_\sigma$. To the best of our knowledge, the only existing convergence analysis, to date, is the one by Sreehari et al. [26] for the case when $\mathcal{D}_\sigma$ is a symmetric smoothing filter [56,57]. However, for general $\mathcal{D}_\sigma$ the convergence is not known.

2. Original Prior. Since $\mathcal{D}_\sigma$ is an off-the-shelf image denoising algorithm, it is unclear what prior g does it correspond to. In [58], Chan addresses this question by explicitly deriving the original prior g when $\mathcal{D}_\sigma$ is a symmetric smoothing filter [58]. In this case, the author shows that g is a modified graph Laplacian prior, with better restoration performance compared to the conventional graph Laplacian [33]. However, beyond symmetric smoothing filters it becomes unclear if we can find the corresponding g.

3. Implementation. The usage of Plug-and-Play ADMM has been reported in a few scattered occasions, with some work in electron tomography [26], compressive sensing [53], and some very recent applications in Poisson recovery [54] and super-resolution [55]. However, the common challenge underpinning these applications is whether one can obtain a fast solver for the inversion step in (5.2a). This has not been a problem for conventional ADMM, because in many cases we can use another variable splitting strategy to replace $\boldsymbol{v} = \boldsymbol{x}$ in (3.3), e.g., using $\boldsymbol{v} = \boldsymbol{Bx}$ when $g(\boldsymbol{x}) = \|\boldsymbol{Bx}\|_1$ [45].

3.1.4 Related Works

Plug-and-Play ADMM was first reported in 2013. Around the same period of time there is an independent series of studies using denoisers for approximate message passing (AMP) algorithms [42, 59–61]. The idea was to replace the shrinkage step of the standard AMP algorithm with any off-the-shelf algorithm in the class of "proper denoisers" – denoisers which ensure that the noise variance is sufficiently suppressed. (See Section 3.2.3 for more discussions.) However, this type of denoise-AMP algorithms rely heavily on the Gaussian statistics of the random measurement matrix $\boldsymbol{A}$ in a specific forward model $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2$. Thus, if $f(\boldsymbol{x})$ departs from quadratic or if $\boldsymbol{A}$ is not random, then the behavior of the denoise-AMP becomes unclear.

Using denoisers as building blocks of an image restoration algorithm can be traced back further, e.g., wavelet denoiser for signal deconvolution [62]. Of particular relevance to Plug-and-Play ADMM is the work of Danielyan et al. [63], where they proposed a variational method for deblurring using BM3D as a prior. The idea was later extended by Zhang et al. to other restoration problems [64]. However, these algorithms are customized for the specific denoiser BM3D. In contrast, the proposed Plug-and-Play ADMM supports any denoiser satisfying appropriate assumptions. Another difference is that when BM3D is used in [63] and [64], the grouping of the image patches are fixed throughout the iterations. Plug-and-Play ADMM allows re-calculation of the grouping at every iteration. In this aspect, the Plug-and-Play ADMM is more general than these algorithms.

A large number of denoisers we use nowadays are patch-based denoising algorithms. All these methods can be considered as variations in the class of universal denoisers [65, 66] which are asymptotically optimal and do not assume external knowledge of the latent image (e.g., prior distribution). Asymptotic optimality of patch-based denoisers has been recognized empirically by Levin et al. [67, 68], who showed that non-local means [69] approaches the MMSE estimate as the number of patches grows to infinity. Recently, Ma et al. [70] made attempts to integrate universal denoisers with approximate message passing algorithms.

3.1.5 Contributions

The objective of this work is to address the first and the third issue mentioned in Section 3.1.3. The contributions of this work are as follows:

First, we modify the original Plug-and-Play ADMM by incorporating a continuation scheme. We show that the new algorithm is guaranteed to converge for a broader class of denoisers known as the *bounded denoisers*. Bounded denoisers are asymptotically invariant in the sense that the denoiser approaches an identity operator as the denoising parameter vanishes. Bounded denoisers are weaker than the non-expansive denoisers presented in [26]. However, for weaker denoisers we should also expect a weaker form of convergence. We prove that the new Plug-and-Play ADMM has a *fixed point* convergence, which complements the global convergence results presented in [26].

Second, we discuss fast implementation techniques for image super-resolution and single photon imaging problems. For the super-resolution problem, conventional ADMM requires multiple variable splits or an inner conjugate gradient solver to solve the subproblem. We propose a polyphase decomposition based method which gives us closed-form solutions. For the single photon imaging problem, existing ADMM algorithm are limited to explicit priors such as total variation. We demonstrate how Plug-and-Play ADMM can be used and we present a fast implementation by exploiting the separable feature of the problem.

The rest of the chapter is organized as follows. We first discuss the Plug-and-Play ADMM algorithm and the convergence properties in Section 3.2. We then discuss the applications in Section 3.3. Experimental results are presented in Section 3.4.

3.2 Plug-and-Play ADMM and Convergence

In this section we present the proposed Plug-and-Play ADMM and discuss its convergence property. Throughout this chapter, we assume that the unknown image $\boldsymbol{x}$ is bounded in an interval $[x_{\min},\ x_{\max}]$ where the upper and lower limits can be obtained from experiment or from prior knowledge. Thus, without loss of generality we assume $\boldsymbol{x} \in [0,1]^n$.

3.2.1 Plug-and-Play ADMM

The proposed Plug-and-Play ADMM algorithm is a modification of the conventional ADMM algorithm in (5.2a)-(5.2c). Instead of choosing a constant ρ, we increase ρ by $\rho_{k+1} = \gamma_k \rho_k$ for $\gamma_k \geq 1$. In optimization literature, this is known as a continuation scheme [71] and has been used in various problems, e.g., [72, 73]. Incorporating this idea into the ADMM algorithm, we obtain the following iteration:

$$\boldsymbol{x}^{(k+1)} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\ f(\boldsymbol{x}) + (\rho_k/2)\|\boldsymbol{x} - (\boldsymbol{v}^{(k)} - \boldsymbol{u}^{(k)})\|^2 \tag{3.10}$$

$$\boldsymbol{v}^{(k+1)} = \mathcal{D}_{\sigma_k}(\boldsymbol{x}^{(k+1)} + \boldsymbol{u}^{(k)}) \tag{3.11}$$

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + (\boldsymbol{x}^{(k+1)} - \boldsymbol{v}^{(k+1)}) \tag{3.12}$$

$$\rho_{k+1} = \gamma_k \rho_k, \tag{3.13}$$

where $\mathcal{D}_{\sigma_k}$ is a denoising algorithm (called a "denoiser" for short), and $\sigma_k \overset{\text{def}}{=} \sqrt{\lambda/\rho_k}$ is a parameter controlling the strength of the denoiser.

There are different options in setting the update rule for ρ_k. In this work we present two options. The first one is a *monotone update* rule which defines

$$\rho_{k+1} = \gamma \rho_k, \qquad \text{for all } \ k \tag{3.14}$$

Algorithm 1 Plug-and-Play ADMM

```
Input: $\rho_0$, $\lambda$, $\eta < 1$, $\gamma > 1$.
while Not Converge do
    $\boldsymbol{x}^{(k+1)} = \underset{\boldsymbol{x}}{\text{argmin}}\ f(\boldsymbol{x}) + (\rho_k/2)\|\boldsymbol{x} - (\boldsymbol{v}^{(k)} - \boldsymbol{u}^{(k)})\|^2$
    $\boldsymbol{v}^{(k+1)} = \mathcal{D}_{\sigma_k}(\boldsymbol{x}^{(k+1)} + \boldsymbol{u}^{(k)})$, where $\sigma_k = \sqrt{\lambda/\rho_k}$
    $\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + (\boldsymbol{x}^{(k+1)} - \boldsymbol{v}^{(k+1)})$
    if $\Delta_{k+1} \geq \eta\Delta_k$ then
        $\rho_{k+1} = \gamma\rho_k$
    else
        $\rho_{k+1} = \rho_k$
    end if
    $k = k + 1$.
end while
```

for a constant $\gamma > 1$. The second option is an *adaptive update* rule by considering the relative residue:

$$\Delta_{k+1} \overset{\text{def}}{=} \frac{1}{\sqrt{n}}\Big(\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_2 + \|\boldsymbol{v}^{(k+1)} - \boldsymbol{v}^{(k)}\|_2 + \|\boldsymbol{u}^{(k+1)} - \boldsymbol{u}^{(k)}\|_2\Big). \tag{3.15}$$

For any $\eta \in [0,\, 1)$ and let $\gamma > 1$ be a constant, we conditionally update ρ_k according to the followings:

- If $\Delta_{k+1} \geq \eta\Delta_k$, then $\rho_{k+1} = \gamma\rho_k$.
- If $\Delta_{k+1} < \eta\Delta_k$, then $\rho_{k+1} = \rho_k$.

The adaptive update scheme is inspired from [74], which was originally used to accelerate ADMM algorithms for convex problems. It is different from the residual balancing technique commonly used in ADMM, e.g., [24], as Δ_{k+1} sums of all primal and dual residues instead of treating them individually. Our experience shows that the proposed scheme is more robust than residual balancing because the denoiser could potentially generate nonlinear effects to the residuals. Algorithm 1 shows the overall Plug-and-Play ADMM.

Remark 1 (Comparison with [25]) *In the original Plug-and-Play ADMM by [25], the update scheme is $\rho_k = \rho$ for some constant ρ. This is valid when the denoiser $\mathcal{D}_\sigma$ is non-expansive and has symmetric gradient. However, for general denoisers which could be expansive, the update scheme for ρ_k becomes crucial to the convergence. (See discussion about non-expansiveness in Section 3.2.2.)*

Remark 2 (Role of σ_k) *Many denoising algorithms nowadays such as BM3D and non-local means require one major parameter* [1]*, typically an estimate of the noise level, to control the strength of the denoiser. In our algorithm, the parameter σ_k in* (3.11) *is reminiscent to the noise level. However, unlike BM3D and non-local means where σ_k is directly linked to the standard deviation of the i.i.d. Gaussian noise, in Plug-and-Play ADMM we treat σ_k simply as a tunable knob to control the amount of denoising because the residue $(\boldsymbol{v} - \widetilde{\boldsymbol{v}}^{(k)})$ at the kth iterate is not exactly Gaussian. The adoption of the Gaussian denoiser $\mathcal{D}_{\sigma_k}$ is purely based on the formal equivalence between* (3.8) *and a Gaussian denoising problem.*

Remark 3 (Role of λ) *In this work, we assume that the parameter λ is pre-defined by the user and is fixed. Its role is similar to the regularization parameter in the conventional ADMM problem. Tuning λ can be done using external tools such as cross validation [75] or SURE [76].*

3.2.2 Global and Fixed Point Convergence

Before we discuss the convergence behavior, we clarify two types of convergence.

We refer to the type of convergence in the conventional ADMM as *global convergence*, i.e., convergence in primal residue, primal objective and dual variables. To ensure global convergence, one sufficient condition is that g is convex, proper and closed [24]. For Plug-and-Play ADMM, a sufficient condition is that $\mathcal{D}_\sigma$ has symmetric gradient and is non-expansive [26]. In this case, g exists due to a proximal

[1]A denoising algorithm often involves many other "internal" parameters. However, as these internal parameters do not have direct interaction with the ADMM algorithm, in this work we keep all internal parameters in their default settings to simplify the analysis.

mapping theorem of Moreau [77]. However, proving non-expansive denoisers could be difficult as it requires

$$\|\mathcal{D}_\sigma(\boldsymbol{x}) - \mathcal{D}_\sigma(\boldsymbol{y})\|^2 \le \kappa \|\boldsymbol{x} - \boldsymbol{y}\|^2$$

for *any* $\boldsymbol{x}$ and $\boldsymbol{y}$, with $\kappa \le 1$. Even for algorithms as simple as non-local means, one can verify numerically that there exists pairs $(\boldsymbol{x}, \boldsymbol{y})$ that would cause $\kappa > 1$. In the Appendix we demonstrate a counter example.

Since $\mathcal{D}_\sigma$ can be arbitrary and we do not even know the existence of g, we consider fixed point convergence instead. Fixed point convergence guarantees that a nonlinear algorithm can enter into a steady state asymptotically. In nonlinear dynamical systems, these limit points are referred to as the stable-fixed-points. For any initial guess lying in a region called the basin of attraction the algorithm will converge [78]. For Plug-and-Play ADMM, we conjecture that fixed point convergence is the best we can ask for unless further assumptions are made on the denoisers.

3.2.3 Convergence Analysis of Plug-and-Play ADMM

We define the class of bounded denoisers.

Definition 3.2.1 *(Bounded Denoiser). A* bounded denoiser *with a parameter σ is a function $\mathcal{D}_\sigma : \mathbb{R}^n \to \mathbb{R}^n$ such that for any input $\boldsymbol{x} \in \mathbb{R}^n$,*

$$\|\mathcal{D}_\sigma(\boldsymbol{x}) - \boldsymbol{x}\|^2/n \le \sigma^2 C, \tag{3.16}$$

for some universal constant C independent of n and σ.

Bounded denoisers are asymptotically invariant in the sense that it ensures $\mathcal{D}_\sigma \to \mathcal{I}$ (i.e., the identity operator) as $\sigma \to 0$. It is a weak condition which we expect most denoisers to have. The asymptotic invariant property of a bounded denoiser prevents trivial mappings from being considered, e.g., $\mathcal{D}_\sigma(\boldsymbol{x}) = 0$ for all $\boldsymbol{x}$.

Remark 4 *It would be useful to compare a bounded denoiser with a "proper denoiser" defined in [59]. A proper denoiser* $\widetilde{\mathcal{D}}_\sigma$ *is a mapping that denoises a noisy input* $\boldsymbol{x}+\sigma\boldsymbol{\epsilon}$ *with the property that*

$$\mathbb{E}\left[\|\widetilde{\mathcal{D}}_\sigma(\boldsymbol{x}+\sigma\boldsymbol{\epsilon})-\boldsymbol{x}\|^2/n\right] \leq \kappa\sigma^2, \tag{3.17}$$

for any $\kappa < 1$*, where* $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$ *is the i.i.d. Gaussian noise. Note that in* (3.17)*, we require the input to be a deterministic signal* $\boldsymbol{x}$ *plus an i.i.d. Gaussian noise. Moreover, the parameter must match with the noise level. In contrast, a bounded denoiser can take any input and any parameter.*

Besides the conditions on $\mathcal{D}_\sigma$ we also assume that the negative log-likelihood function f has bounded gradients:

Assumption 1 *We assume that* $f : [0,1]^n \to \mathbb{R}$ *has bounded gradients. That is, for any* $\boldsymbol{x} \in [0,1]^n$*, there exists* $L < \infty$ *such that* $\|\nabla f(\boldsymbol{x})\|_2/\sqrt{n} \leq L$.

Example 1 *Let* $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|_2^2$ *for* $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ *with eigenvalues bounded between 0 and 1. The gradient of* f *is* $\nabla f(\boldsymbol{x}) = 2\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y})$ *and*

$$\|\nabla f(\boldsymbol{x})\|_2/\sqrt{n} \leq 2\lambda_{\max}(\boldsymbol{A})^2(\|\boldsymbol{x}\|_2+\|\boldsymbol{y}\|_2)/\sqrt{n}.$$

The main convergence result of this work is as follows.

Theorem 3.2.1 *(Fixed Point Convergence of Plug-and-Play ADMM). Under Assumption 1 and for any bounded denoiser* $\mathcal{D}_\sigma$*, the iterates of the Plug-and-Play ADMM defined in Algorithm 1 demonstrates a fixed-point convergence. That is, there exists* $(\boldsymbol{x}^*, \boldsymbol{v}^*, \boldsymbol{u}^*)$ *such that* $\|\boldsymbol{x}^{(k)}-\boldsymbol{x}^*\|_2 \to 0$*,* $\|\boldsymbol{v}^{(k)}-\boldsymbol{v}^*\|_2 \to 0$ *and* $\|\boldsymbol{u}^{(k)}-\boldsymbol{u}^*\|_2 \to 0$ *as* $k \to \infty$.

Proof See Appendix B. ■

Intuitively, what Theorem 5.2.1 states is that as $k \to \infty$, the continuation scheme forces $\rho_k \to \infty$. Therefore, the inversion in (3.10) and the denoising in (3.11) have reducing influence as ρ_k grows. Hence, the algorithm converges to a fixed point. Theorem 5.2.1 also ensures that $\boldsymbol{x}^{(k)} \to \boldsymbol{v}^{(k)}$ which is an important property of the original Plug-and-Play ADMM algorithm [26]. The convergence of $\boldsymbol{x}^{(k)} \to \boldsymbol{v}^{(k)}$ holds because $\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + (\boldsymbol{x}^{(k+1)} - \boldsymbol{v}^{(k+1)})$ converges. In practice, experimentally we observe that if the algorithm is terminated early to reduce the runtime, then $\boldsymbol{v}^{(k)}$ tends to provide a slightly better solution.

3.2.4 Stopping Criteria

Since we are seeking for fixed point convergence, a natural stopping criteria is to determine if $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_2$, $\|\boldsymbol{v}^{(k+1)} - \boldsymbol{v}^{(k)}\|_2$ and $\|\boldsymbol{u}^{(k+1)} - \boldsymbol{u}^{(k)}\|_2$ are sufficiently small. Following the definition of Δ_{k+1} in (3.15), we choose to terminate the iteration when

$$\begin{aligned}\Delta_{k+1} \overset{\text{def}}{=} \frac{1}{\sqrt{n}}\Big(&\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_2 + \|\boldsymbol{v}^{(k+1)} - \boldsymbol{v}^{(k)}\|_2 \\ &+ \|\boldsymbol{u}^{(k+1)} - \boldsymbol{u}^{(k)}\|_2\Big) \leq \texttt{tol}\end{aligned} \tag{3.18}$$

for some tolerance level `tol`. Alternatively, we can also terminate the algorithm when

$$\max\Big\{\epsilon_1, \epsilon_2, \epsilon_3\Big\} \leq \texttt{tol}/3,$$

where $\epsilon_1 = \|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_2/\sqrt{n}$, $\epsilon_2 = \|\boldsymbol{v}^{(k+1)} - \boldsymbol{v}^{(k)}\|_2/\sqrt{n}$ and $\epsilon_3 = \|\boldsymbol{u}^{(k+1)} - \boldsymbol{u}^{(k)}\|_2/\sqrt{n}$.

In practice, the tolerance level does not need to be extremely small in order to achieve good reconstruction quality. In fact, for many images we have tested, setting $\texttt{tol} \approx 10^{-3}$ is often sufficient. Figure 3.1 provides a justification. In this experiment, we tested an image super-resolution problem for 10 testing images (See Configuration 3 in Section 3.4.1 for details). It can be observed that the PSNR becomes steady when `tol` drops below 10^{-3}. Moreover, size of the image does not seem to be an influencing factor. Smaller images such as `Cameraman256`, `House256` and `Peppers256` shows similar characteristics as bigger images. The more influencing factor is the combination of the update ratio γ and the initial value ρ_0. However, unless γ is close to 1 and ρ_0 is extremely small (which does not yield good reconstruction anyway), our experience is that setting `tol` at 10^{-3} is usually valid for $\gamma \in (1,2)$ and $\rho_0 \in (10^{-5}, 10^{-2})$.

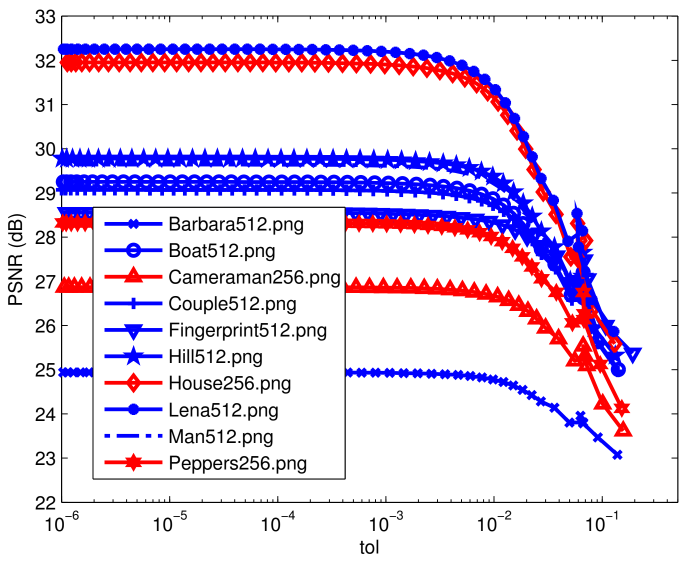


Fig. 3.1.: Stopping criteria. The PSNR drops as the tolerance level increases. However, regardless of the size of the images, the PSNR becomes steady when `tol` $\approx 10^{-3}$.

3.2.5 Initial Parameter ρ_0

The choice of the initial parameter ρ_0 requires some tuning but is typically good for $\rho_0 \in (10^{-5}, 10^{-2})$. Figure 3.2 shows the behavior of the algorithm for different values of ρ_0, ranging from 10^0 to 10^{-4}. We compare the original Plug-and-Play ADMM (i.e., with constant $\rho_k = \rho_0$, the red lines), monotone update rule (i.e., $\rho_{k+1} = \gamma\rho_k$, the blue lines), and the adaptive update rule (the black lines). We make two observations regarding the difference between the proposed algorithm and the original Plug-and-Play ADMM [26]:

- **Stability**: The original Plug-and-Play ADMM [26] requires a highly precise ρ_0. For example, in Figure 3.2 the best PSNR is achieved when $\rho_0 = 1$; When ρ_0 is less than 10^{-2}, the PSNR becomes very poor. The proposed algorithm works for a much wider range of ρ_0.
- **Final PSNR**: The proposed Plug-and-Play ADMM is a generalization of the original Plug-and-Play ADMM. The added degrees of freedom are the new parameters (ρ_0, γ, η). The original Plug-and-Play ADMM is a special case when $\gamma = 1$. Therefore, for optimally tuned parameters, the proposed Plug-and-Play ADMM is always better than or equal to the original Plug-and-Play ADMM. This is verified in Figure 3.2, which shows that the best PSNR is attained by the proposed method.

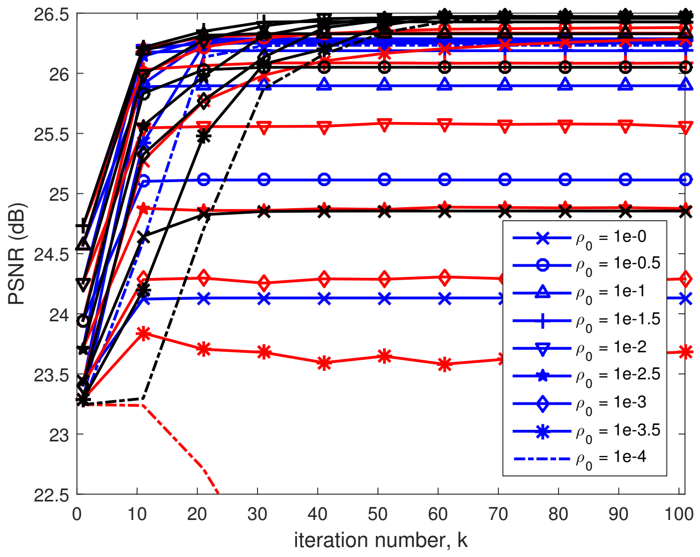


Fig. 3.2.: Influence of ρ_0. Red curves represent the original method in [26]; Blue curves represent the monotone update rule; Black curves represent the adaptive update rule. Note the diversified behavior of the red curve, which implies that a precise ρ_0 is required. The blue and black curves are more robust.

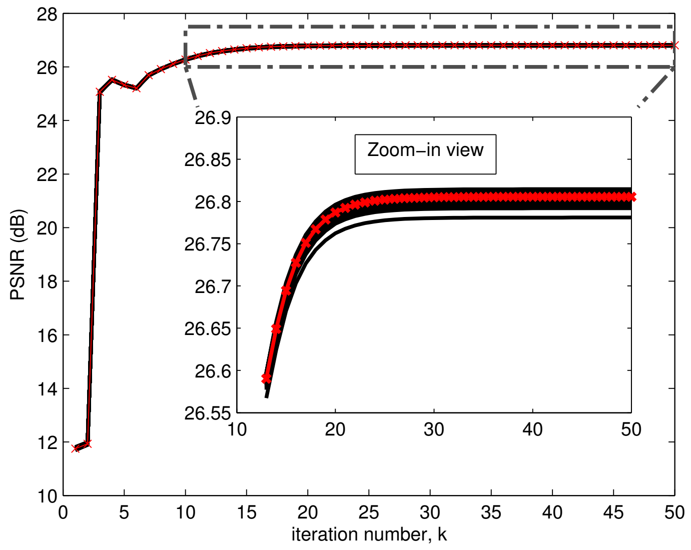


Fig. 3.3.: Influence of the initial point $\boldsymbol{x}^{(0)}$. We start the algorithm with 100 different initial guesses $\boldsymbol{x}^{(0)}$ where each is a uniformly random vector drawn from $[0, 1]^n$. Over these 100 random realizations we plot the average (red line). Note the small fluctuation of the PSNR at the limit.

3.2.6 Initial Guesses

The initial guesses $\boldsymbol{x}^{(0)}$, $\boldsymbol{v}^{(0)}$ and $\boldsymbol{u}^{(0)}$ have less impact to the final PSNR. This can be seen from Figure 3.3. In this experiment, we randomly draw 100 initial guesses $\boldsymbol{x}^{(0)}$ from a uniform distribution in $[0, 1]^n$. The auxiliary variable is set as $\boldsymbol{v}^{(0)} = \boldsymbol{x}^{(0)}$, and the Lagrange multiplier $\boldsymbol{u}^{(0)}$ is 0. As shown in Figure 3.3, the initial guesses do not cause significant difference in term of PSNR at the limit. The standard deviation at the limit is 0.0059 dB, implying that with 99.7% probability (3 standard deviations) the PSNR will stay within ± 0.0176 dB from its average.

3.3 Applications

As we discussed in the introduction, Plug-and-Play ADMM algorithm has a wide range of applications. However, in order to enable the denoising step, Plug-and-Play ADMM uses a specific variable splitting strategy. The challenge it brings, therefore, is whether we can solve the subsequent subproblems efficiently. The purpose of this section is to address this issue by presenting two applications where fast implementation can be achieved.

3.3.1 Application 1: Image Super-resolution

Image super-resolution can be described by a linear forward model with two operations: an anti-aliasing filter and a subsampling process. The function $f(\boldsymbol{x})$ is quadratic in the form

$$f(\boldsymbol{x}) = \|\boldsymbol{S}\boldsymbol{H}\boldsymbol{x} - \boldsymbol{y}\|^2, \tag{3.19}$$

where the matrix $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ is a circulant matrix representing the convolution for the anti-aliasing filter. The matrix $\boldsymbol{S} \in \mathbb{R}^{m\times n}$ is a binary sampling matrix, where the rows are subsets of the identity matrix. By defining $\boldsymbol{G} \overset{\text{def}}{=} \boldsymbol{S}\boldsymbol{H}$ we recognize that when substituting (3.19) into (5.2a), the f-subproblem becomes (we dropped the iteration number k to simplify notation)

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathbb{R}^n}{\operatorname{argmin}} \;\; \|\boldsymbol{G}\boldsymbol{x} - \boldsymbol{y}\|^2 + \frac{\rho}{2}\|\boldsymbol{x} - \widetilde{\boldsymbol{x}}\|^2. \tag{3.20}$$

Consequently, the solution is the pseudo-inverse

$$\widehat{\boldsymbol{x}} = (\boldsymbol{G}^T\boldsymbol{G} + \rho\boldsymbol{I})^{-1}(\boldsymbol{G}^T\boldsymbol{y} + \rho\widetilde{\boldsymbol{x}}). \tag{3.21}$$

For special cases of $\boldsymbol{H}$ and $\boldsymbol{S}$, (3.21) has known efficient implementation as follows.

Example 2 (Non-blind deblurring [18, 45, 73]) *Non-blind deblurring is a special case when $\boldsymbol{S} = \boldsymbol{I}$. In this case, since $\boldsymbol{H}$ is circulant which is diagonalizable by the discrete Fourier transform matrices,* (3.21) *can be efficiently implemented by*

$$\widehat{\boldsymbol{x}} = \mathcal{F}^{-1}\left\{\frac{\overline{\mathcal{F}(h)}\mathcal{F}(\boldsymbol{y}) + \rho\mathcal{F}(\widetilde{\boldsymbol{x}})}{|\mathcal{F}(h)|^2 + \rho}\right\}, \tag{3.22}$$

where $\mathcal{F}(\cdot)$ is the Fourier transform operator, h is the finite impulse response filter representing the blur kernel, $\overline{(\cdot)}$ is the complex conjugate, and the multiplication/division are element-wise operations.

Example 3 (Interpolation [41, 47, 48, 79]) *Image interpolation is a special case when $\boldsymbol{H} = \boldsymbol{I}$. In this case, since $\boldsymbol{S}^T\boldsymbol{S}$ is a diagonal matrix with binary entries,* (3.21) *can be efficiently implemented using an element-wise division:*

$$\widehat{\boldsymbol{x}} = (\boldsymbol{S}^T\boldsymbol{y} + \rho\widetilde{\boldsymbol{x}})./(\boldsymbol{s} + \rho), \tag{3.23}$$

where $\boldsymbol{s} = \operatorname{diag}\left\{\boldsymbol{S}^T\boldsymbol{S}\right\}$.

3.3.2 Polyphase Implementation for Image Super-Resolution

When $\boldsymbol{G} = \boldsymbol{S}\boldsymbol{H}$, solving the f-subproblem becomes non-trivial because $\boldsymbol{H}^T\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{H}$ is neither diagonal nor diagonalizable by the Fourier transform. In literature, the two most common approaches are to introduce multi-variable split to bypass (3.21) (e.g., [45, 80]) or use an inner conjugate gradient to solve (3.21) (e.g., [55]). However, multi-variable splitting requires additional Lagrange multipliers and internal parameters. It also generally leads to slower convergence than single-variable split. Inner conjugate gradient is computationally expensive as it requires an iterative solver. In what follows, we show that when $\boldsymbol{S}$ is the standard K-fold downsampler (i.e., sub-sample the spatial grid uniformly with a factor K along horizontal and vertical directions), and when $\boldsymbol{H}$ is a circular convolution, it is possible to derive a closed-form solution [2].

[2]We assume the boundaries are circularly padded. In case of other types boundary conditions or unknown boundary conditions, we can pre-process the image by padding the boundaries circularly. Then, after the super-resolution algorithm we crop the center region. The alternative approach is to consider multiple variable split as discussed in [81].

Our closed form solution begins by considering the Sherman-Morrison-Woodbury identity, which allows us to rewrite (3.21) as

$$\widehat{\boldsymbol{x}} = \rho^{-1}\boldsymbol{b} - \rho^{-1}\boldsymbol{G}^T(\rho\boldsymbol{I} + \boldsymbol{G}\boldsymbol{G}^T)^{-1}\boldsymbol{G}\boldsymbol{b}, \tag{3.24}$$

where $\boldsymbol{b} \overset{\text{def}}{=} \boldsymbol{G}^T\boldsymbol{y} + \rho\widetilde{\boldsymbol{x}}$. Note that if $\boldsymbol{S} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ with $m < n$, then (3.24) only involves a $m \times m$ inverse, which is smaller than the $n \times n$ inverse in (3.21).

The more critical step is the following observation. We note that the matrix $\boldsymbol{G}\boldsymbol{G}^T$ is given by

$$\boldsymbol{G}\boldsymbol{G}^T = \boldsymbol{S}\boldsymbol{H}\boldsymbol{H}^T\boldsymbol{S}^T.$$

Since $\boldsymbol{S}$ is a K-fold downsampling operator, $\boldsymbol{S}^T$ is a K-fold upsampling operator. Defining $\widetilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{H}^T$, which can be implemented as a convolution between the blur kernel h and its time-reversal, we observe that $\boldsymbol{S}\widetilde{\boldsymbol{H}}\boldsymbol{S}^T$ is a "upsample-filter-downsample" sequence. This idea is illustrated in Figure 3.4.

We next study the polyphase decomposition [82] of Figure 3.4. Polyphase decomposition allows us to write

$$\widetilde{H}(z) = \sum_{k=0}^{K-1} z^{-k}\widetilde{H}_k(z^K), \tag{3.25}$$

where $\widetilde{H}(z)$ is the z-transform representation of the blur matrix $\widetilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{H}^T$, and $\widetilde{H}_k(z^K)$ is the kth polyphase component of $\widetilde{H}(z)$. Illustrating (3.25) using a block diagram, we show in Figure 3.5 the decomposed structure of Figure 3.4. Then, using Noble identity [82], the block diagram on the left hand side of Figure 3.5 becomes the one shown on the right hand side. Since for any $k > 1$, placing a delay z^{-k} between an upsampling and a downsampling operator leads to a zero, the overall system simplifies to a finite impulse response filter $\widetilde{H}_0(z)$, which can be pre-computed.

We summarize this by the following proposition.

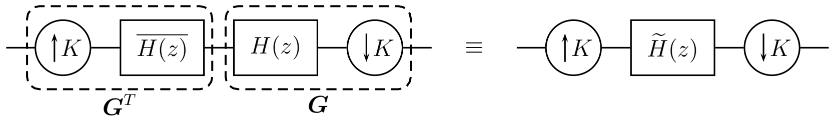


Fig. 3.4.: [Left] Block diagram of the operation $\boldsymbol{G}\boldsymbol{G}^T$. [Right] The equivalent system, where $\widetilde{H}(z) = \overline{H(z)}H(z)$.

Algorithm 2 Compute the 0th polyphase component.

```
Input: $h$: the blur kernel, and $K$: downsampling factor
Let $\widetilde{h} = \mathcal{F}^{-1}(\mathcal{F}(h)\overline{\mathcal{F}(h)})$ be the convolved filter.
Output: $\widetilde{h}_0 = (\downarrow_K)(\widetilde{h})$.
```

Proposition 3.3.1 *The operation of $\boldsymbol{S}\boldsymbol{H}\boldsymbol{H}^T\boldsymbol{S}^T$ is equivalent to applying a finite impulse response filter $\widetilde{H}_0(z)$, which is the 0th polyphase component of the filter $\boldsymbol{H}\boldsymbol{H}^T$.*

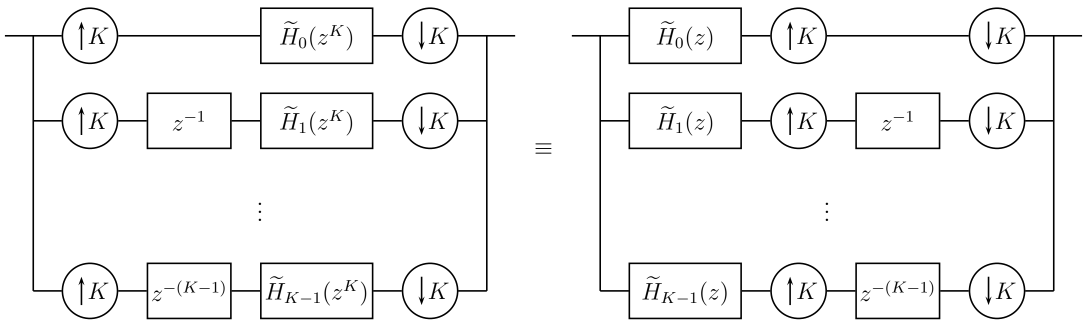


Fig. 3.5.: [Left] Polyphase decomposition of $\widetilde{H}(z)$. [Right] Equivalent representation.

To implement the 0th polyphase component, we observe that it can be done by downsampling the convolved filter $\widetilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{H}^T$. This leads to the procedure illustrated in Algorithm 2.

The implication of Proposition 3.3.1 is that since $\boldsymbol{G}\boldsymbol{G}^T$ is equivalent to a finite impulse response filter $\widetilde{h}_0$, (3.24) can be implemented in closed-form using the Fourier transform:

$$\boldsymbol{x} = \rho^{-1}\boldsymbol{b} - \rho^{-1}\boldsymbol{G}^T\left(\mathcal{F}^{-1}\left\{\frac{\mathcal{F}(\boldsymbol{G}\boldsymbol{b})}{|\mathcal{F}(\widetilde{h}_0)|^2 + \rho}\right\}\right), \tag{3.26}$$

where we recall that $\boldsymbol{b} = \boldsymbol{G}^T\boldsymbol{y} + \rho\widetilde{\boldsymbol{x}}$.

The effectiveness of the proposed closed-form solution can be seen from Figure 3.6. In this figure, we compare with a brute force conjugate gradient method presented in [55]. When $\boldsymbol{H}$ satisfies periodic boundary conditions, the closed-form solution is *exact*. If the boundaries are not periodic, alternative solutions can be considered, e.g., [83].

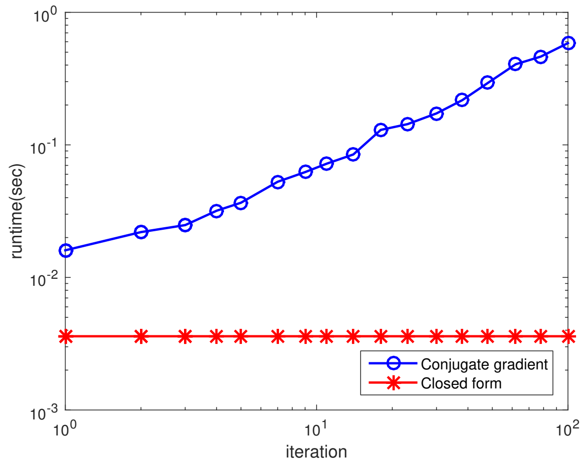


Fig. 3.6.: Runtime of conjugate gradient for solving a $\boldsymbol{x}$-subproblem. Note the non-iterative nature of the closed-form solution.

3.3.3 Application 2: Single Photon Imaging

The second application is a single photon imaging problem using quanta image sensors (QIS) [84]. Using ADMM for QIS was previously reported in [85]. Here, we show how the Plug-and-Play ADMM can be used for the problem.

QIS is a spatial oversampling device. A QIS is composed to many tiny single photon detectors called jots. In each unit space, K jots are used to acquire light corresponding to a pixel in the usual sense (e.g., a pixel in a CMOS sensor). Therefore, for an image of n pixels, a total number of nK jots are required. By assuming homogeneous distribution of the light within each pixel, we consider a simplified QIS imaging model which relates the underlying image $\boldsymbol{x} \in \mathbb{R}^n$ and the actual photon arrival rate at the jots $\boldsymbol{s} \in \mathbb{R}^{nK}$ as

$$\boldsymbol{s} = \alpha \boldsymbol{G} \boldsymbol{x},$$

where the matrix $\boldsymbol{G} \in \mathbb{R}^{nK \times n}$ is

$$\boldsymbol{G} = \frac{1}{K} \begin{bmatrix} \mathbf{1}_{K\times 1} & \mathbf{0}_{K\times 1} & \dots & \mathbf{0}_{K\times 1} \\ \mathbf{0}_{K\times 1} & \mathbf{1}_{K\times 1} & \dots & \mathbf{0}_{K\times 1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{K\times 1} & \mathbf{0}_{K\times 1} & \dots & \mathbf{1}_{K\times 1} \end{bmatrix}, \tag{3.27}$$

and α is a sensor gain. Given $\boldsymbol{s}$, the photons arriving at the sensors follow a Poisson distribution with a rate given by $\boldsymbol{s}$. Let Z_i be the random variable denoting the number of photons at jot i, we have

$$p(z_i) = \frac{s_i^{-z_i} e^{-s_i}}{z_i!}, \quad i = 1, \dots, nK. \tag{3.28}$$

The final QIS output, Y_i, is a binary bit resulted from truncating Z_i using a threshold q. That is,

$$Y_i = \begin{cases} 1, & \text{if } Z_i \geq q, \\ 0, & \text{if } Z_i < q. \end{cases}$$

When $q = 1$, the probability of observing $Y_i = y_i$ given s_i is

$$p(y_i \,|\, s_i) = \begin{cases} e^{-s_i}, & \text{if } y_i = 0, \\ 1 - e^{-s_i}, & \text{if } y_i = 1. \end{cases}$$

The recovery goal is to estimate $\boldsymbol{x}$ from the observed binary bits $\boldsymbol{y}$. Taking the negative log and summing over all pixels, the function f is defined as

$$\begin{aligned} f(\boldsymbol{x}) &\overset{\text{def}}{=} p(\boldsymbol{y} \,|\, \boldsymbol{s}) = \sum_{i=1}^{nK} -\log\ p(y_i \,|\, s_i) \\ &= \sum_{i=1}^{nK} -\log\left((1-y_i)e^{-s_i} + y_i\left(1-e^{-s_i}\right)\right) \\ &= \sum_{j=1}^{n} -K_j^0 \log(e^{-\frac{\alpha x_j}{K}}) - K_j^1 \log(1-e^{-\frac{\alpha x_j}{K}}), \end{aligned} \tag{3.29}$$

where $K_j^1 = \sum_{i=1}^{K} y_{(j-1)K+i}$ is the number of ones in the jth unit pixel, and $K_j^0 = \sum_{i=1}^{K}(1-y_{(j-1)K+i})$ is the number of zeros in the jth unit pixel. (Note that for any j, $K_j^1 + K_j^0 = K$.) Consequently, substituting (3.29) into (3.10) yields the f-subproblem

$$\min_{\boldsymbol{x}} \sum_{j=1}^{n} -K_j^0 \log(e^{-\frac{\alpha x_j}{K}}) - K_j^1 \log(1-e^{-\frac{\alpha x_j}{K}}) + \frac{\rho}{2}(x_j - \widetilde{x}_j)^2. \tag{3.30}$$

Since this optimization is separable, we can solve each individual variable x_j independently. Thus, for every j, we solve a single-variable optimization by taking derivative with respect to x_j and setting to zero, yielding

$$Ke^{-\frac{\alpha x_j}{K}}(\alpha + \rho(x_j - \widetilde{x}_j)) = \alpha K_j^0 + \rho K(x_j - \widetilde{x}_j),$$

which is a one-dimensional root finding problem. By constructing an offline lookup table in terms of K_0, ρ and $\widetilde{x}_j$, we can solve (3.30) efficiently.

3.4 Experimental Results

In this section we present the experimental results. For consistency we use BM3D in all experiments, although other bounded denoisers will also work. We shall not compare Plug-and-Play ADMM using different denoisers as it is not the focus of the work.

3.4.1 Image Super-Resolution

We consider a set of 10 standard test images for this experiment as shown in Figure 3.7. All images are gray-scaled, with sizes between 256×256 and 512×512. Four sets of experimental configurations are studied, and are shown in Table 3.1.

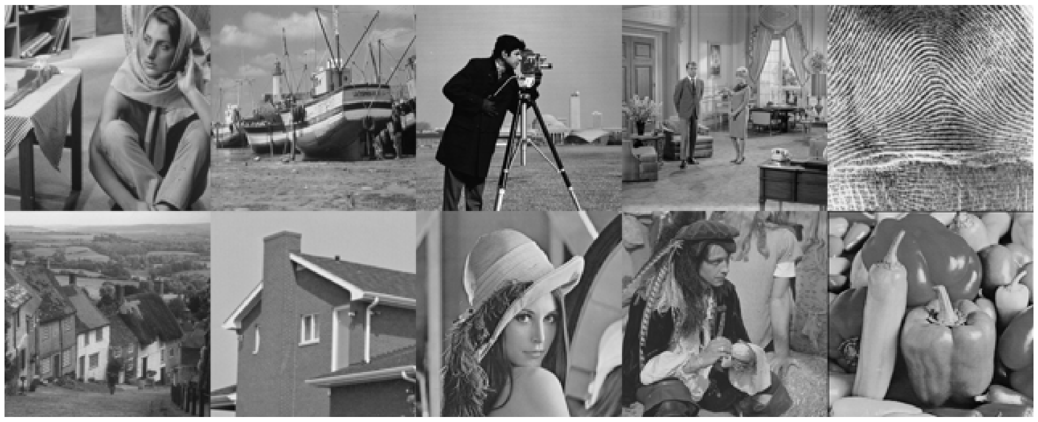


Fig. 3.7.: 10 testing images for the experiment.

Table 3.1.: Configurations and parameters.

Config	Description
1	$K = 2$, $\boldsymbol{H}$ = bicubic, Noise = 0
2	$K = 4$, $\boldsymbol{H}$ = bicubic, Noise = 0
	$\rho_0 = 1.3 \times 10^{-5}$, $\gamma = 2.5$, $\lambda = 10^{-5}$
3	$K = 2$, $\boldsymbol{H}$ = Gaussian of std 1, Noise = 5/255
4	$K = 4$, $\boldsymbol{H}$ = Gaussian of std 1, Noise = 5/255
	$\rho_0 = 1 \times 10^{-5}$, $\gamma = 1.2$, $\lambda = 10^{-4}$

We compared the proposed algorithm with several existing super-resolution algorithms. These methods include the deep convolutional neural network method (DCNN) by Dong et al. [49], the statistical patch-based sparse representation method (SPSR) by Peleg and Elad [50], the transformed self-exemplar method (TSE) by Huang et al. [86], the classical sparse representation method for super-resolution (SR) by Yang et al. [52], and the Gaussian process regression method (GPR) by He and Siu [51]. Among these methods, we note that TSE and GPR are single image meth-

Table 3.2.: Image Super Resolution Results. When noise is present, the PSNR values are averaged over 5 random realizations of the noise pattern. Gray color rows represent external database methods.

	Images											
	1	2	3	4	5	6	7	8	9	10	Dataset	Avg STD
Size	512^2	512^2	256^2	512^2	512^2	512^2	256^2	512^2	512^2	256^2	Avg	per image
Factor: $\times 2$; Anti-aliasing Filter: Bicubic; Noise: 0												
DCNN [49]	25.71	31.83	28.79	31.13	32.73	**32.60**	35.11	36.34	**33.10**	33.05	32.04	–
SR [52]	**25.87**	31.51	27.92	30.94	33.02	32.46	34.79	36.14	32.80	32.67	31.81	–
SPSR [50]	25.71	31.49	27.85	31.00	33.30	32.35	34.37	36.18	32.67	32.66	31.76	–
TSE [86]	25.66	31.64	28.17	31.01	32.88	32.45	34.78	36.22	32.88	**33.29**	31.90	–
GPR [51]	24.99	29.99	26.44	29.50	30.36	31.33	33.08	34.19	31.09	30.98	30.20	–
Ours - M	25.87	**31.85**	**28.81**	**31.42**	**33.63**	32.56	**35.30**	**36.45**	32.86	33.06	**32.18**	–
Ours - A	25.74	31.68	28.44	31.27	33.49	32.39	35.20	36.02	32.65	32.65	31.95	–
Factor: $\times 4$; Anti-aliasing Filter: Bicubic; Noise: 0												
DCNN [49]	23.93	26.84	23.76	26.08	24.18	28.32	29.48	30.45	28.17	27.88	26.91	–
SR [52]	23.91	26.39	23.43	26.02	24.44	28.17	29.15	30.19	27.94	27.42	26.71	–
SPSR [50]	23.90	26.49	23.42	26.02	25.11	28.14	29.22	30.24	27.85	27.62	26.80	–
TSE [86]	23.90	26.62	23.83	26.10	24.70	28.06	30.03	30.29	28.03	27.97	26.95	–
GPR [51]	23.55	25.47	22.54	25.27	22.33	27.79	27.61	28.74	26.76	25.79	25.58	–
Ours - M	**23.99**	26.87	23.82	**26.32**	25.54	**28.35**	30.16	**30.74**	**28.17**	**28.26**	27.22	–
Ours - A	23.99	**26.87**	**23.83**	26.33	**25.58**	28.29	**30.48**	30.62	28.12	28.22	**27.23**	–
Factor: $\times 2$; Anti-aliasing Filter: Gaussian 9×9, $\sigma = 1$; Noise: 5/255												
DCNN [49]	23.61	26.30	23.75	26.21	23.79	27.50	27.84	28.15	27.05	26.07	26.03	0.0219
SR [52]	23.61	26.25	23.71	26.15	23.80	27.41	27.71	28.07	26.99	26.10	25.98	0.0221
SPSR [50]	23.75	26.57	23.88	26.47	23.91	27.80	28.19	28.58	27.33	26.39	26.29	0.0208
TSE [86]	23.57	26.22	23.65	26.12	23.79	27.34	27.55	28.00	26.93	26.11	25.93	0.0238
GPR [51]	23.82	26.81	23.91	26.63	24.05	28.38	29.16	29.54	27.78	26.76	26.68	0.0170
Ours - M	**24.64**	**29.41**	**26.73**	**29.22**	**28.82**	**29.82**	**32.65**	**32.76**	**29.66**	**30.10**	**29.38**	0.0267
Ours - M*	24.01	27.09	24.17	27.00	25.03	28.46	29.32	29.78	27.85	26.99	26.97	0.0178
Factor: $\times 4$; Anti-aliasing Filter: Gaussian 9×9, $\sigma = 1$; Noise: 5/255												
DCNN [49]	20.72	21.30	18.91	21.68	16.10	23.39	22.33	22.99	22.46	20.23	21.01	0.0232
SR [52]	20.67	21.30	18.86	21.51	16.37	23.15	22.19	22.85	22.26	20.33	20.95	0.0212
SPSR [50]	20.85	21.58	19.18	21.85	16.59	23.52	22.42	23.05	22.53	20.50	21.21	0.0217
TSE [86]	20.59	21.24	18.80	21.49	16.40	23.14	22.21	22.78	22.21	20.30	20.92	0.0252
GPR [51]	21.55	22.68	19.90	22.77	17.70	24.57	23.51	24.37	23.63	21.35	22.20	0.0313
Ours - M	**23.62**	**25.75**	**23.06**	**25.30**	**24.48**	**27.17**	**29.14**	**29.42**	**26.86**	**26.86**	**26.17**	0.0223
Ours - M*	21.21	22.12	19.43	22.43	16.90	24.37	23.13	23.95	23.39	21.13	21.81	0.0253

ods whereas DCNN, SPSR and SR require training using external databases. The

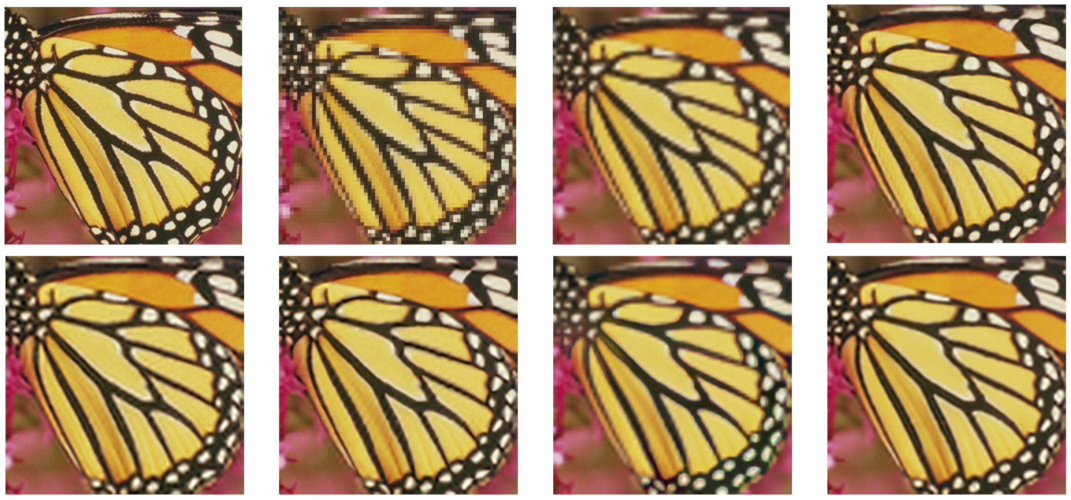


Fig. 3.8.: Image Super Resolution Results. [Top](from left to right). Ground truth; The low resolution input; Bicubic interpolation; DCNN [49] (24.19dB). [Bottom](from left to right). SPSR [50] (22.44dB); TSE [86] (22.80dB); GPR [51] (20.81dB); Ours-M (23.49dB).

results of the experiment are shown in Table 4.3. For the proposed algorithm, we present the two update rules as Ours-M (for monotone update rule), and Ours-A (for adaptive update rule). When noise is present in the simulation, we conduct a Monte-Carlo simulation over 5 random realizations. In this case, the reported PSNR values are the average over the random realizations. The per image standard deviation is reported in the last column of Table 4.3 (if applicable).

For configurations 3 and 4 when we use a Gaussian anti-aliasing filter, we observe that not all existing methods can handle such case as the training part of those algorithms was performed on a bicubic model. Therefore, for fair comparison, we present two versions of the proposed algorithm. The first version Ours-M assumes the correct knowledge about the Gaussian filter, whereas the second version Ours-M* ignores such assumption and use the bicubic model for reconstruction.

From the PSNR results shown in Table 4.3, we observe very similar performance of the competing methods. For configurations 1 and 2, the proposed algorithm shows the best performance overall, although in some occasions the deep neural network [49] is better. For configurations 3 and 4, we observe a significant gap between Ours-M and the competing methods. This is caused by the model mismatch of the competing methods as the implementations provided by the authors only support the bicubic model. For fairness, we consider Ours-M* by pretending that the anti-aliasing filter is bicubic. In this case, Ours-M* still performs better than the others for configuration 3, but slightly worse than GPR [51] for configuration 4.

For visual comparison we conduct a color image experiment. In this experiment, we simulate a low resolution image by downsampling the color image by a factor 4 using a bicubic anti-aliasing filter. Then, we apply the proposed algorithm to the three color channels individually to recover the image. The result is shown in Figure 3.8. As seen, the proposed method produces better results than SPSR [50], TSE [86] and GPR [51], with slightly sharper edges and less halo artifacts. We also observe that the deep neural network [49] shows better results than that in Table 4.3. One possibility is that the training data used to train the neural network are natural images that have better correlation to Figure 3.8. However, considering the training-free nature of the proposed Plug-and-Play algorithm, losing to a well-trained neural network is not surprising.

3.4.2 Single Photon Imaging

We next consider the single-photon imaging problem. In this experiment, we consider four sets of experiments for $K = 4, 6, 8, 10$ (along horizontal and vertical directions). The sensor gain is set as $\alpha = K^2$. For comparison, we choose the two existing algorithms. The first one is the maximum likelihood estimation (MLE) method by Yang et al. [87]. For our specific choice of $\boldsymbol{G}$ in (3.27), the MLE solution has a closed-form expression. The second method is a total variation method by Chan and Lu [85,88]. This method utilizes the ADMM algorithm when solving the problem. We are aware of other existing Poisson denoising methods such as [16, 54]. However, none of these methods are directly applicable to the quantized Poisson problem.

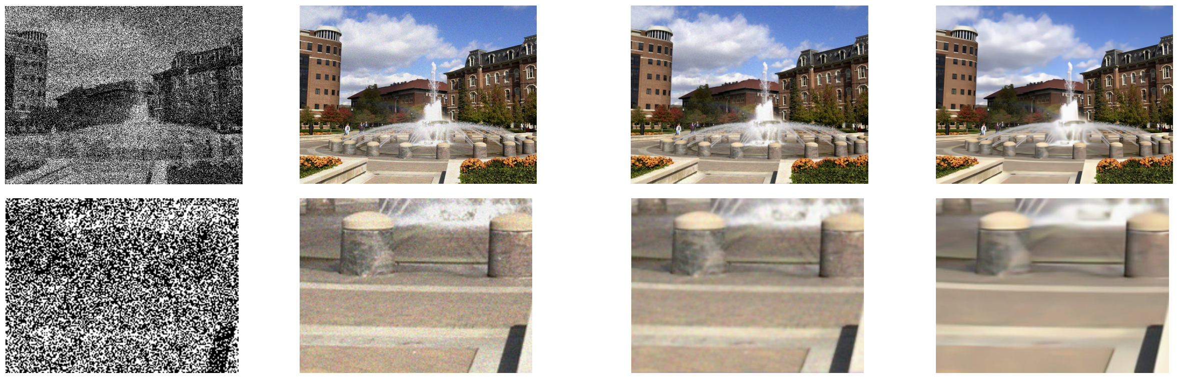
Since for this problem the observed binary pattern is a truncated Poisson random variable, we perform a Monte-Carlo simulation by repeating each case for 8 independent trials. We then report the average and the standard deviation of these 8 independent trials. As shown in Table 3.3, the standard deviation is indeed insignificant compared to the average PSNR. Here, we report the dataset average over the 10 images to ensure sufficient variability of the test. To visually compare the performance, in Figure 3.9 we show the result of a color image. In this experiment, we process the 3 color channels individually. For each channel, we simulate the photon arrivals by assuming $K = 8$. Then, we reconstruct the image using different algorithms. The result in Figure 3.9 shows that visually the proposed algorithm produces images with less noise.

Table 3.3.: Single Photon Imaging Results. The PSNR values are averaged over 8 random realizations of the photon arrivals.

	Images											
	1	2	3	4	5	6	7	8	9	10	Dataset	Avg STD
Size	512^2	512^2	256^2	512^2	512^2	512^2	256^2	512^2	512^2	256^2	Avg	per image
						$K = 4$						
Yang et al. [87]	14.80	14.18	14.68	14.39	14.28	14.90	14.21	14.48	14.78	14.59	14.53	0.0157
Chan-Lu [85]	22.59	24.76	23.63	24.74	21.47	25.65	25.38	26.42	25.35	24.78	24.48	0.0425
Ours-M	25.99	25.72	25.58	26.03	23.54	26.60	28.15	28.17	26.17	26.10	26.20	0.0821
Ours-A	**26.06**	**25.75**	**25.64**	**26.10**	**23.59**	**26.66**	**28.27**	**28.27**	**26.18**	**26.16**	**26.27**	0.0801
						$K = 6$						
Yang et al. [87]	17.94	17.27	17.67	17.61	17.29	18.22	17.22	17.62	18.00	17.65	17.65	0.0147
Chan-Lu [85]	23.97	26.25	25.53	26.26	24.47	26.66	26.75	27.32	26.58	26.40	26.02	0.0339
Ours-M	**28.34**	**27.76**	27.60	**27.91**	**25.66**	**28.30**	**30.34**	**30.06**	**27.75**	28.15	**28.19**	0.0451
Ours-A	**28.34**	27.72	**27.61**	27.84	25.62	28.25	30.28	29.86	27.71	**28.16**	28.14	0.0472
						$K = 8$						
Yang et al. [87]	20.28	19.68	20.00	20.05	19.53	20.64	19.49	19.99	20.42	19.97	20.01	0.0183
Chan-Lu [85]	25.14	27.09	26.56	27.24	25.75	27.55	27.17	27.89	27.49	27.07	26.90	0.0325
Ours-M	**29.79**	**29.07**	**29.14**	**29.25**	**27.19**	**29.55**	**31.70**	**31.43**	**28.99**	**29.52**	**29.56**	0.0527
Ours-A	29.74	29.00	29.09	29.16	27.14	29.51	31.54	31.35	28.94	29.43	29.49	0.0520
						$K = 10$						
Yang et al. [87]	22.14	21.60	21.89	21.98	21.32	22.51	21.32	21.86	22.35	21.83	21.88	0.0198
Chan-Lu [85]	26.20	27.57	27.26	27.70	26.41	27.99	27.64	28.16	27.95	27.66	27.46	0.0264
Ours-M	**30.88**	**30.19**	**30.34**	**30.31**	**28.29**	**30.48**	**32.68**	**32.29**	**29.97**	**30.56**	**30.60**	0.0386
Ours-A	30.81	30.12	30.31	30.22	28.22	30.41	32.51	32.17	29.90	30.47	30.51	0.0397

3.5 Conclusion

We presented a continuation scheme for Plug-and-Play ADMM. We showed that for any *bounded denoisers* (denoisers that asymptotically converges to the identity op-



(a) Binary input (b) Yang et al. [87] 20.16dB (c) Chan and Lu [85] 26.74dB (d) Ours-C 28.81dB

Fig. 3.9.: Single photon imaging results. The bottom row is a zoomed-in figure of the top row.

erator), the new Plug-and-Play ADMM has a provable fixed point convergence. We demonstrated two applications of the new algorithm for single image super-resolution and the single photon imaging problem. For the single image super-resolution problem, we presented a closed-form approach to solve one of the two subproblems in the ADMM algorithm. The closed-form result allows significantly faster implementation than iterative methods. Experimentally, we found that Plug-and-Play ADMM performs better than several existing methods.

Acknowledgement

We thank Professor Charles Bouman and Suhas Sreehari for useful discussions. We thank Professor Gregery Buzzard for the constructive critique about the proof of Theorem 1. We thank the anonymous reviewers for the useful feedback which significantly improves the work.

3.6 Appendix

3.6.1 Counter Example of Non-Expansive Denoiser

As mentioned in Section II.B, showing non-expansiveness of a denoiser could be difficult. Here we provide a counter example for the non-local means [69].

To show that non-local means are expansive, we only need to find a pair $(\boldsymbol{x}, \boldsymbol{y})$ such that

$$\kappa = \|\mathcal{D}_\sigma(\boldsymbol{x}) - \mathcal{D}_\sigma(\boldsymbol{y})\|^2 / \|\boldsymbol{x} - \boldsymbol{y}\|^2 > 1.$$

To construct such example, we show in Figure 3.10 a pair of $(\boldsymbol{x}, \boldsymbol{y})$ obtained through an inpainting problem using Plug-and-Play ADMM with constant ρ, i.e., $\gamma = 1$. (In fact, it does not matter how we obtain this pair of $(\boldsymbol{x}, \boldsymbol{y})$. All we need to show is that there exists $(\boldsymbol{x}, \boldsymbol{y})$ which makes $\kappa > 1$.)

The non-local means is a weighted average operation with weights

$$W_{ij} = \exp\{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/(2\sigma^2)\},$$

where $\boldsymbol{x}_i$ is the ith patch of the image $\boldsymbol{x}$. To further ensure that $\boldsymbol{W}$ is doubly stochastic so that its eigenvalues are bounded between 0 and 1, we apply Sinkhorn-Knopp [89] to $\boldsymbol{W}$ until convergence. Define this doubly stochastic matrix as $\widetilde{\boldsymbol{W}}$. Then, the denoised output is given by

$$\widetilde{\boldsymbol{x}} = \mathcal{D}_\sigma(\boldsymbol{x}) \overset{\text{def}}{=} \widetilde{\boldsymbol{W}}\boldsymbol{x}.$$

Therefore, the ratio we need to check is

$$\kappa = \|\widetilde{\boldsymbol{W}}_{\boldsymbol{x}}(\boldsymbol{x}) - \widetilde{\boldsymbol{W}}_{\boldsymbol{y}}(\boldsymbol{y})\|^2 / \|\boldsymbol{x} - \boldsymbol{y}\|^2,$$

where the subscript $(\cdot)_{\boldsymbol{x}}$ specifies the dependency of $\widetilde{\boldsymbol{W}}$ on $\boldsymbol{x}$ (or $\boldsymbol{y}$). The denoised results are shown in Figure 3.10 (c) and (d). Although it may look subtle, one can verify that $\kappa = 1.1775$ which violates the requirement of non-expansiveness. This happens because $\widetilde{\boldsymbol{W}}_{\boldsymbol{x}} \neq \widetilde{\boldsymbol{W}}_{\boldsymbol{y}}$ for $\boldsymbol{x} \neq \boldsymbol{y}$. The dependency on $\boldsymbol{x}$ and $\boldsymbol{y}$ makes the operators nonlinear, and hence makes non-expansiveness difficult to validate.

Readers at this point may wonder why the proposed Plug-and-Play ADMM can alleviate the expansive issue. In a nutshell, the reason is that we force $\rho \to \infty$ so that $\sigma \to 0$. Consequently, the weight $\boldsymbol{W} \to \boldsymbol{I}$ as $\rho \to \infty$. For the original Plug-and-Play ADMM in [25], $\boldsymbol{W} \nrightarrow \boldsymbol{I}$ because ρ is fixed.

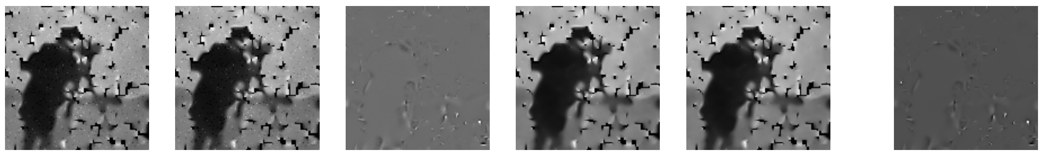

(a) $\boldsymbol{x}$ (b) $\boldsymbol{y}$ (c) $\boldsymbol{x}-\boldsymbol{y}$ (d) $\mathcal{D}_\sigma(\boldsymbol{x})$ (e) $\mathcal{D}_\sigma(\boldsymbol{y})$ (f) $\mathcal{D}_\sigma(\boldsymbol{x})-\mathcal{D}_\sigma(\boldsymbol{y})$

Fig. 3.10.: Counter example showing non-local means is expansive. $\kappa = \|\mathcal{D}_\sigma(\boldsymbol{x}) - \mathcal{D}_\sigma(\boldsymbol{y})\|^2/\|\boldsymbol{x}-\boldsymbol{y}\|^2 = 1.1775$.

3.6.2 Proof of Theorem 1

To simplify the notations we first define a triplet $\boldsymbol{\theta}^{(k)} \overset{\text{def}}{=} (\boldsymbol{x}^{(k)}, \boldsymbol{v}^{(k)}, \boldsymbol{u}^{(k)})$. Let $\boldsymbol{\Theta}$ be the domain of $\boldsymbol{\theta}^{(k)}$ for all k. On $\boldsymbol{\Theta}$ we define a distance function $D : \boldsymbol{\Theta} \times \boldsymbol{\Theta} \to \mathbb{R}$ such that

$$D(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(j)}) = \frac{1}{\sqrt{n}}\Big(\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(j)}\|_2 + \|\boldsymbol{v}^{(k)} - \boldsymbol{v}^{(j)}\|_2 + \|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(j)}\|_2\Big).$$

It then follows that $\Delta_{k+1} = D(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\theta}^{(k)})$. Since $\boldsymbol{\Theta} \subseteq \mathbb{R}^{3n}$ and $\mathbb{R}^{3n}$ is a complete metric space, as long as we can show that $\{\boldsymbol{\theta}^{(k)}\}_{k=1}^\infty$ is a Cauchy sequence in $\boldsymbol{\Theta}$ with the distance function D, then $\boldsymbol{\theta}^{(k)}$ should converge.

The Plug-and-Play ADMM involves two cases of the parameter update:

- Case 1: If $\Delta_{k+1} > \eta\Delta_k$, then $\rho_{k+1} = \gamma\rho_k$.
- Case 2: If $\Delta_{k+1} \leq \eta\Delta_k$, then $\rho_{k+1} = \rho_k$.

At iteration k, if Case 1 holds, then by Lemma 1, $\boldsymbol{\theta}^{(k+1)}$ satisfies

$$D(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\theta}^{(k)}) \leq \frac{C'}{\sqrt{\rho_k}}, \tag{3.31}$$

for some universal constant $C' > 0$ independent of k. On the other hand, if Case 2 holds, then since $\Delta_{k+1} = D(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\theta}^{(k)})$ we have

$$D(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\theta}^{(k)}) \leq \eta D(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k-1)}). \tag{3.32}$$

As $k \to \infty$, one of the following situations will happen:

(S_1) : Case 1 occurs infinitely many times but Case 2 occurs finitely many times;

(S_2) : Case 2 occurs infinitely many times but Case 1 occurs finitely many times;

(S_3) : Both Case 1 and Case 2 occur infinitely many times.

These three cases can be analyzed as follows. When (S_1) happens, there must exists a K_1 such that for $k \geq K_1$ only Case 1 will be visited. Thus,

$$D(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\theta}^{(k)}) \leq \frac{C'}{\sqrt{\rho_{K_1-1}}\sqrt{\gamma}^{k-K_1}}.$$

When (S_2) happens, there must exists a K_2 such that for $k \geq K_2$ only Case 2 will be visited. Thus, we have

$$\begin{aligned} D(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\theta}^{(k)}) &\leq \eta^{k-K_2} D(\boldsymbol{\theta}^{(K_2)}, \boldsymbol{\theta}^{(K_2-1)}) \\ &\leq \eta^{k-K_2} \frac{C'}{\rho_{K_2-1}}. \end{aligned}$$

(S_3) is a union o the (S_1) and (S_2). Therefore, as long as we can show under (S_1) and (S_2) the sequence $\{\boldsymbol{\theta}^{(k)}\}_{k=1}^{\infty}$ converges, the sequence will also converge under (S_3). To summarize, we show in Lemma 2 that regardless which of (S_1)-(S_3), for any k we have

$$D(\boldsymbol{\theta}^{k+1}, \boldsymbol{\theta}^{(k)}) \leq C'' \delta^k,$$

for some constants C'' and $0 < \delta < 1$. Therefore,

$$D(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\theta}^{(k)}) \to 0, \tag{3.33}$$

as $k \to \infty$.

To prove $\{\boldsymbol{\theta}^{(k)}\}_{k=1}^{\infty}$ is a Cauchy sequence, we need to show

$$D(\boldsymbol{\theta}^{(m)}, \boldsymbol{\theta}^{(k)}) \to 0, \tag{3.34}$$

for all integers $m > k$ and $k \to \infty$. This result holds because for any finite m and k,

$$\begin{aligned} D(\boldsymbol{\theta}^{(m)}, \boldsymbol{\theta}^{(k)}) &\le \sum_{n=k+1}^{m} C''\delta^{n} \\ &= \sum_{\ell=1}^{m-k} C''\delta^{\ell+k} \\ &= C''\delta^{k}\frac{1-\delta^{m-k+1}}{1-\delta}. \end{aligned}$$

Therefore, as $k \to \infty$, $D(\boldsymbol{\theta}^{(m)}, \boldsymbol{\theta}^{(k)}) \to 0$. Hence, $\{\boldsymbol{\theta}^{(k)}\}_{k=1}^{\infty}$ is a Cauchy sequence. Since a Cauchy sequence in $\mathbb{R}^{3n}$ always converges, there must exists $\boldsymbol{\theta}^{*} = (\boldsymbol{x}^{*}, \boldsymbol{v}^{*}, \boldsymbol{u}^{*})$ such that

$$D(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{*}) \to 0. \tag{3.35}$$

Consequently, we have $\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{*}\|_2 \to 0$, $\|\boldsymbol{v}^{(k)} - \boldsymbol{v}^{*}\|_2 \to 0$ and $\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{*}\|_2 \to 0$. This completes the proof.

Lemma 1 *At iteration k, if Case 1 holds, then*

$$D(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\theta}^{(k)}) \le \frac{C'}{\sqrt{\rho_k}}, \tag{3.36}$$

for some universal constant $C' > 0$ independent of k.

Proof Following the definition of $D(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\theta}^{(k)})$, it is sufficient to show that

$$\frac{1}{\sqrt{n}}\left\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\right\|_2 \leq \frac{C_1}{\sqrt{\rho_k}},$$
$$\frac{1}{\sqrt{n}}\left\|\boldsymbol{v}^{(k+1)} - \boldsymbol{v}^{(k)}\right\|_2 \leq \frac{C_2}{\sqrt{\rho_k}},$$
$$\frac{1}{\sqrt{n}}\left\|\boldsymbol{u}^{(k+1)} - \boldsymbol{u}^{(k)}\right\|_2 \leq \frac{C_3}{\sqrt{\rho_k}},$$

for some universal constants C_1, C_2 and C_3.

Let us consider

$$\boldsymbol{x}^{(k+1)} = \underset{\boldsymbol{x}}{\operatorname{argmin}}\; f(\boldsymbol{x}) + \frac{\rho_k}{2}\|\boldsymbol{x} - (\boldsymbol{v}^{(k)} - \boldsymbol{u}^{(k)})\|^2.$$

The first order optimality implies that

$$\boldsymbol{x} - (\boldsymbol{v}^{(k)} - \boldsymbol{u}^{(k)}) = -\frac{1}{\rho_k}\nabla f(\boldsymbol{x}).$$

Since the minimizer is $\boldsymbol{x} = \boldsymbol{x}^{(k+1)}$, substituting $\boldsymbol{x} = \boldsymbol{x}^{(k+1)}$ and using the fact that ∇f is bounded yields

$$\frac{1}{\sqrt{n}}\left\|\boldsymbol{x}^{(k+1)} - (\boldsymbol{v}^{(k)} - \boldsymbol{u}^{(k)})\right\|_2 = \frac{\|\nabla f(\boldsymbol{x})\|_2}{\rho_k\sqrt{n}} \leq \frac{L}{\rho_k}. \tag{3.37}$$

Next, let $\widetilde{\boldsymbol{v}}^{(k)} = \boldsymbol{x}^{(k+1)} + \boldsymbol{u}^{(k)}$ and $\sigma_k = \sqrt{\lambda/\rho_k}$. Define

$$\boldsymbol{v}^{(k+1)} = \mathcal{D}_{\sigma_k}(\widetilde{\boldsymbol{v}}^{(k)}).$$

Since $\mathcal{D}_{\sigma_k}$ is a bounded denoiser, we have that

$$\begin{aligned}\frac{1}{\sqrt{n}}\left\|\boldsymbol{v}^{(k+1)} - (\boldsymbol{x}^{(k+1)} + \boldsymbol{u}^{(k)})\right\|_2 &= \frac{1}{\sqrt{n}}\left\|\boldsymbol{v}^{(k+1)} - \widetilde{\boldsymbol{v}}^{(k)}\right\|_2 \\ &= \frac{1}{\sqrt{n}}\left\|\mathcal{D}_{\sigma_k}(\widetilde{\boldsymbol{v}}^{(k)}) - \widetilde{\boldsymbol{v}}^{(k)}\right\|_2 \leq \sigma_k\sqrt{C} = \frac{\sqrt{\lambda}\sqrt{C}}{\sqrt{\rho_k}}.\end{aligned} \tag{3.38}$$

We can now bound $\left\|\boldsymbol{v}^{(k+1)} - \boldsymbol{v}^{(k)}\right\|_2$ as follows.

$$\begin{aligned}\frac{1}{\sqrt{n}}\left\|\boldsymbol{v}^{(k+1)} - \boldsymbol{v}^{(k)}\right\|_2 \le \frac{1}{\sqrt{n}}\left\|\boldsymbol{v}^{(k+1)} - (\boldsymbol{x}^{(k+1)} + \boldsymbol{u}^{(k)})\right\|_2 \\ + \frac{1}{\sqrt{n}}\left\|(\boldsymbol{x}^{(k+1)} + \boldsymbol{u}^{(k)}) - \boldsymbol{v}^{(k)}\right\|_2.\end{aligned}$$

Using (3.37) and (3.38), we have

$$\begin{aligned}&\frac{1}{\sqrt{n}}\left\|\boldsymbol{v}^{(k+1)} - \boldsymbol{v}^{(k)}\right\|_2 \le \frac{\sqrt{\lambda}\sqrt{C}}{\sqrt{\rho_k}} + \frac{L}{\rho_k} \\ &= \frac{1}{\sqrt{\rho_k}}\left(\sqrt{\lambda}\sqrt{C} + \frac{L}{\sqrt{\rho_k}}\right) \le \frac{1}{\sqrt{\rho_k}}\left(\sqrt{\lambda}\sqrt{C} + \frac{L}{\sqrt{\rho_0}}\right).\end{aligned} \tag{3.39}$$

Similarly, we can show that

$$\begin{aligned}&\frac{1}{\sqrt{n}}\left\|\boldsymbol{u}^{(k+1)}\right\|_2 = \frac{1}{\sqrt{n}}\left\|\boldsymbol{u}^{(k)} + (\boldsymbol{x}^{(k+1)} - \boldsymbol{v}^{(k+1)})\right\|_2 \\ &= \frac{1}{\sqrt{n}}\left\|\boldsymbol{u}^{(k)} + \boldsymbol{x}^{(k+1)} - \mathcal{D}_{\sigma_k}(\widetilde{\boldsymbol{v}}^{(k)})\right\|_2 \\ &= \frac{1}{\sqrt{n}}\left\|\boldsymbol{u}^{(k)} + \boldsymbol{x}^{(k+1)} - (\mathcal{D}_{\sigma_k}(\widetilde{\boldsymbol{v}}^{(k)}) - \widetilde{\boldsymbol{v}}^{(k)}) - \widetilde{\boldsymbol{v}}^{(k)}\right\|_2 \\ &\overset{(a)}{=} \frac{1}{\sqrt{n}}\left\|\mathcal{D}_{\sigma_k}(\widetilde{\boldsymbol{v}}^{(k)}) - \widetilde{\boldsymbol{v}}^{(k)}\right\|_2 \le \frac{\sqrt{\lambda}\sqrt{C}}{\sqrt{\rho_k}},\end{aligned} \tag{3.40}$$

where (a) holds because $\widetilde{\boldsymbol{v}}^{(k)} = \boldsymbol{u}^{(k)} + \boldsymbol{x}^{(k+1)}$. Thus,

$$\begin{aligned}\frac{1}{\sqrt{n}}\left\|\boldsymbol{u}^{(k+1)} - \boldsymbol{u}^{(k)}\right\|_2 &\le \frac{1}{\sqrt{n}}\left(\left\|\boldsymbol{u}^{(k+1)}\right\|_2 + \left\|\boldsymbol{u}^{(k)}\right\|_2\right) \\ &\le \frac{2\sqrt{\lambda}\sqrt{C}}{\sqrt{\rho_k}}.\end{aligned}$$

Finally, since $\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + (\boldsymbol{x}^{(k+1)} - \boldsymbol{v}^{(k+1)})$, we have

$$\begin{aligned}
&\frac{1}{\sqrt{n}} \left\| \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)} \right\|_2 \\
&= \frac{1}{\sqrt{n}} \left\| \left(\boldsymbol{u}^{(k+1)} - \boldsymbol{u}^{(k)} + \boldsymbol{v}^{(k+1)} \right) - \left(\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)} + \boldsymbol{v}^{(k)} \right) \right\|_2 \\
&\le \frac{1}{\sqrt{n}} \Big(\left\| \boldsymbol{u}^{(k+1)} - \boldsymbol{u}^{(k)} \right\|_2 + \left\| \boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)} \right\|_2 \\
&\qquad + \left\| \boldsymbol{v}^{(k+1)} - \boldsymbol{v}^{(k)} \right\|_2 \Big) \\
&\le \frac{2\sqrt{\lambda}\sqrt{C}}{\sqrt{\rho_k}} + \frac{2\sqrt{\lambda}\sqrt{C}}{\sqrt{\rho_{k-1}}} + \frac{1}{\sqrt{\rho_k}} \left(\sqrt{\lambda}\sqrt{C} + \frac{L}{\sqrt{\rho_k}} \right) \\
&\le \left((3 + 2\sqrt{\gamma})\sqrt{\lambda}\sqrt{C} + \frac{L}{\sqrt{\rho_0}} \right) \frac{1}{\sqrt{\rho_k}}.
\end{aligned} \tag{3.41}$$

■

Lemma 2 *The sequence $\{\boldsymbol{\theta}^{(k)}\}_{k=1}^{\infty}$ always satisfies*

$$D(\boldsymbol{\theta}^{k+1}, \boldsymbol{\theta}^{(k)}) \le C'' \delta^k, \tag{3.42}$$

for some constants C'' and $0 < \delta < 1$.

Proof At any iteration k, it holds that

$$\begin{aligned}
D(\boldsymbol{\theta}^{k+1}, \boldsymbol{\theta}^{(k)}) &\le \max \left(\frac{C'}{\sqrt{\rho_{K_1-1}}\sqrt{\gamma}^{k-K_1}}, \eta^{k-K_2} \frac{C'}{\rho_{K_2-1}} \right) \\
&\le \max \left(C_1' \left(\frac{1}{\sqrt{\gamma}} \right)^k, C_2' \eta^k \right),
\end{aligned}$$

where $C_1' = C' \sqrt{\frac{\gamma^{K_1}}{\rho_{K_1-1}}}$ and $C_2' = \frac{C' \eta^{-K_2}}{\rho_{K_2-1}}$. Therefore, by letting

$$C'' = \max\,(C_1', C_2'), \quad \text{and} \quad \delta = \max\,(1/\sqrt{\gamma}, \eta),$$

we obtain the desired result, as $\gamma > 1$. ■

4. PARAMETER-FREE PLUG-AND-PLAY ADMM FOR IMAGE RESTORATION

4.1 Introduction

4.1.1 Plug-and-Play ADMM

With the astonishing number of applications of the alternating direction method of multiplier (ADMM, [24]), it is reasonably safe to say that ADMM is almost *the* workhorse of most, if not all, image restoration algorithms we use nowadays [18,45,46]. ADMM is a generic algorithm that solves optimization problems in the form

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad f(\boldsymbol{x}) + \lambda g(\boldsymbol{x}) \tag{4.1}$$

for some cost function f and regularization function g. In model-based image processing, f is called the data fidelity term, and g is called the prior term [44].

ADMM has many attractive features. Apart from its simple implementation and broad applicability, the variable splitting nature of the algorithm offers additional degrees of freedom in designing the steps of the algorithm. In particular, if we use ADMM to solve (4.1), the algorithm proceeds by solving a sequence of subproblems in the following *modules*:

$$\boldsymbol{x}^{(k+1)} = \underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{argmin}} \quad f(\boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{x} - \widetilde{\boldsymbol{x}}^{(k)}\|^2, \tag{4.2}$$

$$\boldsymbol{v}^{(k+1)} = \underset{\boldsymbol{v}\in\mathbb{R}^n}{\text{argmin}} \quad \lambda g(\boldsymbol{v}) + \frac{\rho}{2}\|\boldsymbol{v} - \widetilde{\boldsymbol{v}}^{(k)}\|^2, \tag{4.3}$$

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \rho(\boldsymbol{x}^{(k+1)} - \boldsymbol{v}^{(k+1)}), \tag{4.4}$$

where $\widetilde{\boldsymbol{x}}^{(k)} \overset{\text{def}}{=} \boldsymbol{v}^{(k)} - (1/\rho)\boldsymbol{u}^{(k)}$, $\widetilde{\boldsymbol{v}}^{(k)} \overset{\text{def}}{=} \boldsymbol{x}^{(k+1)} + (1/\rho)\boldsymbol{u}^{(k)}$, and $\boldsymbol{u}^{(k)}$ is called the Lagrange multiplier. In this set of equations, subproblem (5.2a) is an inversion module that minimizes f using a quadratic regularization, whereas subproblem (5.2b) is a denoising module that denoises $\widetilde{\boldsymbol{v}}^{(k)}$ using a regularization function g.

Recognizing the inversion-denoising modules of the ADMM allows us to re-design the steps. One possibility is to replace the denoising module by an off-the-shelf image denoising algorithm, i.e.,

$$\boldsymbol{v}^{(k+1)} = \mathcal{D}_{\sigma}\left(\widetilde{\boldsymbol{v}}^{(k)}\right), \tag{4.5}$$

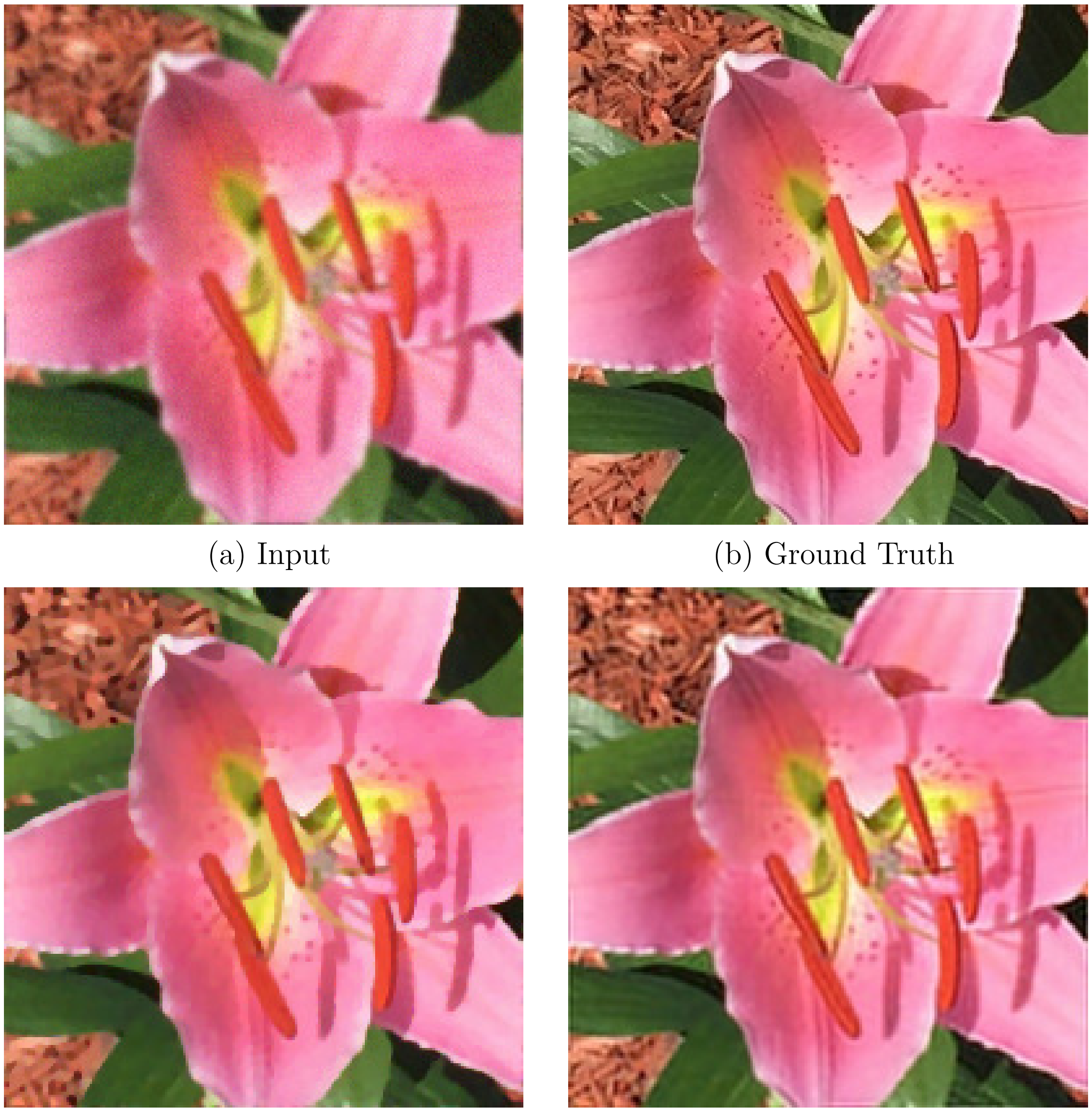
for some denoiser $\mathcal{D}_{\sigma}$ with a noise level $\sigma \overset{\text{def}}{=} \sqrt{\lambda/\rho}$. The resulting algorithm is called the Plug-and-Play ADMM, with the name attributed to Venkatakrishnan et al. [25].

Since the introduction of Plug-and-Play ADMM, many applications have been developed, e.g., X-ray computed tomography [26], image interpolation [58], super-resolution [4, 55], Poisson denoising [54], and single photon imaging [4]. In [90], the same framework was used in camera processing, with a named coined flexible ISP.

To provide readers a quick comparison between Plug-and-Play ADMM and conventional ADMM algorithms, we show in Figure 4.1 a deblurring result using Plug-and-Play ADMM with BM3D [91] as the denoiser and the same result using conventional ADMM with the total variation regularization. As can be seen in the figure, when both algorithms are tuned to their best parameter λ, Plug-and-Play demonstrates more than 1dB improvement over the total variation.

4.1.2 Problem of Plug-and-Play

While Plug-and-Play ADMM offers promising image restoration results, it has a significant problem due to the parameter ρ. ρ controls the strength of the intermediate regularization, and is typically assigned by the user. If ρ is set too large, the quadratic regularization in (5.2b) dominates and so the denoiser is weak. If ρ is set too small, the denoiser is strong but the subproblem (5.2a) becomes ill-conditioned. Therefore, finding an optimal ρ is essential to the convergence of the algorithm.



(a) Input

(b) Ground Truth

(c) Total Variation [18] 29.97dB

(d) Plug-Play [4] 31.29dB

Fig. 4.1.: Image deblurring using Plug-and-Play ADMM with BM3D denoiser compared to conventional ADMM with total variation (TV) prior. In this example, the regularization parameter is optimally tuned to $\lambda = 10^{-4}$ for Plug-Play, and $\lambda = 5 \times 10^{-3}$ for TV. The internal parameter of Plug-Play is $\rho = 1$. The blur in this example is Gaussian of size 9×9 with radius $\sigma = 1$. The noise level is $5/255$.

If f is convex and if the denoiser $\mathcal{D}_\sigma$ has a doubly stochastic gradient, Sreehari et al. [26] showed that $\mathcal{D}_\sigma$ is a non-expansive proximal operator and so ρ does not affect the final solution. As a rule-of-thumb, our experience shows that $\rho = 1$ is often a reliable choice. However, for the broader class of bounded denoisers, $\mathcal{D}_\sigma$ could be expansive and so the convergence depends on ρ. In this case, one possible solution is to define a sequence of increasing ρ's such that $\rho_{k+1} = \gamma\rho_k$ for some constant $\gamma > 1$ [4]. This will ensure that the iterates $\boldsymbol{x}^{(k)}$ and $\boldsymbol{v}^{(k)}$ can converge to a fixed point. However, we now have to choose the initial value ρ_0 and the update constant γ. Therefore, if we like to choose ρ properly, an automatic update scheme within the algorithm would be desirable.

4.1.3 Related Work and Contributions

The contribution of this work is a *parameter-free* Plug-and-Play ADMM. Here, by parameter-free we mean that the internal parameter ρ is updated as part of the ADMM algorithm, thus is free of tuning. It does not, however, mean that the regularization parameter λ is automatically tuned. Should λ be tuned automatically, a few existing methods can be considered, e.g., SURE [76] and cross validation [75], etc.

Our key idea behind the parameter-free Plug-and-Play ADMM is the Generalized Approximate Message Passing (GAMP) in the compressive sensing literature [34, 92–94]. GAMP is a generic algorithm that solves problems in the form of (4.1), typically for $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|^2$ with a random matrix $\boldsymbol{A}$ and a regularization function $g(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$. In GAMP, the internal parameters are self-updated, which is a feature this work attempts to obtain. Another piece of related work is the Denoiser-AMP by Metzler et al. [59], where a denoiser was used to replace the shrinkage step in the classical AMP. Convergence of Denoiser-AMP is known for i.i.d. Gaussian matrix $\boldsymbol{A}$ but not for general matrices.

The goal of this chapter is to derive a parameter-free Plug-and-Play ADMM from GAMP. In Section II we provide a brief introduction to the GAMP algorithm. Then in Section III, we show how the GAMP algorithm can be modified into a parameter-free Plug-and-Play ADMM. Experimental results are shown in Section IV.

4.2 Generalized Approximate Message Passing

In this section we provide a brief introduction to the generalized approximate message passing (GAMP) algorithm. For full discussions of GAMP, we refer the readers to [93]. Among all image restoration problems, we are particularly interested in the deblurring problem with f in the form

$$f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2,$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the unknown variable, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ a convolution matrix, and $\boldsymbol{y} \in \mathbb{R}^n$ is the observed signal. The regularization function g is unimportant for Plug-and-Play ADMM, because we will later replace it by a denoiser. However, since the conventional GAMP requires an explicit g, we will keep the function g in this section with the assumption that it is separable. We will remove g in Section III.

4.2.1 GAMP

The GAMP algorithm begins by considering the following problem

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{v}}{\text{minimize}} \quad & \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda g(\boldsymbol{v}) \\ \text{subject to} \quad & \boldsymbol{x} = \boldsymbol{v}. \end{aligned} \tag{4.6}$$

Clearly, at the optimal point, (4.6) has the same solution as the original unconstrained problem.

The next step GAMP does is to separately consider $f(\boldsymbol{x})$ and $g(\boldsymbol{v})$ as the *output node* and the *input node* of a bipartite graph, respectively. Then the algorithm seeks the equilibrium of the two sides, by passing intermediate variables (called *messages*) forward and backward between the nodes. Specifically, by initializing two vectors $\boldsymbol{\tau}_x^{(0)} = \mathbf{1}_{n\times 1}$ and $\boldsymbol{u}^{(0)} = \mathbf{0}_{n\times 1}$, the computation on the **output node** involves:

$$\widetilde{\boldsymbol{x}}^{(k+1)} = \boldsymbol{x}^{(k)} - \boldsymbol{\tau}_x^{(k)} \cdot \boldsymbol{u}^{(k)}, \tag{4.7}$$

$$\boldsymbol{x}^{(k+1)} = \text{prox}_{\boldsymbol{\tau}_x^{(k)} f}(\widetilde{\boldsymbol{x}}^{(k)}), \tag{4.8}$$

$$\boldsymbol{\pi}_x^{(k+1)} = \tau_x^{(k)} \cdot \frac{\partial}{\partial \widetilde{\boldsymbol{x}}} \text{prox}_{\boldsymbol{\tau}_x^{(k)} f}(\widetilde{\boldsymbol{x}}) \Big|_{\widetilde{\boldsymbol{x}} = \widetilde{\boldsymbol{x}}^{(k)}}, \tag{4.9}$$

$$\boldsymbol{\tau}_v^{(k+1)} = (\boldsymbol{\tau}_x^{(k)} - \boldsymbol{\pi}_x^{(k+1)})./(\boldsymbol{\tau}_x^{(k)})^2, \tag{4.10}$$

$$\boldsymbol{u}^{(k+1)} = (\boldsymbol{x}^{(k+1)} - \widetilde{\boldsymbol{x}}^{(k+1)})./\boldsymbol{\tau}_x^{(k)}. \tag{4.11}$$

In this set of equations, the function $\text{prox}_{\boldsymbol{\tau} f}$ is the proximal operator, defined as

$$\text{prox}_{\boldsymbol{\tau} f}(\widetilde{\boldsymbol{x}}) = \underset{\boldsymbol{x}}{\text{argmin}} \;\; \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{\tau}}^2 + \frac{1}{2}\|\boldsymbol{x} - \widetilde{\boldsymbol{x}}\|^2, \tag{4.12}$$

where the norm $\|\cdot\|_{\boldsymbol{\tau}}^2$ is a weighted norm given by $\|\boldsymbol{x}\|_{\boldsymbol{\tau}}^2 = \sum_{i=1}^n \tau_i x_i^2$. The variable $\boldsymbol{\tau}_x$ can be regarded as a vector version of the internal parameter ρ in ADMM, and $\boldsymbol{\pi}_x$ can be regarded as a measure of the variance $\boldsymbol{x}$ conditioned on $\boldsymbol{u}$.

The computation on the **input node** involves

$$\widetilde{\boldsymbol{v}}^{(k+1)} = \boldsymbol{v}^{(k)} + \boldsymbol{\tau}_v^{(k+1)} \cdot \boldsymbol{u}^{(k+1)}, \tag{4.13}$$

$$\boldsymbol{v}^{(k+1)} = \text{prox}_{\boldsymbol{\tau}_v^{(k+1)} \lambda g}(\widetilde{\boldsymbol{v}}^{(k)}), \tag{4.14}$$

$$\boldsymbol{\pi}_v^{(k+1)} = \boldsymbol{\tau}_v^{(k+1)} \cdot \frac{\partial}{\partial \widetilde{\boldsymbol{v}}} \text{prox}_{\boldsymbol{\tau}_v^{(k)} \lambda g}(\widetilde{\boldsymbol{v}}) \Big|_{\widetilde{\boldsymbol{v}} = \widetilde{\boldsymbol{v}}^{(k)}}, \tag{4.15}$$

$$\boldsymbol{\tau}_x^{(k+1)} = \boldsymbol{\pi}_v^{(k+1)}. \tag{4.16}$$

For separable $g(\boldsymbol{v}) = \sum_{i=1}^n g_i(v_i)$, the proximal operator $\text{prox}_{\boldsymbol{\tau} g}(\widetilde{\boldsymbol{v}})$ reads as

$$\text{prox}_{\tau_i \lambda g}(_i) = \underset{v}{\text{argmin}} \;\; \tau_i \lambda g_i(v) + \frac{1}{2}(v -_i)^2. \tag{4.17}$$

4.2.2 Equivalence between GAMP and ADMM

In the above input and output computation, if we ignore Equations (4.9)–(4.10) and (4.15)–(4.16), we will arrive at an ADMM algorithm with *vector-valued* parameters $\boldsymbol{\tau}_x$ and $\boldsymbol{\tau}_v$, instead of a common scalar parameter ρ. More specifically, GAMP and ADMM are related according to the following theorem [92]:

Theorem 4.2.1 *The iterates of the GAMP satisfy*

$$\begin{aligned}
\boldsymbol{v}^{(k+1)} &= \underset{\boldsymbol{v}}{argmin} \;\; L(\boldsymbol{x}^{(k)}, \boldsymbol{v}, \boldsymbol{u}^{(k)}) + \frac{1}{2}\|\boldsymbol{v} - \boldsymbol{v}^{(k)}\|^2_{\boldsymbol{\tau}_v^{(k)}}, \\
\boldsymbol{x}^{(k+1)} &= \underset{\boldsymbol{x}}{argmin} \;\; L(\boldsymbol{x}, \boldsymbol{v}^{(k+1)}, \boldsymbol{u}^{(k)}) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{v}^{(k+1)}\|^2_{\boldsymbol{\tau}_x^{(k)}}, \\
\boldsymbol{u}^{(k+1)} &= \boldsymbol{u}^{(k)} + \frac{1}{\boldsymbol{\tau}_x^{(k)}}(\boldsymbol{x}^{(k+1)} - \boldsymbol{v}^{(k+1)}),
\end{aligned}$$

where $L(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{u}) = f(\boldsymbol{x}) + \lambda g(\boldsymbol{v}) + \boldsymbol{u}^T(\boldsymbol{x} - \boldsymbol{v})$ is the Lagrangian function.

Therefore, the key difference between GAMP and ADMM is the parameters $\boldsymbol{\tau}_v^{(k)}$ and $\boldsymbol{\tau}_x^{(k)}$. This suggests that if we want to derive a Plug-and-Play ADMM from GAMP, we must first define the parameters $\boldsymbol{\tau}_v^{(k)}$ and $\boldsymbol{\tau}_x^{(k)}$.

4.3 Parameter-Free Plug-and-Play

We now derive the parameter-free Plug-and-Play using the GAMP formulation above. There are two major modifications we need for the derivation.

- The vector-valued parameters $\boldsymbol{\tau}_x$ and $\boldsymbol{\tau}_v$ should become scalars τ_x and τ_v. This would allow us to consider arbitrary denoisers which are not-necessarily separable.
- The proximal operator in (4.14) is replaced by an off-the-shelf denoiser as defined in (4.5) so that it fits the Plug-and-Play framework.

With these two modifications we can consider the output and the input nodes. We also note that among the equations (4.7)-(4.16), the biggest challenges are the proximal operators. The following two subsections will address these operators.

4.3.1 Output Node

For a convolution matrix $\boldsymbol{A}$, the proximal operator can be shown as

$$\begin{aligned}\operatorname{prox}_{\tau_x f}(\widetilde{\boldsymbol{x}}) &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \ \ \frac{\tau_x}{2}\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|^2 + \frac{1}{2}\|\boldsymbol{x}-\widetilde{\boldsymbol{x}}\|^2 \\ &= \left(\tau_x \boldsymbol{A}^T\boldsymbol{A}+\boldsymbol{I}\right)^{-1}\left(\tau_x\boldsymbol{A}^T\boldsymbol{y}+\widetilde{\boldsymbol{x}}\right). \end{aligned} \tag{4.18}$$

Taking derivative with respect to $\widetilde{\boldsymbol{x}}$ yields

$$\frac{\partial}{\partial\widetilde{\boldsymbol{x}}}\operatorname{prox}_{\tau_x f}(\widetilde{\boldsymbol{x}}) = \left(\tau_x\boldsymbol{A}^T\boldsymbol{A}+\boldsymbol{I}\right)^{-1}\boldsymbol{1} \tag{4.19}$$

Note that this is a vector of gradients. Since we are looking for a scalar, one option is to consider the divergence. This yields

$$\begin{aligned}\operatorname{div}\Big\{\operatorname{prox}_{\tau_x f}(\widetilde{\boldsymbol{x}})\Big\} &= \frac{1}{n}\boldsymbol{1}^T\left(\tau_x\boldsymbol{A}^T\boldsymbol{A}+\boldsymbol{I}\right)^{-1}\boldsymbol{1} \\ &\overset{(a)}{=} \frac{1}{n}\boldsymbol{1}^T\boldsymbol{F}^T\left(\tau_x|\boldsymbol{\Lambda}|^2+\boldsymbol{I}\right)^{-1}\boldsymbol{F}\boldsymbol{1} \\ &\overset{(b)}{=} (\tau_x|\lambda_{1,1}|^2+1)^{-1} = (\tau_x+1)^{-1}, \end{aligned} \tag{4.20}$$

where in (a) we used the fact that a convolution matrix $\boldsymbol{A}$ is diagonalizable by the Fourier transform matrix $\boldsymbol{F}$ to yield $\boldsymbol{A}=\boldsymbol{F}^T\boldsymbol{\Lambda}\boldsymbol{F}$, and in (b) we observe that $\boldsymbol{F}\boldsymbol{1} = \sqrt{n}[1,0,\ldots,0]^T$. The scalar $\lambda_{1,1}$ is the first entry of the eigenvalue matrix $\boldsymbol{\Lambda}$, which is 1 for convolutional matrices. Substituting this result into (4.9), we have

$$\pi_x^{(k+1)} = \tau_x^{(k)}/(\tau_x^{(k)}+1). \tag{4.21}$$

4.3.2 Input Node

On the input node, we have to replace the proximal operator by a denoiser $\mathcal{D}_\sigma$. Here, the noise level of the denoiser, σ, should be defined as $\sigma = \sqrt{\tau_x \lambda}$. This explains (4.14).

For the derivative in (4.15), we note that since τ_v is now a scalar, we have to replace the derivative by its divergence (which is the sum of gradients). This gives

$$\pi_v^{(k+1)} = \tau_v^{(k+1)} \text{div} \mathcal{D}_\sigma(\widetilde{\boldsymbol{v}}^{(k)}). \tag{4.22}$$

Calculating the divergence can be performed numerically using a Monte Carlo scheme. More specifically, the divergence of the denoiser at $\widetilde{\boldsymbol{v}}$ can be approximated by

$$\begin{aligned} \text{div}\mathcal{D}_\sigma(\widetilde{\boldsymbol{v}}) &= \lim_{\epsilon \to 0} \mathbb{E}_{\boldsymbol{b}} \left\{ \boldsymbol{b}^T \left(\frac{\mathcal{D}_\sigma(\widetilde{\boldsymbol{v}} + \epsilon \boldsymbol{b}) - \mathcal{D}_\sigma(\widetilde{\boldsymbol{v}})}{\epsilon} \right) \right\} \\ &\approx \boldsymbol{b}^T \left(\frac{\mathcal{D}_\sigma(\widetilde{\boldsymbol{v}} + \epsilon \boldsymbol{b}) - \mathcal{D}_\sigma(\widetilde{\boldsymbol{v}})}{n\epsilon} \right), \end{aligned} \tag{4.23}$$

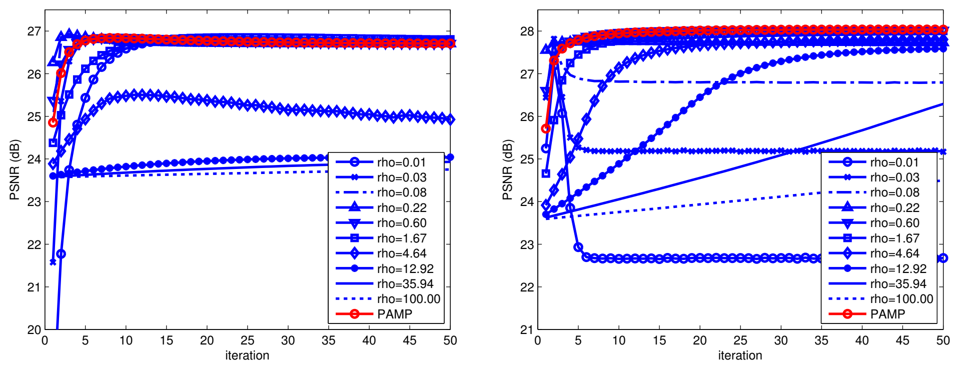
where $\boldsymbol{b} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is a random vector, and $\epsilon \ll 1$ is a small constant (typically $\epsilon = 10^{-3}$). The approximation of the expectation generally holds for large n due to concentration of measure [76].

Numerically, computing (4.23) only requires evaluating the denoiser twice: once for $\mathcal{D}_\sigma(\widetilde{\boldsymbol{v}})$, and the other time for $\mathcal{D}_\sigma(\widetilde{\boldsymbol{v}} + \epsilon \boldsymbol{b})$.

4.3.3 Final Algorithm

The final algorithm can be derived by substituting (4.18) into (4.8), (4.21) into (4.9) for the output nodes, and (4.5) into (4.14), (4.22) into (4.15) for the input nodes. Moreover, we can simplify steps by defining

$$\rho_x = 1/\tau_x, \qquad \rho_v = 1/\tau_v. \tag{4.24}$$



(a) Total Variation Denoiser (b) BM3D Denoiser

Fig. 4.2.: PSNR of PAMP compared to Plug-and-Play ADMM using a fixed ρ. (a) using total variation denoising as the denoiser; (b) using BM3D as the denoiser. In this figure, all PSNR values are averaged over 10 testing images.

Then we can show that the GAMP algorithm can be simplified to Algorithm 3. We call the resulting algorithm Plug-and-Play generalized approximate message passing (PAMP).

As shown in Algorithm 3, the difference between PAMP and Plug-and-Play ADMM is the parameters ρ_x and ρ_v. In Plug-and-Play, the parameters share the same value ρ and is fixed throughout the iterations. In PAMP, ρ_x and ρ_v are automatically updated as part of the algorithm. Therefore, PAMP is a *parameter-free* algorithm.

4.4 Experimental Results

In this section we present experimental results to evaluate the performance of the proposed PAMP algorithm. We focus on the image deblurring problem, although the method can easily be extended to image interpolation and image super-resolution.

We test the algorithm using 10 standard gray-scale images. Each image is blurred by a spatially invariant Gaussian blur kernel of size 9×9 and standard deviation 1. Additive i.i.d. Gaussian noise of zero mean and standard deviation $\sigma = 5/255$ is added to the blurred image. Two denoisers $\mathcal{D}_\sigma$ are considered in this experiment: total variation denoising [95], and BM3D [91]. For total variation, we use the MATLAB

Algorithm 3 Proposed Algorithm: PAMP

```
1: Initialize $\boldsymbol{u}^{(0)} = \boldsymbol{x}^{(0)} = \boldsymbol{0}$, $\rho_v^{(0)} = 1$.
2: for $k = 0, 1, \ldots, k_{\max}$ do
3:     % ($\boldsymbol{v}$-subproblem)
4:     $\widetilde{\boldsymbol{v}} = \boldsymbol{x}^{(k)} + (1/\rho_v^{(k)})\boldsymbol{u}^t$
5:     $\boldsymbol{v}^{(k+1)} = \mathcal{D}_\sigma(\widetilde{\boldsymbol{v}})$, where $\sigma = \sqrt{\lambda/\rho_v^{(k)}}$
6:     $\rho_x^{(k+1)} = \rho_v^{(k)}/\text{div}\mathcal{D}_\sigma(\widetilde{\boldsymbol{v}})$
7:
8:     % ($\boldsymbol{x}$-subproblem)
9:     $\widetilde{\boldsymbol{x}} = \boldsymbol{v}^{(k+1)} - (1/\rho_x^{(k+1)})\boldsymbol{u}^{(k)}$
10:    $\boldsymbol{x}^{(k+1)} = (\boldsymbol{A}^T\boldsymbol{A} + \rho_x^{(k+1)}\boldsymbol{I})^{-1}(\boldsymbol{A}^T\boldsymbol{y} + \rho_x^{(k+1)}\widetilde{\boldsymbol{x}})$
11:    $\rho_v^{(k+1)} = \rho_x^{(k+1)}/(\rho_x^{(k+1)} + 1)$
12:
13:    % (Multiplier update)
14:    $\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \rho_x^{(k+1)}(\boldsymbol{x}^{(k+1)} - \boldsymbol{v}^{(k+1)})$
15: end for
```

implementation in [18], whereas for BM3D, we use the code available on author's website. When total variation is used, we set the regularization parameter $\lambda = 10^{-2}$. When BM3D is used, we set $\lambda = 10^{-3}$. These values are selected as they produce the best overall result for the entire dataset.

Since the objective of PAMP is to automatically select ρ, we compare PAMP with Plug-and-Play ADMM which uses a fixed ρ. We select 10 values of ρ from 10^{-2} to 10^2 in the logarithmic scale. For each ρ, we run Plug-and-Play ADMM for 50 iterations and record the PSNR values. For PAMP, we initialize the algorithm with $\rho_v^{(0)} = 1$ and let the algorithm to update ρ_x and ρ_v internally.

The results are shown in Figure 4.2. In this figure, the PSNR values are averaged over the 10 testing images. As can be observed, the parameter ρ has important influence to the Plug-and-Play ADMM algorithm where large ρ tends to converge slower and approaches a solution with low PSNR. If ρ is too small, e.g., $\rho = 0.01$ in the BM3D case, the PSNR actually drops rapid after the first iteration. As for PAMP, we observe that in both denoisers the solution picks a rapid convergence path with almost the highest PSNR.

Additional results, including other types of blur and other noise levels, can be found at https://engineering.purdue.edu/ChanGroup/

4.5 Discussion and Conclusion

Why it works? Line 6 and Line 11 of Algorithm 3 reveals that there are two opposite forces in updating ρ_v and ρ_x:

$$\rho_x \leftarrow \rho_v / \text{div}\mathcal{D}_\sigma(\widetilde{\boldsymbol{v}}), \tag{4.25}$$

$$\rho_v \leftarrow \rho_x/(\rho_x + 1) = \rho_x \text{ div} \left\{ \text{prox}_{(1/\rho_x)f}(\widetilde{\boldsymbol{x}}) \right\}. \tag{4.26}$$

Divergence of a function is an indicator of the sensitivity with respect to the input. When $\text{div}\mathcal{D}_\sigma$ is large, the denoiser $\mathcal{D}_\sigma$ behaves sensitively at $\widetilde{\boldsymbol{v}}$ and so the denoised output is less reliable. Thus, PAMP makes ρ_x small to attenuate the influence of the denoiser when solving the inversion. Now, since ρ_x becomes small, the inversion is weak and so in the next iteration a strong denoiser is needed. This is achieved by decreasing ρ_v in (4.26). These two opposite forces form a trajectory of the pair (ρ_x, ρ_v). As $k \to \infty$, one can show that (ρ_x, ρ_v) approaches a steady state where the two divergence terms coincides: $\text{div}\mathcal{D}_\sigma(\widetilde{\boldsymbol{v}}) = \text{div} \left\{ \text{prox}_{(1/\rho_x)f}(\widetilde{\boldsymbol{x}}) \right\}$.

Convergence? Convergence of GAMP is an open problem. The best result we know so far is that with appropriately designed damping strategies, GAMP converges for strictly convex functions f and g [96]. However, when damping is not used, there are examples where GAMP diverges [97]. Moving from an explicit g to an implicit denoiser $\mathcal{D}_\sigma$ will cause additional challenges yet to be studied.

Conclusion. Plug-and-Play generalized approximate message passing (PAMP) is a new algorithm that automatically updates the internal parameters of a conventional Plug-and-Play ADMM. The update rules are based on the measure of the divergence of the subproblems, and are derived from the generalized approximate message passing (GAMP). At the current stage, numerical results show that PAMP is a promising algorithm as it generates solutions through a rapid convergence path. Intuitive arguments of the algorithm are made. Future work should focus on the convergence analysis.

Table 4.1.: Image Super Resolution Results.

	Images											
	Barbara	Boat	Cameraman	Couple	Fingerprint	Hill	House	Lena	Man	Peppers	Dataset	Avg STD
Size	512^2	512^2	256^2	512^2	512^2	512^2	256^2	512^2	512^2	256^2	Avg	per image
Factor: ×2; Anti-aliasing Filter: Gaussian 9×9, $\sigma = 1$; Noise: 0.05; λ:10^{-3}												
3 PAMP	23.93	27.20	**25.10**	27.04	**25.83**	27.85	29.36	29.53	27.52	**26.51**	26.99	–
VanillaPP	**24.00**	**27.26**	24.95	**27.06**	25.74	**28.04**	**30.04**	**30.08**	**27.62**	26.23	**27.10**	–
Factor: ×2; Anti-aliasing Filter: Gaussian 9×9, $\sigma = 1$: 0; λ:10^{-6}												
3 PAMP	**25.80**	**30.99**	**27.29**	**30.65**	**32.95**	**32.08**	**33.72**	**35.53**	**32.18**	**28.30**	**30.95**	–
VanillaPP	25.35	30.32	26.49	29.97	31.23	31.48	32.90	34.34	31.30	27.73	30.11	–
Factor: ×4; Anti-aliasing Filter: Gaussian 9×9, $\sigma = 1$; Noise: 0.05; λ:10^{-2}												
3 PAMP	**23.02**	**24.53**	**22.31**	**24.13**	**22.26**	**25.87**	**26.94**	**27.32**	**25.26**	**23.06**	**24.47**	–
VanillaPP	22.45	23.41	21.41	23.28	19.43	25.00	25.27	26.00	24.14	21.81	23.22	–
Factor: ×4; Anti-aliasing Filter: Gaussian 9×9, $\sigma = 1$; Noise: 0; λ:10^{-3}												
PAMP	**23.78**	**25.82**	**23.15**	**25.52**	**23.53**	**27.41**	**28.84**	**29.88**	**27.19**	**24.59**	**25.97**	–
VanillaPP	23.11	24.55	22.07	24.45	20.96	26.14	26.53	27.59	25.63	23.03	24.41	–

Table 4.2.: Image Inpainting Results.

	Images											
	Barbara	Boat	Cameraman	Couple	Fingerprint	Hill	House	Lena	Man	Peppers	Dataset	Avg STD
Size	512^2	512^2	256^2	512^2	512^2	512^2	256^2	512^2	512^2	256^2	Avg	per image
Missing Pixels: 80%; Noise: 0; λ:10^{-2}												
3 PAMP	**26.81**	**26.32**	23.49	**26.36**	**24.19**	**27.47**	**30.17**	**29.54**	**26.82**	**25.47**	**26.66**	–
VanillaPP	23.90	25.76	**24.13**	25.66	22.79	26.90	29.62	28.59	26.09	24.64	25.81	–
Missing Pixels: 80%; Noise: 0.05; λ:10^{-2}												
3 PAMP	**26**	**25.65**	23.00	**25.69**	**23.77**	**26.83**	**29.55**	**28.66**	**26.23**	**24.74**	**26.01**	–
VanillaPP	23.66	25.52	**23.94**	25.32	22.62	26.57	29.31	28.31	25.81	24.26	25.53	–
Missing Pixels: 60%; Noise: 0; λ:10^{-3}												
3 PAMP	**32.15**	30.45	26.50	30.67	**30.81**	**31.13**	**34.71**	33.13	30.38	**30.27**	31.02	–
VanillaPP	31.93	**30.89**	**27.47**	**31.09**	29.89	31.09	34.62	**33.90**	**30.69**	29.60	**31.12**	–
Missing Pixels: 60%; Noise: 0.1; λ:10^{-2}												
PAMP	**27.65**	**26.88**	**25.67**	**26.77**	**24.93**	**27.44**	30.47	**29.42**	**27.04**	**26.94**	**27.32**	–
VanillaPP	26.91	26.68	25.46	26.61	24.37	27.27	**30.55**	29.31	26.82	26.60	27.06	–

Table 4.3.: Image Deblurring Results.

	Images											
	Barbara	Boat	Cameraman	Couple	Fingerprint	Hill	House	Lena	Man	Peppers	Dataset	Avg STD
Size	512^2	512^2	256^2	512^2	512^2	512^2	256^2	512^2	512^2	256^2	Avg	per image
Anti-aliasing Filter: Gaussian 9×9, $\sigma = 1$; Noise: 0.05; λ:10^{-3}												
3 PAMP	**24.98**	**29.30**	**27.06**	**29.17**	28.23	**29.66**	**32.09**	**32.19**	**29.52**	**27.68**	**28.99**	–
VanillaPP	24.69	29.25	26.90	29.12	**28.24**	29.60	32.02	**32.19**	29.49	27.46	28.90	–
Anti-aliasing Filter: Gaussian 9×9, $\sigma = 1$; Noise: 0.1; λ:10^{-2}												
3 PAMP	**23.99**	**27.20**	**25.15**	**27.02**	**25.60**	**28.02**	**30.17**	**29.98**	**27.62**	**26.36**	**27.11**	–
VanillaPP	23.93	27.18	25.11	27.01	25.54	28.01	30.20	**29.98**	27.61	26.27	27.08	–
Anti-aliasing Filter: Gaussian 9×9, $\sigma = 2$; Noise: 0.1; λ:10^{-2}												
3 PAMP	**23.40**	**25.33**	**23.11**	**25.08**	**23.18**	**26.65**	**28.35**	**28.15**	**26.22**	**23.85**	**25.33**	–
VanillaPP	23.35	25.29	23.06	25.05	23.08	26.61	28.28	28.11	26.21	23.78	25.28	–
Motion Filter: Gaussian $[20, 45]$; Noise: 0.05; λ:10^{-3}												
PAMP	**23.03**	**24.04**	**21.77**	**23.71**	**19.39**	**25.43**	**24.79**	**26.97**	**25.05**	**22.13**	**23.63**	–
VanillaPP	23.00	23.98	21.71	23.65	19.29	25.36	24.76	26.91	25.00	22.03	223.57	–

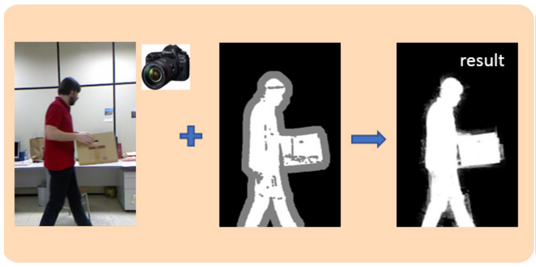
5. FOREGROUND EXTRACTION FOR VIRTUAL REALITY FILMING USING MULTI-AGENT CONSENSUS EQUILIBRIUM

5.1 Introduction

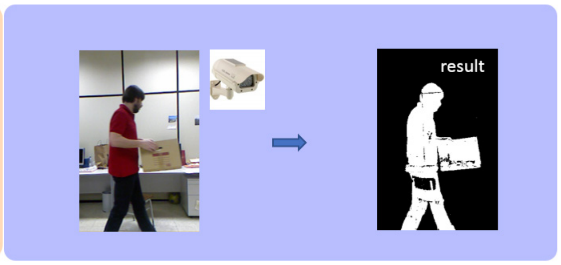
5.1.1 Motivation and Scope

The proliferation of virtual reality displays and rendering technology over the past decade is rapidly changing the landscape of cinematography [98–101]. From the traditional motion pictures to the recent 3D animation, it is safe to argue that the next wave in the film-making industry is immersive experience, e.g., head-mounted virtual reality displays. In order to offer sufficient content to these displays, data has to be acquired in special ways, e.g., using 360-degree volumetric imagers [102]. Typically, such videos are high-definition 8192×4320, full frame rate at 60 fps, and are captured using as many as 100 cameras simultaneously. This is an enormous amount of data: A five-minute video sequence using the above configuration is already equal to more than one million images. Efficient image processing of these images is therefore critical.

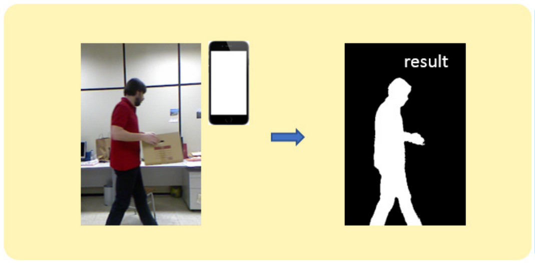
The goal of this work is to extract foreground masks from a video so that virtual background can be substituted to create a virtual scene. Foreground extraction is one of the most labor-intensive tasks in virtual reality filming. The demand on the quality and the number of images makes the problem significantly challenging than conventional video segmentations. In virtual reality filming industry, the standard approach is to recruit professional artists to manually label foreground objects for every frame. However, even with the help of industry-grade production softwares, e.g., NUKE, producing high quality masks in large volume is non-trivial. The problem



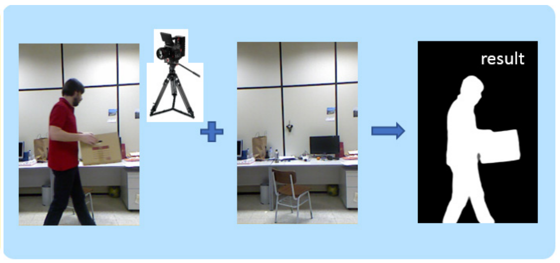
(a) Alpha matting



(b) Background subtraction



(c) Video segmentation

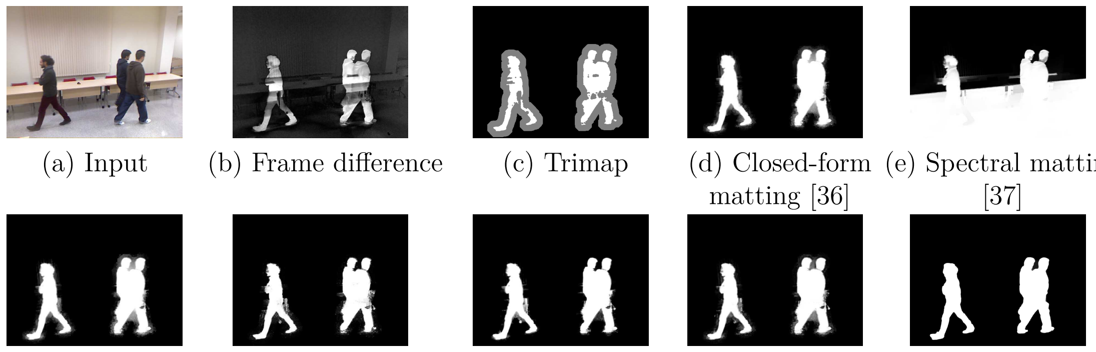


(d) Ours

Method	Alpha Matting	Bkgnd Subtraction	Video Segmentation	Ours
Goal	foreground estimation	object detection	saliency detection	foreground estimation
Input	image+trimap	video	video	image+plate
Accuracy	high	low (binary)	medium	high
Automatic	semi	full	full	full
Supervised	semi	no	no	semi

Fig. 5.1.: Objective and Assumptions. (a) Alpha matting (e.g., [103]) uses a high-quality trimap to refine the uncertain regions of the boundaries. (b) Background Subtraction (e.g., [104]) aims at locating objects in a video. The results are usually low-quality. (c) Unsupervised Video Segmentation (e.g., [105]) aims at identifying the salient objects. Unsalient foreground objects, e.g., the box, will be missed. (d) This work assumes a plate image. The goal is to extract high quality foreground objects.

can be alleviated by means of chroma-keying [106], i.e., using green screens. However, green screens limit the filming environment which is not always allowed. In this chapter, we present a *fully automatic* method to extract *high-quality* foreground masks from videos. The assumption made here is that a static background image, called the plate image, is available. Discussions about the feasibility and challenges of the plate will be presented shortly.



(a) Input (b) Frame difference (c) Trimap (d) Closed-form matting [36] (e) Spectral matting [37]

(f) Learning-based matting [107] (g) K-nearest neighbors [108] (h) Comprehensive sampling [109] (i) DCNN [103] (j) Ours (without trimap)

Fig. 5.2.: Comparison with existing alpha-matting algorithms on real images with a frame-difference based trimap. (a) Input image. (b) Frame difference. (c) Trimap generated by morphographic operation (dilation / erosion) of the binary mask. (d) - (i) Alpha matting algorithms available on `alphamatting.com`. (j) Proposed method. This sequence is from the dataset of [110].

5.1.2 Related Work

The problem studied in this chapter has a subtle but important difference with the existing foreground / background segmentation methods in the literature, as illustrated in Figure 5.1. At the high level, these differences can be summarized in three forms: (1) Quality of the output masks. (2) Requirement of the input (3) Automation and supervision. We briefly describe the comparison with existing methods.

- **Alpha Matting** a [36,38,39,103,111–116]. Alpha matting is a semi-supervised method. Given a user labeled "trimap", the algorithm uses a linear color model to predict the likelihood of a pixel being foreground or background as shown in Figure 5.1(a). The biggest limitation is that the trimaps have to be *error-free.* As soon there is a false alarm of miss in the trimap, the resulting mask will be severely distorted. Figure 5.4 shows the performance of several state-of-the-art alpha matting algorithms applied to a multi-object-moving scene. In video setting, methods such as [117–119] suffer similar issues of error-prone trimaps due to temporal propagation. Two-stage methods such as [120] requires initial segmentation [121] to provide the trimap and suffer the same problem. Other methods [122, 123] require additional sensor data, e.g., depth, which is not always available.

- **Background Subtraction** [104, 124–127]. Background subtraction is a fully automatic and unsupervised method. Existing background subtraction methods range from the simple frame difference method to the more sophisticated mixture models [124]. Most background subtraction methods are used to track objects instead of extracting the alpha mattes. They are fully-automated and are real time, but the foreground masks generated are usually of low quality.

- **Unsupervised Video Segmentation** [105, 128–137]. Unsupervised video segmentation, as it is named, is unsupervised and fully automatic. The idea is to use different saliency cues to identify objects in the video, and then segments them out. Early approaches such as [132] and [128, 130, 138] use motion boundary, motion-trajectory and objectness as cues. More recent works such as [105, 136, 137] use visual attention, optical flows and leverages the strength of deep neural networks. See [139] for additional discussions of the state-of-the-art. With post-processing methods such as conditional random field [129], the output masks can achieve reasonably good quality. However, saliency detection fails to extract foreground objects that are not salient. Figure 5.1(c) shows an example where the foreground box is missed.

The biggest difference of the present work is the availability of a *plate* image. The plate is a unique character of filming. In filming, it is almost always possible to capture the plate image before the subjects enter the scene. This is very different from conventional video recording where we do not have the same level of controllability, e.g., using cell phones to record a street event. Therefore, although the assumption does not hold in general, the specific application in filming justifies the need.

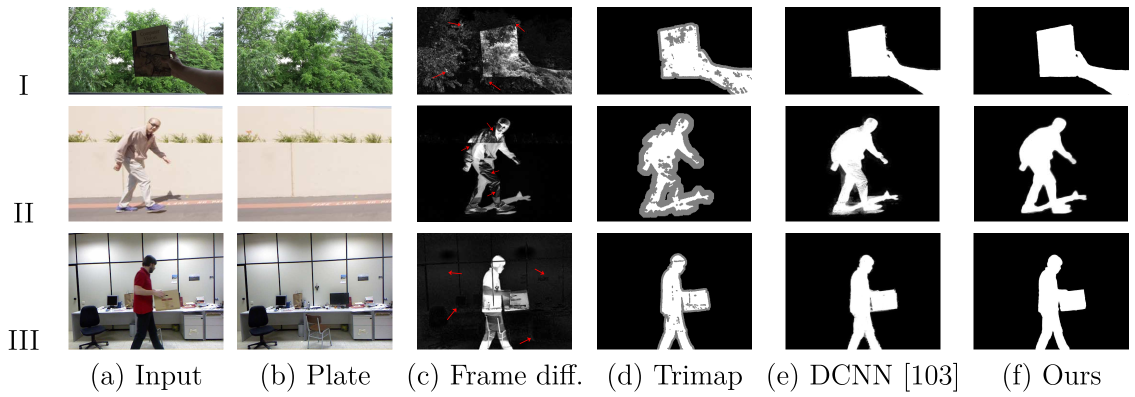
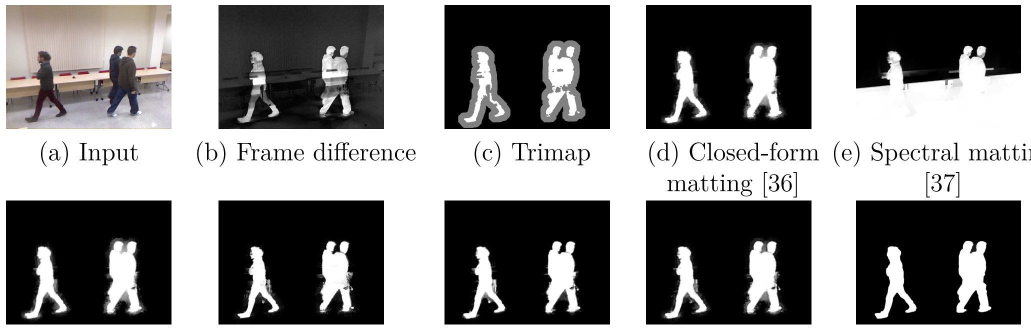


Fig. 5.3.: Three common issues of automatic foreground extraction. Case I: Vibrating background. Notice the small vibration of the leaves in the background. Case II. Similar foreground / background color. Notice the missing parts of the body of the man, and the excessive large uncertainty region of the trimap. Case III. Auto-exposure. Notice the false alarm in the background of the frame difference map. We compare our method with DCNN [103], a semi-supervised alpha matting method using the generated trimaps. The video data of Case III is from [110].

5.1.3 Challenges of the Plate Images

Readers at this point may argue that the plate assumption is strong: If the plate is available, it seems that a standard frame difference with morphographic operations (e.g., erosion / dilation) would be enough to provide a trimap, and thus a sufficiently powerful alpha matting algorithm would work. However, except for synthetic videos, plate images are never perfect. Below are three typical problems:

[leftmargin=*]**Background vibration**. While we assume that the plate does not contain large moving objects, small vibration of the background generally exists. Figure 5.3 Case I shows an example where the background tree vibrates. **Color similarity**. When foreground color is very similar to the background color, the trimap generated will have false alarms and misses. Figure 5.3 Case II shows an example where the cloth of the man has a similar color to the wall. **Auto-exposure**. If auto-exposure is used, the background intensity will change over time. Figure 5.3 Case III shows an example where the background cabinet becomes dimmer when the man leaves the room.



(a) Input (b) Frame difference (c) Trimap (d) Closed-form matting [36] (e) Spectral matting [37]

(f) Learning-based matting [107] (g) K-nearest neighbors [108] (h) Comprehensive sampling [109] (i) DCNN [103] (j) Ours (without trimap)

Fig. 5.4.: Comparison with existing alpha-matting algorithms on real images with a frame-difference based trimap. (a) Input image. (b) Frame difference. (c) Trimap generated by morphographic operation (dilation / erosion) of the binary mask. (d) - (i) Alpha matting algorithms available on `alphamatting.com`. (j) Proposed method. This sequence is from the dataset of [110].

As shown in the examples, error in frame difference can be easily translated to false alarms and misses in the trimap. While we can increase the uncertainty region of the trimap to rely more on the color constancy model of the alpha matting, in general the alpha matting performs worse when the uncertainty region grows. We have also tested more advanced background estimation algorithms, e.g., [104] in OpenCV. However, the results are similar or sometimes even worse.

5.1.4 Contributions

This work contributes to the literature in three ways.

- We present the first fully automatic foreground extraction method where a plate image is present. The problem is unique for filming and the solution is new. Under five different metrics and for 14 video sequences, our method shows a consistently better performance compared to state-of-the-art unsupervised video segmentation, background subtraction, and alpha matting algorithms.
- Our solution leverages multi-agent consensus equilibrium (MACE) [35]. MACE is a principled and provably convergent framework for integrating multiple

sources of experts. MACE has been proven an effective method for image restoration, e.g., deblurring and denoising. This work is the first demonstration of MACE for non image-restoration tasks.

- We collect and release a dataset which contains raw input video sequences, ground truth masks, and plate images. The dataset is the first of the kind in the literature.

In order to explain all the essential concepts, we organize the chapter in a way that the general framework and the specific components are addressed in two different sections. In Section 2 we describe the MACE framework. We will discuss its derivation, the intuition, and the algorithm. Then in Section 3 we go deep into the details of each component of the MACE framework that are specific to foreground extraction. Experimental results are presented in Section 4.

5.2 Multi-Agent Consensus Equilibrium

Our proposed method is based on the Multi-Agent Consensus Equilibrium (MACE), recently developed by Buzzard et al. [35]. In this section, we describe the key components of MACE and briefly discuss why it works.

5.2.1 ADMM

The starting point of MACE is the alternating direction method of multipliers (ADMM) algorithm [24]. The ADMM algorithm aims at solving a constrained minimization:

$$\underset{\boldsymbol{x}_1,\boldsymbol{x}_2}{\text{minimize}} \; f_1(\boldsymbol{x}_1) + f_2(\boldsymbol{x}_2), \quad \text{subject to} \quad \boldsymbol{x}_1 = \boldsymbol{x}_2, \tag{5.1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^n$, and $f_i : \mathbb{R}^n \to \mathbb{R}$ are mappings, typically a forward model describing the image formation process and a prior distribution of the latent image. ADMM solves the problem by solving a sequence of subproblems as follows:

$$\boldsymbol{x}_1^{(k+1)} = \underset{\boldsymbol{v}\in\mathbb{R}^n}{\text{argmin}} \quad f_1(\boldsymbol{v}) + \frac{\rho}{2}\|\boldsymbol{v} - (\boldsymbol{x}_2^{(k)} - \boldsymbol{u}^{(k)})\|^2, \tag{5.2a}$$

$$\boldsymbol{x}_2^{(k+1)} = \underset{\boldsymbol{v}\in\mathbb{R}^n}{\text{argmin}} \quad f_2(\boldsymbol{v}) + \frac{\rho}{2}\|\boldsymbol{v} - (\boldsymbol{x}_1^{(k+1)} + \boldsymbol{u}^{(k)})\|^2, \tag{5.2b}$$

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + (\boldsymbol{x}_1^{(k+1)} - \boldsymbol{x}_2^{(k+1)}). \tag{5.2c}$$

In the last equation (5.2c), the vector $\boldsymbol{u}^{(k)} \in \mathbb{R}^n$ is the Lagrange multiplier associated with the constraint. Under mild conditions, e.g., when f_1 and f_2 are convex, close, and proper, global convergence of the algorithm can be proved [26]. Recent studies show that ADMM converges even for some non-convex functions [4].

When f_1 and f_2 are convex, the minimizations in (5.2a) and (5.2b) are known as the proximal maps of f_1 and f_2, respectively [140]. If we define the proximal maps as

$$F_i(\boldsymbol{z}) = \underset{\boldsymbol{v}\in\mathbb{R}^n}{\text{argmin}} \quad f_i(\boldsymbol{v}) + \frac{\rho}{2}\|\boldsymbol{v} - \boldsymbol{z}\|^2, \tag{5.3}$$

then it is not difficult to see that at the optimal point, (5.2a) and (5.2b) become

$$F_1(\boldsymbol{x}^* - \boldsymbol{u}^*) = \boldsymbol{x}^*, \tag{5.4a}$$

$$F_2(\boldsymbol{x}^* + \boldsymbol{u}^*) = \boldsymbol{x}^*, \tag{5.4b}$$

where $(\boldsymbol{x}^*, \boldsymbol{u}^*)$ are the solutions to the original constrained optimization in (5.1). (5.4a) and (5.4b) shows that the solution $(\boldsymbol{x}^*, \boldsymbol{u}^*)$ can now be considered as a fixed point of the system of equations.

Rewriting (5.2a)-(5.2c) in terms of (5.4a) and (5.4b) allows us to consider agents F_i that are not necessarily proximal maps, i.e., f_i is not convex or F_i may not be expressible as optimizations. One example is to use an off-the-shelf image denoiser for F_i, e.g., BM3D, non-local means, or neural network denoisers. Such algorithm is known as the Plug-and-Play ADMM [4, 25, 26] (and variations thereafter [35, 141]).

5.2.2 MACE and Intuition

MACE generalizes the above ADMM formulation. Instead of minimizing a sum of two functions, MACE minimizes a sum of N functions $f_1, \ldots, f_N$:

$$\underset{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N}{\text{minimize}} \quad \sum_{i=1}^{N} f_i(\boldsymbol{x}_i), \quad \boldsymbol{x}_1 = \ldots = \boldsymbol{x}_N. \tag{5.5}$$

In this case, the equations in (5.4a)-(5.4b) are generalized to

$$\begin{aligned} F_i(\boldsymbol{x}^* + \boldsymbol{u}_i^*) &= \boldsymbol{x}^*, \quad \text{for } i = 1, \ldots, N \\ \textstyle\sum_{i=1}^{N} \boldsymbol{u}_i^* &= 0. \end{aligned} \tag{5.6}$$

What does (5.6) buy us? Intuitively, (5.6) suggests that in a system containing N agents, each agent will create a tension $\boldsymbol{u}_i^* \in \mathbb{R}^n$. For example, if F_1 is an inversion step whereas F_2 is a denoising step, then F_1 will not agree with F_2 because F_1 tends to recover details but F_2 tends to smooth out details. The agents $F_1, \ldots, F_N$ will reach an equilibrium state where the sum of the tension is zero. This explains the name “consensus equilibrium”, as the the algorithm is seeking a consensus among all the agents.

How does the equilibrium solution look like? The following theorem, shown in [35], provides a way to connect the equilibrium condition to a fixed point of an iterative algorithm.

Theorem 5.2.1 (MACE solution [35]) *Let* $\underline{\boldsymbol{u}}^* \overset{def}{=} [\boldsymbol{u}_1^*; \dots; \boldsymbol{u}_N^*]$. *The consensus equilibrium* $(\boldsymbol{x}^*, \underline{\boldsymbol{u}}^*)$ *is a solution to the MACE equation* (5.6) *if and only if the points* $\boldsymbol{v}_i^* \overset{def}{=} \boldsymbol{x}^* + \boldsymbol{u}_i^*$ *satisfy*

$$\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{v}_i^* = \boldsymbol{x}^* \tag{5.7}$$

$$(2\mathcal{G} - \mathcal{I})(2\mathcal{F} - \mathcal{I})\underline{\boldsymbol{v}}^* = \underline{\boldsymbol{v}}^*, \tag{5.8}$$

where $\underline{\boldsymbol{v}}^* \overset{def}{=} [\boldsymbol{v}_1^*; \dots; \boldsymbol{v}_N^*] \in \mathbb{R}^{nN}$, *and* $\mathcal{F}, \mathcal{G} : \mathbb{R}^{nN} \to \mathbb{R}^{nN}$ *are mappings defined as*

$$\mathcal{F}(\underline{\boldsymbol{z}}) = \begin{bmatrix} F_1(\boldsymbol{z}_1) \\ \vdots \\ F_N(\boldsymbol{z}_N) \end{bmatrix}, \quad \textit{and} \quad \mathcal{G}(\underline{\boldsymbol{z}}) = \begin{bmatrix} \langle \underline{\boldsymbol{z}} \rangle \\ \vdots \\ \langle \underline{\boldsymbol{z}} \rangle \end{bmatrix}, \tag{5.9}$$

where $\langle \underline{\boldsymbol{z}} \rangle \overset{def}{=} \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{z}_i$ *is the average of* $\underline{\boldsymbol{z}}$.

Algorithm 4 MACE Algorithm

Initialize $\underline{\boldsymbol{v}}^t = [\boldsymbol{v}_1^t, \dots, \boldsymbol{v}_N^t]$. $t = 1, \dots, T$ % Perform agent updates, $(2\mathcal{F} - \mathcal{I})(\underline{\boldsymbol{v}}^t)$

$$\begin{bmatrix} \boldsymbol{z}_1^t \\ \vdots \\ \boldsymbol{z}_N^t \end{bmatrix} = \begin{bmatrix} 2F_1(\boldsymbol{v}_1^t) - \boldsymbol{v}_1^t \\ \vdots \\ 2F_N(\boldsymbol{v}_N^t) - \boldsymbol{v}_N^t \end{bmatrix} \tag{5.10}$$

% Perform the data aggregation $(2\mathcal{G} - \mathcal{I})(\underline{\boldsymbol{z}}^t)$

$$\begin{bmatrix} \boldsymbol{v}_1^{t+1} \\ \vdots \\ \boldsymbol{v}_N^{t+1} \end{bmatrix} = \begin{bmatrix} 2\langle \underline{\boldsymbol{z}}^t \rangle - \boldsymbol{z}_1^t \\ \vdots \\ 2\langle \underline{\boldsymbol{z}}^t \rangle - \boldsymbol{z}_N^t \end{bmatrix} \tag{5.11}$$

Output $\langle \underline{\boldsymbol{v}}^T \rangle$.

Theorem 5.2.1 provides a full characterization of the MACE solution. The operator $\mathcal{G}$ in Theorem 5.2.1 is a consensus agent that takes a set of inputs $\boldsymbol{z}_1, \dots, \boldsymbol{z}_N$ and maps them to their average $\langle \underline{\boldsymbol{z}} \rangle$. In fact, we can show that $\mathcal{G}$ is a projection and that $(2\mathcal{G} - \mathcal{I})$ is its self-inverse [35]. As a result, (5.8) is equivalent to

$(2\mathcal{F}-\mathcal{I})\underline{\boldsymbol{v}}^* = (2\mathcal{G}-\mathcal{I})\underline{\boldsymbol{v}}^*$. That is, we want the individual agents $F_1, \ldots, F_N$ to match with the consensus agent $\mathcal{G}$ such that the equilibrium holds: $(2\mathcal{F}-\mathcal{I})\underline{\boldsymbol{v}}^* = (2\mathcal{G}-\mathcal{I})\underline{\boldsymbol{v}}^*$.

The algorithm of the MACE is illustrated in Algorithm 4. According to (5.8), $\underline{\boldsymbol{v}}^*$ is a fixed point of the set of equilibrium equations. Finding the fixed point can be done by iteratively updating $\underline{\boldsymbol{v}}^{(t)}$ through the procedure

$$\underline{\boldsymbol{v}}^{(t+1)} = (2\mathcal{G}-\mathcal{I})(2\mathcal{F}-\mathcal{I})\underline{\boldsymbol{v}}^{(t)}. \tag{5.12}$$

Therefore, the algorithmic steps are no more complicated than updating the individual agents $(2\mathcal{F}-\mathcal{I})$ in parallel, and then aggregating the results through $(2\mathcal{G}-\mathcal{I})$.

The convergence of MACE is guaranteed when $\mathcal{T}$ is non-expansive [35]summarized in the proposition below.

Proposition 5.2.1 *Let $\mathcal{F}$ and $\mathcal{G}$ be defined as* (5.9)*, and let $\mathcal{T} \overset{def}{=} (2\mathcal{G}-\mathcal{I})(2\mathcal{F}-\mathcal{I})$. Then the following results hold:*

(i) $\mathcal{F}$ is firmly non-expansive if all F_i's are firmly non-expansive.

(ii) $\mathcal{G}$ is firmly non-expansive.

(iii) $\mathcal{T}$ is non-expansive if $\mathcal{F}$ and $\mathcal{G}$ are firmly non-expansive.

Proof See Appendix. ■

5.3 Designing MACE Agents

After describing the MACE framework, in this section we discuss how each agent is designed for our problem.

5.3.1 Agent 1: Dual-Layer Closed-Form Matting

The first agent we use in MACE is a modified version of the classic closed-form matting. More precisely, we define the agent as

$$F_1(\boldsymbol{z}) = \operatorname*{argmin}_{\boldsymbol{\alpha}} \;\; \boldsymbol{\alpha}^T\boldsymbol{\alpha} + \lambda_1(\boldsymbol{\alpha}-\boldsymbol{z})^T\widetilde{\boldsymbol{D}}(\boldsymbol{\alpha}-\boldsymbol{z}), \tag{5.13}$$

where $\widetilde{\boldsymbol{L}}$ and $\widetilde{\boldsymbol{D}}$ are matrices, and will be explained below. The constant λ_1 is a parameter.

Review of Closed-Form Matting. To understand the meaning of (5.13), we recall that the classical closed-form matting is an algorithm that tries to solve

$$\begin{aligned} &J(\boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b}) \\ &= \sum_{j\in I}\left(\sum_{i\in w_j}\left(\alpha_i - \sum_c a_j^c I_i^c - b_j\right)^2 + \epsilon\sum_c (a_j^c)^2\right). \end{aligned} \tag{5.14}$$

Here, (a^r, a^g, a^b, b) are the linear combination coefficients of the color line model $\alpha_i \approx \sum_{c\in\{r,g,b\}} a^c I_i^c + b$, and α_i is the alpha matte value of the ith pixel [36]. The weight w_j is a 3×3 window of pixel j. With some algebra, we can show that the marginalized energy function $J(\boldsymbol{\alpha}) \overset{\text{def}}{=} \min_{\boldsymbol{a},\boldsymbol{b}} J(\boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b})$ is equivalent to

$$J(\boldsymbol{\alpha}) \overset{\text{def}}{=} \min_{\boldsymbol{a},\boldsymbol{b}} J(\boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{\alpha}^T\boldsymbol{L}\boldsymbol{\alpha}, \tag{5.15}$$

where $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ is the so-called matting Laplacian matrix. When trimap is given, we can regularize $J(\boldsymbol{\alpha})$ by minimizing the overall energy function:

$$\widehat{\boldsymbol{\alpha}} = \operatorname*{argmin}_{\boldsymbol{\alpha}} \;\; \boldsymbol{\alpha}^T\boldsymbol{L}\boldsymbol{\alpha} + \lambda(\boldsymbol{\alpha}-\boldsymbol{z})^T\boldsymbol{D}(\boldsymbol{\alpha}-\boldsymbol{z}), \tag{5.16}$$

where $\boldsymbol{D}$ is a binary diagonal matrix with entries being one for pixels that are labeled in the trimap, and zero otherwise. The vector $\boldsymbol{z} \in \mathbb{R}^n$ contains specified alpha values given by the trimap. Thus, for large λ, the minimization in (5.16) will force the solution to satisfy the constraints given by the trimap.

Dual-Layer Matting Laplacian $\widetilde{\boldsymbol{L}}$. In the presence of the plate image, we have two pieces of complementary information: $\boldsymbol{I} \in \mathbb{R}^{n\times 3}$ the color image containing the foreground object, and $\boldsymbol{P} \in \mathbb{R}^{n\times 3}$ the plate image. Correspondingly, we have alpha matte $\boldsymbol{\alpha}^I$ for $\boldsymbol{I}$, and the alpha matte $\boldsymbol{\alpha}^P$ for $\boldsymbol{P}$. When $\boldsymbol{P}$ is given, we can redefine the color line model as

$$\begin{bmatrix} \alpha_i^I \\ \alpha_i^P \end{bmatrix} \approx \sum_{c\in\{r,g,b\}} a^c \begin{bmatrix} I_i^c \\ P_i^c \end{bmatrix} + b. \tag{5.17}$$

In other words, we ask the coefficients (a^r, a^g, a^b, b) to fit simultaneously the actual image $\boldsymbol{I}$ and the plate image $\boldsymbol{P}$. When (5.17) is assumed, the energy function $J(\boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b})$ becomes

$$\begin{aligned} \widetilde{J}(\boldsymbol{\alpha}^I, \boldsymbol{\alpha}^P, \boldsymbol{a}, \boldsymbol{b}) = \sum_{j\in I} \Bigg\{ \sum_{i\in w_j} \Big(\alpha_i^I - \sum_c a_j^c I_i^c - b_j \Big)^2 \\ +\eta \sum_{i\in w_j} \Big(\alpha_i^P - \sum_c a_j^c P_i^c - b_j \Big)^2 + \epsilon \sum_c (a_j^c)^2 \Bigg\}, \end{aligned} \tag{5.18}$$

where we added a constant η to regulate the relative emphasis between $\boldsymbol{I}$ and $\boldsymbol{P}$.

Theorem 5.3.1 *The marginal energy function*

$$\widetilde{J}(\boldsymbol{\alpha}) \overset{def}{=} \min_{\boldsymbol{a},\boldsymbol{b}} \widetilde{J}(\boldsymbol{\alpha}, \boldsymbol{0}, \boldsymbol{a}, \boldsymbol{b}) \tag{5.19}$$

can be equivalently expressed as $\widetilde{J}(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^T \widetilde{\boldsymbol{L}} \boldsymbol{\alpha}$*, where* $\widetilde{\boldsymbol{L}} \in \mathbb{R}^{n \times n}$ *is the modified matting Laplacian, with the* (i, j)*th element*

$$\widetilde{L}_{i,j} = \sum_{k|(i,j)\in w_k} \Bigg\{ \delta_{ij} - \frac{1}{2|w_k|} \Bigg(1 + (\boldsymbol{I}_i - \boldsymbol{\mu}_k)^T \left(\boldsymbol{\Sigma}_k - n(1+\eta)\boldsymbol{\mu}_k \boldsymbol{\mu}_k^T\right)^{-1} (\boldsymbol{I}_j - \boldsymbol{\mu}_k) \Bigg) \Bigg\}. \tag{5.20}$$

Here, δ_{ij} *is the Kronecker delta,* $\boldsymbol{I}_i \in \mathbb{R}^3$ *is the color vector at the ith pixel. The vector* $\boldsymbol{\mu}_k \in \mathbb{R}^3$ *is defined as*

$$\boldsymbol{\mu}_k = \frac{1}{2|w_k|} \sum_{j \in w_k} (\boldsymbol{I}_j + \boldsymbol{P}_j), \tag{5.21}$$

and the matrix $\boldsymbol{\Sigma}_k \in \mathbb{R}^{3\times 3}$ *is*

$$\boldsymbol{\Sigma}_k = \frac{1}{2} \Bigg\{ \frac{1}{|w_k|} \sum_{j \in w_k} (\boldsymbol{I}_j - \boldsymbol{\mu}_k)(\boldsymbol{I}_j - \boldsymbol{\mu}_k)^T + \frac{1}{|w_k|} \sum_{j \in w_k} (\boldsymbol{P}_j - \boldsymbol{\mu}_k)(\boldsymbol{P}_j - \boldsymbol{\mu}_k)^T \Bigg\}. \tag{5.22}$$

Proof See Appendix. ■

Because of the plate term in (5.18), the modified matting Laplacian $\widetilde{\boldsymbol{L}}$ is positive definite. See Appendix for proof. The original $\boldsymbol{L}$ in (5.15) is only positive semi-definite.

Dual-Layer Regularization $\widetilde{\boldsymbol{D}}$. The diagonal regularization matrix $\widetilde{\boldsymbol{D}}$ in (5.13) is reminiscent to the binary matrix $\boldsymbol{D}$ in (5.16), but $\widetilde{\boldsymbol{D}}$ is defined through a sigmoid function applied to the input $\boldsymbol{z}$. To be more precise, we define $\widetilde{\boldsymbol{D}} \overset{\text{def}}{=} \text{diag}(\widetilde{d}_i)$, where

$$\widetilde{d}_i = \text{diag}\left\{ \frac{1}{1 + \exp\{-\kappa(z_i - \theta)\}} \right\} \in \mathbb{R}^{n \times n}, \tag{5.23}$$

and z_i is the i-th element of the vector $\boldsymbol{z} \in \mathbb{R}^n$, which is the argument of F_1. The scalar constant $\kappa > 0$ is a user defined parameter specifying the stiffness of the sigmoid function, and $0 < \theta < 1$ is another user defined parameter specifying the center of the transient. Typical values of (κ, θ) for our MACE framework are $\kappa = 30$ and $\theta = 0.8$.

A closer inspection of $\boldsymbol{D}$ and $\widetilde{\boldsymbol{D}}$ reveals that $\boldsymbol{D}$ is performing a hard-threshold whereas $\widetilde{\boldsymbol{D}}$ is performing a soft-threshold. In fact, the matrix $\boldsymbol{D} \overset{\text{def}}{=} \text{diag}(d_i)$ has diagonal entries

$$d_i = \begin{cases} 0, & \theta_1 < z_i < \theta_2, \\ 1, & \text{otherwise.} \end{cases} \tag{5.24}$$

for two cutoff values θ_1 and θ_2. This hard-threshold is equivalent to the soft-threshold in (5.23) when $\kappa \to \infty$.

There are a few reasons why (5.23) is preferred over (5.24), especially when we have the plate image. First, the soft-threshold in (5.23) tolerates more error present in $\boldsymbol{z}$, because the values of $\widetilde{\boldsymbol{D}}$ represent the probability of having foreground pixels. Second, the one-sided threshold in (5.23) ensures that the background portion of the image is handled by the plate image rather than the input $\boldsymbol{z}$. This is usually beneficial when the plate is reasonably accurate.

5.3.2 Agent 2: Background Estimator

Our second agent is a background estimator, defined as

$$F_2(\boldsymbol{z}) = \underset{\boldsymbol{\alpha}}{\text{argmin}} \quad \|\boldsymbol{\alpha} - \boldsymbol{r}_0\|^2 + \lambda_2 \|\boldsymbol{\alpha} - \boldsymbol{z}\|^2 + \gamma \boldsymbol{\alpha}^T (1 - \boldsymbol{\alpha}). \tag{5.25}$$

The reason of introducing F_2 is that in F_1, the matrix $\widetilde{\boldsymbol{D}}$ is determined by the current estimate $\boldsymbol{z}$. While $\widetilde{\boldsymbol{D}}$ handles part of the error in $\boldsymbol{z}$, large missing pixels and false alarms can still cause problems especially in the interior regions. The goal of F_2 is to complement F_1 for these interior regions.

Initial Background Estimate $\boldsymbol{r}_0$. Let us take a look at (5.25). The first two terms are quadratic. The interpretation is that given some fixed initial estimate $\boldsymbol{r}_0$ and the current input $\boldsymbol{z}$, $F_2(\boldsymbol{z})$ returns a linearly combined estimate between $\boldsymbol{r}_0$ and $\boldsymbol{z}$. The initial estimate $\boldsymbol{r}_0$ consists of two parts:

$$\boldsymbol{r}_0 = \boldsymbol{r}_c \odot \boldsymbol{r}_e, \tag{5.26}$$

where $\odot$ means elementwise multiplication. The first term $\boldsymbol{r}_c$ is the *color* term, measuring the similarity between foreground and background colors. The second term $\boldsymbol{r}_e$ is the *edge* term, measuring the likelihood of foreground edges relative background edges. In the followings we will discuss these two terms one by one.

Defining the Color Term $\boldsymbol{r}_c$. We define $\boldsymbol{r}_c$ by measuring the distance $\|\boldsymbol{I}_j - \boldsymbol{P}_j\|^2 = \sum_{c\in\{r,g,b\}}(I_j^c - P_j^c)^2$ between a color pixel $\boldsymbol{I}_j \in \mathbb{R}^3$ and a plate pixel $\boldsymbol{P}_j \in \mathbb{R}^3$. Ideally, we would like $\boldsymbol{r}_c$ to be small when $\|\boldsymbol{I}_j - \boldsymbol{P}_j\|^2$ is large.

In order to improve the robustness of $\|\boldsymbol{I}_j - \boldsymbol{P}_j\|^2$ against noise and illumination fluctuation, we modify $\|\boldsymbol{I}_j - \boldsymbol{P}_j\|^2$ by using the bilateral weighted average over a small neighborhood:

$$\Delta_i = \sum_{j\in\Omega_i} w_{ij}\|\boldsymbol{I}_j - \boldsymbol{P}_j\|^2, \tag{5.27}$$

where Ω_i specifies a small window around the pixel i. The bilateral weight w_{ij} is defined as

$$w_{ij} = \frac{\widetilde{w}_{ij}}{\sum_j \widetilde{w}_{ij}}, \tag{5.28}$$

where

$$\widetilde{w}_{ij} = \exp\left\{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2h_s^2}\right\}\exp\left\{-\frac{\|\boldsymbol{I}_i - \boldsymbol{I}_j\|^2}{2h_r^2}\right\}. \tag{5.29}$$

Here, $\boldsymbol{x}_i$ denotes the spatial coordinate of pixel i, $\boldsymbol{I}_i \in \mathbb{R}^3$ denotes the ith color pixel of the color image $\boldsymbol{I}$, and (h_s, h_r) are the parameters controlling the bilateral weight strength. The typical values of hs and hr are both 5.

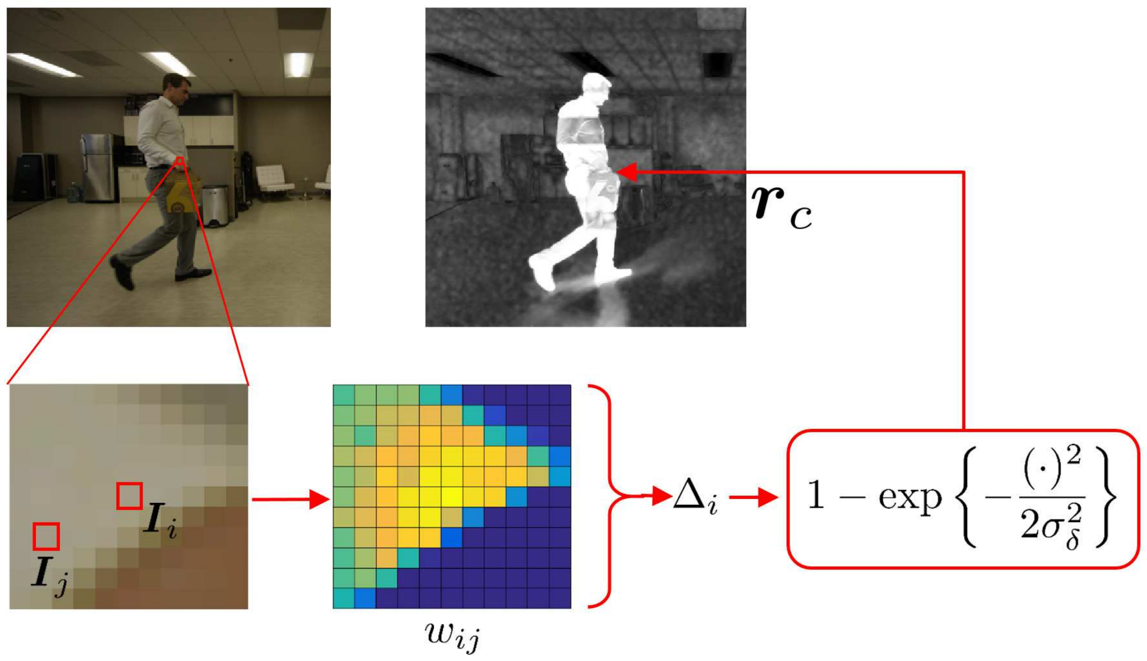


Fig. 5.5.: Illustration of how to construct the estimate $\boldsymbol{r}_c$. We compute the distance between the foreground and the background. The distance has a bilateral weight to improve robustness. The actual $\boldsymbol{r}_0$ represents the probability of having a foreground pixel.

We now need a mapping which maps the distance $\boldsymbol{\Delta} \overset{\text{def}}{=} [\Delta_1, \ldots, \Delta_n]^T$ to a vector of numbers $\boldsymbol{r}_c$ in $[0,1]^n$ so that the term $\|\boldsymbol{\alpha} - \boldsymbol{r}_0\|^2$ makes sense. To this end, we choose a simple Gaussian function:

$$\boldsymbol{r}_c = 1 - \exp\left\{-\frac{\boldsymbol{\Delta}^2}{2\sigma_\delta^2}\right\}, \tag{5.30}$$

where σ_δ is a user tunable parameter. We tested other possible mappings such as the sigmoid function and the cumulative distribution function of a Gaussian. However, we do not see significant difference compared to (5.30). The typical value for σ_δ is 10.

Defining the Edge Term $\boldsymbol{r}_e$. The color term $\boldsymbol{r}_c$ is able to capture most of the difference between the image and the plate. However, it also generates false alarms if there is illumination change. For example, shadow due to the foreground object is often falsely labeled as foreground. See the shadow near the foot in Figure 5.5.

In order to reduce the false alarm due to minor illumination change, we first create a "super-pixel" mask by grouping similar colors. Our super-pixels are generated by applying a standard flood-fill algorithm [142] to the image $\boldsymbol{I}$. This gives us a partition of the image $\boldsymbol{I}$ as

$$\boldsymbol{I} \rightarrow \{\boldsymbol{I}^{S_1}, \boldsymbol{I}^{S_2}, \ldots, \boldsymbol{I}^{S_m}\}, \tag{5.31}$$

where $S_1, \ldots, S_m$ are the m super-pixel index sets. The plate image is partition using the same super-pixel indices, i.e., $\boldsymbol{P} \rightarrow \{\boldsymbol{P}^{S_1}, \boldsymbol{P}^{S_2}, \ldots, \boldsymbol{P}^{S_m}\}$.

While we are generating the super-pixels, we also compute the gradients of $\boldsymbol{I}$ and $\boldsymbol{P}$ for every pixel $i = 1, \ldots, n$. Specifically, we define $\nabla \boldsymbol{I}_i = [\nabla_x \boldsymbol{I}_i, \nabla_y \boldsymbol{I}_i]^T$ and $\nabla \boldsymbol{P}_i = [\nabla_x \boldsymbol{P}_i, \nabla_y \boldsymbol{P}_i]^T$, where $\nabla_x \boldsymbol{I}_i \in \mathbb{R}^3$ (and $\nabla_y \boldsymbol{I}_i \in \mathbb{R}^3$) are the two-tap horizontal (and vertical) finite difference at the i-th pixel. To measure how far $\boldsymbol{I}_i$ is from $\boldsymbol{P}_i$, we compute

$$\theta_i = \|\nabla \boldsymbol{I}_i - \nabla \boldsymbol{P}_i\|_2. \tag{5.32}$$

Thus, θ_i is small for background regions because $\boldsymbol{I}_i \approx \boldsymbol{P}_i$, but is large when there is a foreground pixel in $\boldsymbol{I}_i$. If we set a threshold operation after θ_i, i.e., set $\theta_i = 1$ if $\theta_i > \tau_\theta$ for some threshold τ_θ, then shadows can be removed as their gradients are weak.

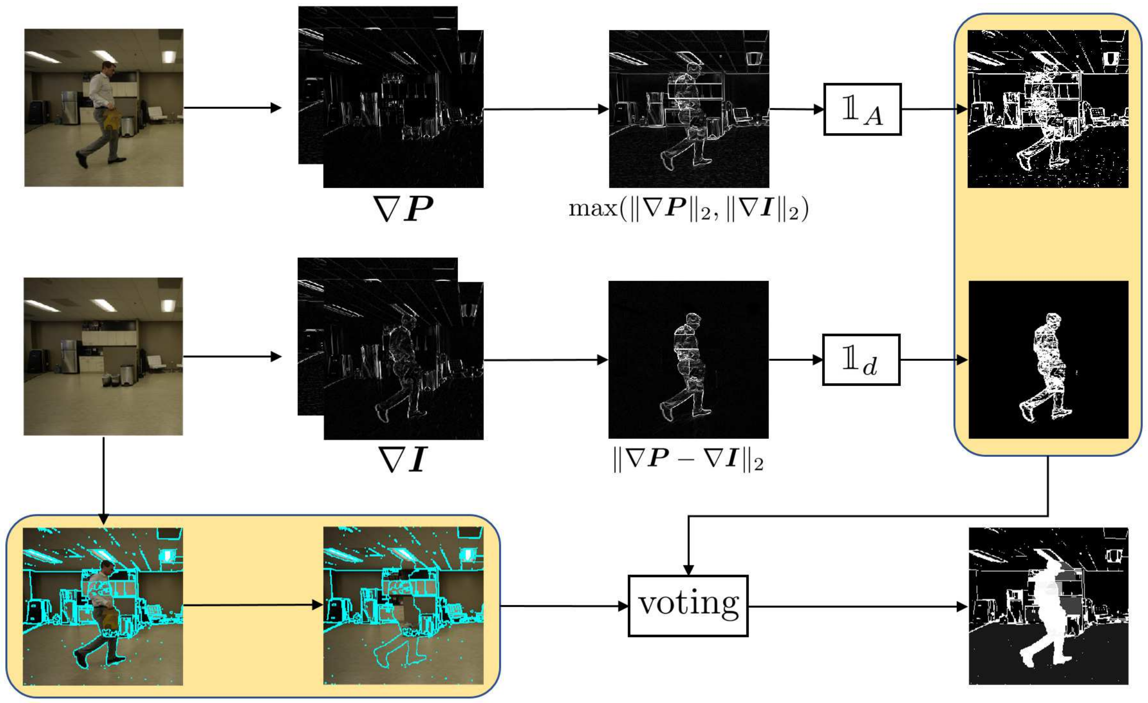
Now that we have computed θ_i, we still need to map it back to a quantity similar to the alpha matte. To this end, we compute a normalization term

$$A_i = \max\left(\|\nabla \boldsymbol{I}_i\|_2, \|\nabla \boldsymbol{P}_i\|_2\right), \tag{5.33}$$

and normalize $\mathbb{1}\{\theta_i > \tau_\theta\}$ by

$$(r_e)_i \overset{\text{def}}{=} \frac{\sum_{j \in S_i} \mathbb{1}\{A_i > \tau_A\} \mathbb{1}\{\theta_i > \tau_\theta\}}{\sum_{j \in S_i} \mathbb{1}\{A_i > \tau_A\}}, \tag{5.34}$$

where $\mathbb{1}$ denotes the indicator function, and τ_A and τ_θ are thresholds. In essence, (5.34) says in the i-th super-pixel S_i, we count the number of edges $\mathbb{1}\{\theta_i > \tau_\theta\}$ that



$\nabla \boldsymbol{P}$
$\max(\|\nabla \boldsymbol{P}\|_2, \|\nabla \boldsymbol{I}\|_2)$
$\mathbb{1}_A$
$\nabla \boldsymbol{I}$
$\|\nabla \boldsymbol{P} - \nabla \boldsymbol{I}\|_2$
$\mathbb{1}_d$
voting

Fig. 5.6.: Illustration of how to construct the estimate $\boldsymbol{r}_e$.

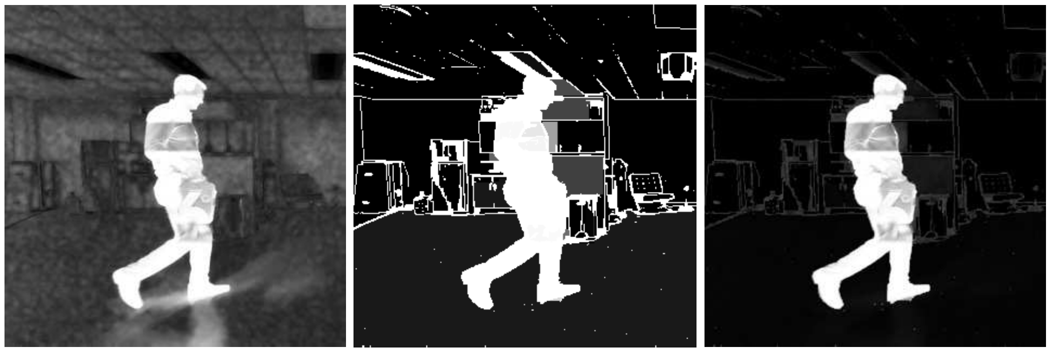


Fig. 5.7.: Comparison between $\boldsymbol{r}_c$, $\boldsymbol{r}_e$, and $\boldsymbol{r}_0$.

have strong difference between $\boldsymbol{I}_i$ and $\boldsymbol{P}_i$. However, we do not want to count every pixel but only pixels that already contains strong edges, either in $\boldsymbol{I}$ or $\boldsymbol{P}$. Thus, we take the weighted average using $\mathbb{1}\{A_i > \tau_A\}$ as the weight. This defines $\boldsymbol{r}_e$, as the weighted average $(r_e)_i$ is shared among all pixels in the super-pixel S_i. Figure 5.6 shows a pictorial illustration.

Why is $\boldsymbol{r}_e$ helpful? If we look at $\boldsymbol{r}_c$ and $\boldsymbol{r}_e$ in Figure 5.7, we see that the foreground pixels of $\boldsymbol{r}_c$ and $\boldsymbol{r}_e$ coincide but background pixels roughly cancel each other. The reason is that while $\boldsymbol{r}_c$ creates weak holes in the foreground, $\boldsymbol{r}_e$ fills the gap by ensuring the foreground is marked.

Regularization $\boldsymbol{\alpha}^T(1-\boldsymbol{\alpha})$. The last term $\boldsymbol{\alpha}^T(1-\boldsymbol{\alpha})$ in (5.25) is a regularization to force the solution to either 0 or 1. The effect of this term can be seen from the fact that $\boldsymbol{\alpha}^T(1-\boldsymbol{\alpha})$ is a symmetric concave quadratic function with a value zero for $\boldsymbol{\alpha} = \mathbf{1}$ or $\boldsymbol{\alpha} = \mathbf{0}$. Therefore, it introduces penalty for solutions that are away from 0 or 1. For $\gamma \leq 1$, one can show that the Hessian matrix of the function $f_2(\boldsymbol{\alpha}) = \|\boldsymbol{\alpha} - \boldsymbol{r}_0\|^2 + \gamma\boldsymbol{\alpha}^T(1-\boldsymbol{\alpha})$ is positive semidefinite. Thus, f_2 is strongly convex with parameter γ.

5.3.3 Agent 3: Total Variation Denoising

The third agent we use in this work is the total variation denoising:

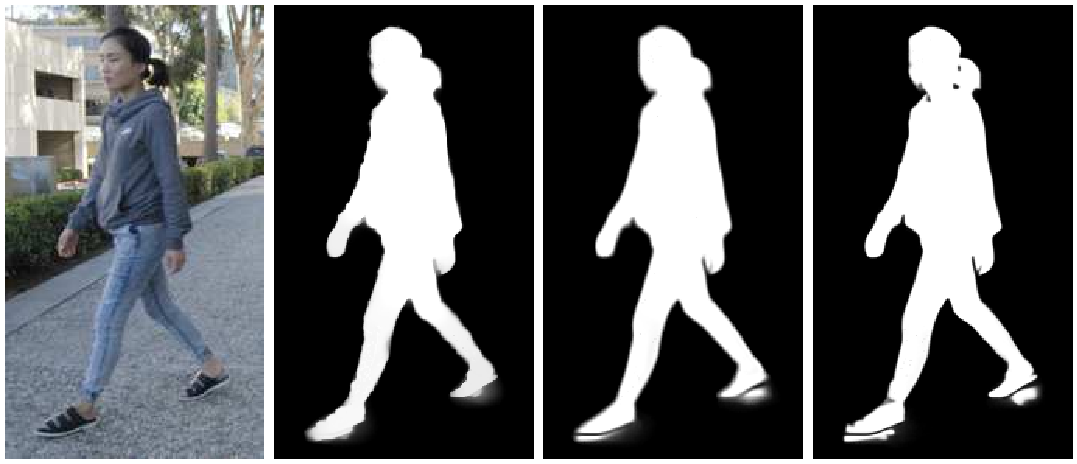
$$F_3(\boldsymbol{z}) = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \quad \|\boldsymbol{\alpha}\|_{\mathrm{TV}} + \lambda_3\|\boldsymbol{\alpha} - \boldsymbol{z}\|^2, \tag{5.35}$$

where λ_3 is a parameter. The norm $\|\cdot\|_{\mathrm{TV}}$ is defined in space-time:

$$\|\boldsymbol{v}\|_{\mathrm{TV}} \overset{\mathrm{def}}{=} \sum_{i,j,t} \sqrt{\beta_x(\nabla_x \boldsymbol{v})^2 + \beta_y(\nabla_y \boldsymbol{v})^2 + \beta_t(\nabla_t \boldsymbol{v})^2}, \tag{5.36}$$

where $(\beta_x, \beta_y, \beta_t)$ controls the relative strength of the gradient in each direction. In this work, for spatial total variation we set $(\beta_x, \beta_y, \beta_t) = (1, 1, 0)$, and for spatial-temporal total variation we set $(\beta_x, \beta_y, \beta_t) = (1, 1, 0.25)$.

A denoising agent is used in the MACE framework because we want to ensure smoothness of the resulting matte. The choice of the total variation denoising operation is a balance betweeen complexity and performance. Users can use stronger denoisers such as BM3D. However, these patch based image denoising algorithms rely on the patch matching procedure, and so they tend to under-smooth repeated patterns of false alarm / misses. Neural network denoisers are better candidates but they need to be trained with the specifically distorted alpha mattes. From our experience, we do not see any particular advantage of using CNN-based denoisers. Figure 5.8 shows some comparison.



(a) Input (b) TV [18] (c) BM3D [91] (d) IRCNN [143]

Fig. 5.8.: Comparison of different denoisers used in MACE. Shown are the results when MACE converges. The shadow near the foot is a typical place of false alarm, and many denoisers cannot handle.

5.3.4 Parameters and Runtime

The typical values for parameters of the proposed method are presented in Table 5.1. λ_1 and λ_2 are rarely changed, while λ_3 determines the denoising strength of Agent 3. γ has a default value of 0.05. Inceasing γ causes more binary results with clearer boundaries. τ_A and τ_θ determine the edge term $\boldsymbol{r}_e$ in Agent 2 and are fixed. σ_δ determines the color term $\boldsymbol{r}_c$ in Agent 2. Large σ_δ produces less false negative but more false positive. Overall, the performance is reasonably stable to these parameters.

Table 5.2.: Description of the video sequences used in our experiments.

		resolution	FGD %	time/Fr (sec)	indoor/ outdoor	shadow	lighting issues	Backgrd vibration	camouflage	green screen	ground truth
Purdue Dataset	Book	540x960	19.75%	231	outdoor			✓	✓		✓
	Building	632x1012	4.03%	170.8	outdoor	✓		✓	✓		✓
	Coach	790x1264	4.68%	396.1	outdoor			✓		✓	✓
	Studio	480x270	55.10%	58.3	indoor						✓
	Road	675x1175	1.03%	232.9	outdoor	✓		✓	✓		✓
	Tackle	501x1676	4.80%	210.1	outdoor	✓		✓		✓	✓
	Gravel	790x1536	2.53%	280.1	outdoor	✓		✓	✓		✓
	Office	623x1229	3.47%	185.3	indoor	✓			✓		✓
Public Dataset [110]	Bootstrap	480x640	13.28%	109.1	indoor	✓	✓		✓		✓
	Cespatx	480x640	10.31%	106.4	indoor	✓	✓		✓		✓
	DCam	480x640	12.23%	123.6	indoor	✓	✓		✓		✓
	Gen	480x640	10.23%	100.4	indoor	✓	✓		✓		✓
	Multipeople	480x640	9.04%	99.5	indoor	✓	✓		✓		✓
	Shadow	480x640	11.97%	115.2	indoor	✓	✓				✓

Table 5.1.: Typical values for parameters

Parameter	λ_1	λ_2	λ_3	γ	τ_A	τ_θ	σ_δ
Value	0.01	2	4	0.05	0.01	0.02	10

In terms of runtime, the most time-consuming part is Agent 1 because we need to solve a large-scale sparse least squares problem. Its runtime is determined by the number of foreground pixels. Table 5.2 shows the runtime of the sequences we tested. In generating these results, we used an un-optimized MATLAB code on a Intel i7-4770k. The typical runtime is about 1-3 minutes per frame. From our experience working with professional artists, even with professional film production software, e.g., NUKE, it takes 15 minutes to label a ground truth label using the plate and temporal cues. Therefore, the runtime benefit offered by our algorithm is substantial. The current runtime can be significantly improved by using multi-core CPU or GPU, because our algorithm handles each frame independently.

5.4 Experimental Results

5.4.1 Dataset

To evaluate the proposed method, we create a Purdue dataset containing 8 video sequences using the HypeVR Inc. 360 degree camera as shown in Figure 5.9. The original image resolution is 8192×4320 at a frame rate of 48fps, and these images are then downsampled and cropped to speed up the matting process. In addition to these videos, we also include 6 videos sequences from a public dataset [110], making a total of 14 video sequences. Snapshots of the sequences are shown in Figure 5.10. All video sequences are captured without camera motion. Plate images are available, either during the first or the last few frames of the video. To enable objective evaluation, for each video sequence we randomly select 10 frames and manually generate the ground truths. Thus totally there are 140 frames with ground truths.

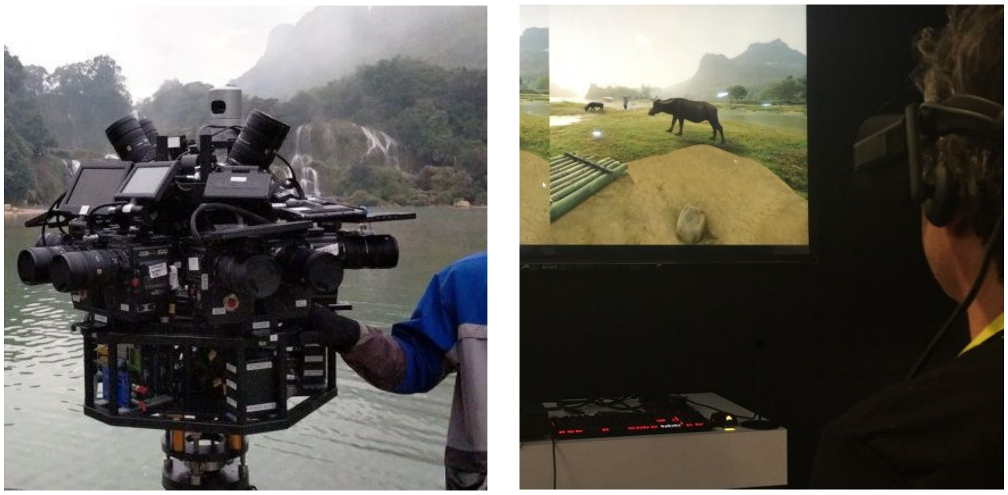
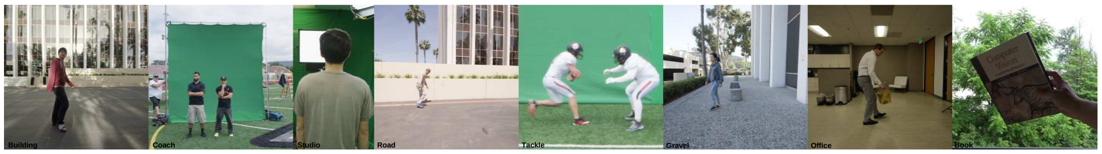
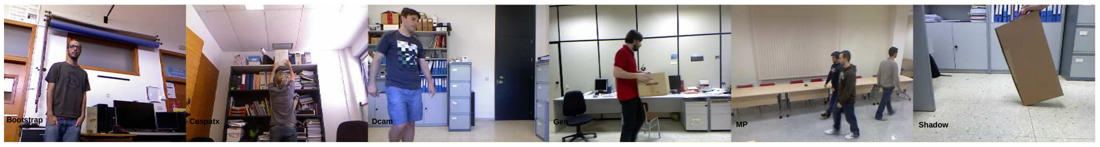


Fig. 5.9.: [Left] The camera system we used for this work. [Right] Display and headset to view the processed content.

The characteristics of the dataset is summarized in Table 5.2. The Purdue dataset has various resolution, and the Public dataset has one resolution 480×640. The foreground percentage for the Purdue dataset videos ranges from 1.03% to 55.10%, whereas that public dataset has similar foreground percentage around 10%. The runtime of the algorithm (per frame) is determined by the resolution and the foreground percentage. In terms of content, the Purdue dataset focuses on outdoor scenes whereas the public dataset are only indoor. The `shadow` column indicates the presence of shadow. `Lighting issues` include illumination change due to auto-exposure and auto-white-balance. The `background vibration` only applies to outdoor scenes where the background objects have minor movements, e.g., moving grass or tree



(a) Snapshots of the Purdue Dataset



(b) Snapshots of a public dataset [110]

Fig. 5.10.: Snapshots of the videos we use in the experiment. Top row: Building, Coach, Studio, Road, Tackle, Gravel, Office, Book. Bottom row:Bootstrap, Cespatx, Dcam, Gen, MP, Shadow.

Table 5.3.: Description of the competing methods.

	Methods		Supervised	Key idea
Unsupervised Video Segmentation	NLVS	[133]	no	non-local voting
	AGS	[105]	no	visual attention
	MOA	[137]	no	2-stream adaptation
	PDB	[136]	no	pyramid ConvLSTM
Alpha matting	Matting	[144]	trimap	Trimap generation + alpha matting
Background subtraction	ViBe	[127]	no	pixel model based
	Pbas	[104]	no	non-parametric
Other	BSVS	[117]	key frame	bilateral space
	Grabcut	[121]	plate	iterative graph cuts

branches. The `camouflage` column indicates the similarity in color between the foreground and background, which is a common problem for most sequences. The `green screen` column shows which of the sequences have green screens to mimic the common chroma-keying environment.

5.4.2 Competing methods

We categorize the competing methods into four different groups. The key ideas are summarized in Table 5.3.

- **Video Segmentation**: We consider four unsupervised video segmentation methods: Visual attention (AGS) [105], pyramid dilated bidirectional ConvLSTM (PDB) [136], motion adaptive object segmentation (MOA) [137], non-local consensus voting (NLVS) [133]. These methods are fully-automatic and do not require a plate image. All algorithms are downloaded from the author's websites and are run under default configurations.
- **Background Subtraction**: We consider two background subtraction algorithms Pixel-based adaptive segmenter (Pbas) [104], Visual background extractor (ViBe) [127]. Both algorithms are downloaded from the author's websites and are run under default configurations.
- **Alpha matting**: We consider one of the state-of-the-art alpha matting algorithm using CNN [103]. The trimaps are generated by applying frame difference between the plate and color images, followed by morphological and thresholding operations.
- **Others**: We consider the bilateral space video segmentation (BSVS) [117] which is a semi-supervised method. It requires the user to provide ground truth labels for key frames. We also modified the original Grabcut [121] to use the plate image instead of asking for user input.

5.4.3 Metrics

The following four metrics are used.

• **Intersection-ver-union (IoU)** measures the overlap between the estimate mask and the ground truth mask:

$$\text{IoU} = \frac{\sum_i \min(\widehat{x}_i, x_i)}{\sum_i \max(\widehat{x}_i, x_i)},$$

where $\widehat{x}_i$ is the i-pixel of the estimated alpha matte, and x_i is that of the ground truth. Higher IoU score is better.

Table 5.4.: Average results comparison with competing methods: AGS [105],PDB [136],MOA [137], NLVS [133], Trimap + DCNN [103], Pbas [104], ViBe [127], BSVS [117], Grabcut [121]. Higher intersection-over-union (IoU), higher Contour accuracy (F) [145], higher Structure measure (S) [146], lower MAE and lower Temporal instability (T) [145] indicate better performance.

		Unsupervised Video Segmentation				Matting	Bkgnd Subtract.		Others	
Metric	Our	AGS [105]	PDB [137]	MOA [137]	NLVS [133]	Tmap [144]	Pbas [104]	ViBe [127]	BSVS [117]	Gcut [121]
IoU	**0.9321**	0.8781	0.8044	0.7391	0.5591	0.7866	0.5425	0.6351	0.8646	0.6574
MAE	**0.0058**	0.0113	0.0452	0.0323	0.0669	0.0216	0.0842	0.0556	0.0093	0.0392
F [145]	**0.9443**	0.9112	0.8518	0.7875	0.6293	0.7679	0.6221	0.5462	0.8167	0.6116
S [146]	**0.9672**	0.938	0.8867	0.8581	0.784	0.9113	0.7422	0.8221	0.9554	0.8235
T [145]	**0.165**	0.1885	0.2045	0.1948	0.229	0.1852	0.328	0.2632	0.2015	0.232

• **Mean-absolute-error (MAE)** measures the average absolute difference between the ground truth and the estimate. Lower MAE is better.

• **Contour accuracy (F)** [145] measures the performance from a contour based perspective. Higher F score is better.

• **Structure measure (S)** [146] simultaneously evaluates region-aware and object-aware structural similarity between the result and the ground truth. Higher S score is better.

• **Temporal instability (T)** [145] that performs contour matching with polygon representations between two adjacent frames. Lower T score is better.

5.4.4 Results

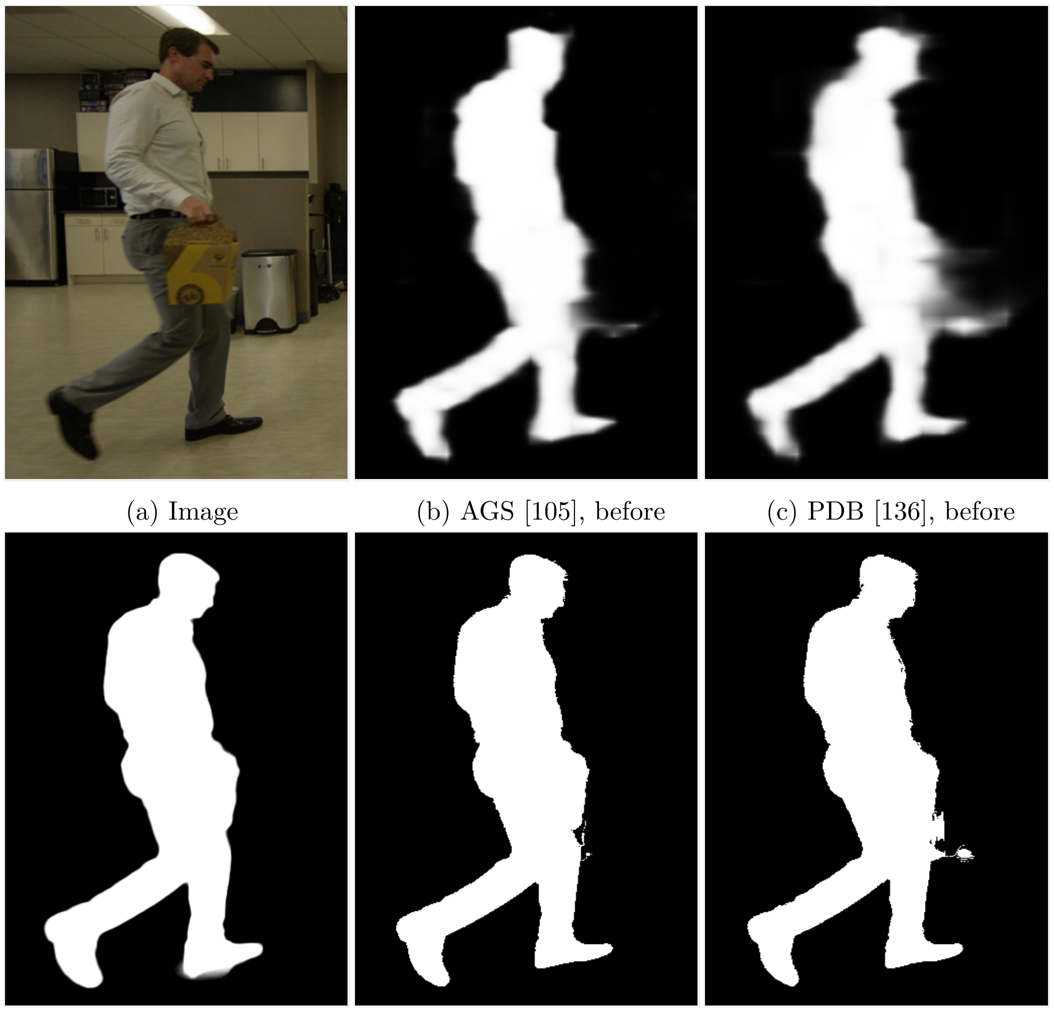
• **Comparison with video segmentation methods**: The results are shown in Table 5.4, where we list the average IoU, MAE, F, S and T scores over the datasets. In this table, we notice that the deep-learning solutions AGS [105], MOA [137] and PDB [136] are significantly better than classical optical flow based NLVS [133] in all the metrics. However, since the deep-learning solutions are targeting for saliency detection, foreground but unsalient objects will be missed. AGS performs the best among the three with a F measure of 0.91, S measure of 0.94 and T measure of 0.19. PDB performs better than MOA in most metrics other than the T measure, with PDB scoring 0.2 while MOA scoreing 0.19.

We should also comment on the reliance on conditional random field of these deep learning solutions. In Figure 5.11 we show the raw outputs of AGS [105] and PDB [136]. While the salient object is correctly identified, the masks are coarse. Only after the conditional random field [129] the results become significantly better. In contrast, the raw output of our proposed algorithm is already high quality.

• **Comparison with trimap + alpha-matting methods**: In this experiment we compare with several state-of-the-art alpha matting algorithms. The visual comparison is shown in Figure 5.4, and the performance of DCNN [103] is shown in Table 5.4. In order to make this method work, careful tuning during the trimap generation stage is required.

Figure 5.4 and Table 5.4 show that most alpha matting algorithms suffer from false alarms near the boundary, e.g., spectral matting [37], closed-form mating [36], learning-based matting [107] and comprehensive matting [109]. The more recent methods such as K-nearest neighbors matting [108] and DCNN [103] have equal amount of false alarm and miss. Yet, the overall performance is still worse than the proposed method and AGS. It is also worth noting that the matting approach achieves the second lowest T score (0.19), which is quite remarkable considering it is only a single-image method.

• **Comparison with background subtraction methods**: Background subtraction methods Pbas [104] and ViBe [127] are not able to obtain a score higher than 0.65 for IoU. Their MAE values are also significantly larger than the proposed method. Their temporal consistency is lagging by larger than 0.25 for T measure. Qualitatively, we observe that background subtraction methods perform most badly for scenes where the foreground objects are mostly stable or only have rotational movements. This is a common drawback of background subtraction algorithms, since they learn the background model in a online fashion and will gradually include non-moving objects



(a) Image (b) AGS [105], before (c) PDB [136], before

(d) Ours (e) AGS [105], after (f) PDB [136], after

Fig. 5.11.: Dependency of conditional random field. (a) Input. (b) Raw output of the neural network part of AGS [105]. (c) Raw output of neural network part of PDB [136]. (d) Our result without post-processing. (e) Post-processing of AGS using conditional random field. (f) Post-processing of PDB using conditional random field. Notice the rough raw output of the deep neural network parts.

into the background model. Without advanced design to ensure spatial and temporal consistency, the results also show errors even when the foreground objects are moving.

•

textbfComparison with other methods: Semi-supervised BSVS [117] requires ground truth key frames to learn a model. After the model is generated, the algorithm will

Table 5.5.: Ablation study of the algorithm. We show the performance by eliminating one of the agents, and replacing the denoising agent with other denoisers. Higher intersection-over-union (IoU), higher Contour accuracy (F) [145], higher Structure measure (S) [146], lower MAE and lower Temporal instability (T) [145] indicate better performance.

Metric	Our	w/o F_1	w/o F_2	w/oF_3	BM3D	IrCNN
IoU	**0.9321**	0.7161	0.7529	0.7775	0.8533	0.8585
MAE	**0.0058**	0.0655	0.0247	0.0368	0.0128	0.0121
F [145]	**0.9443**	0.7560	0.9166	0.6510	0.7718	0.8506
S [146]	**0.9672**	0.8496	0.9436	0.8891	0.9334	0.9320
T [145]	**0.165**	too large	0.1709	too large	0.1911	0.1817

overwrite the key frames with the estimates. When conducting this experiment, we ensure that the key frames used to generate the model are not used during testing. The result of this experiment shows that despite the key frames, BSVS [117] still performs worse than the proposed method. It is particularly weak when the background is complex where the key frames fail to form a reliable model.

The modified Grabcut [121] uses the plate image as a guide for the segmentation. However, because of the lack of additional prior models the algorithm does not perform well. This is particularly evident in images where colors are similar between foreground and background. Overall, Grabcut scores badly in most metrics, only slightly better than the background subtraction methods.

5.4.5 Ablation study

Since the proposed framework contains three different agents F_1, F_2 and F_3, we conduct an ablation study to verify the relative importance of the individual agents. To do so, we remove one of the three agents while keeping the other two fixed. The result is shown in Table 5.5. For T score, results for w/o F_1 and w/o F_3 are omitted,

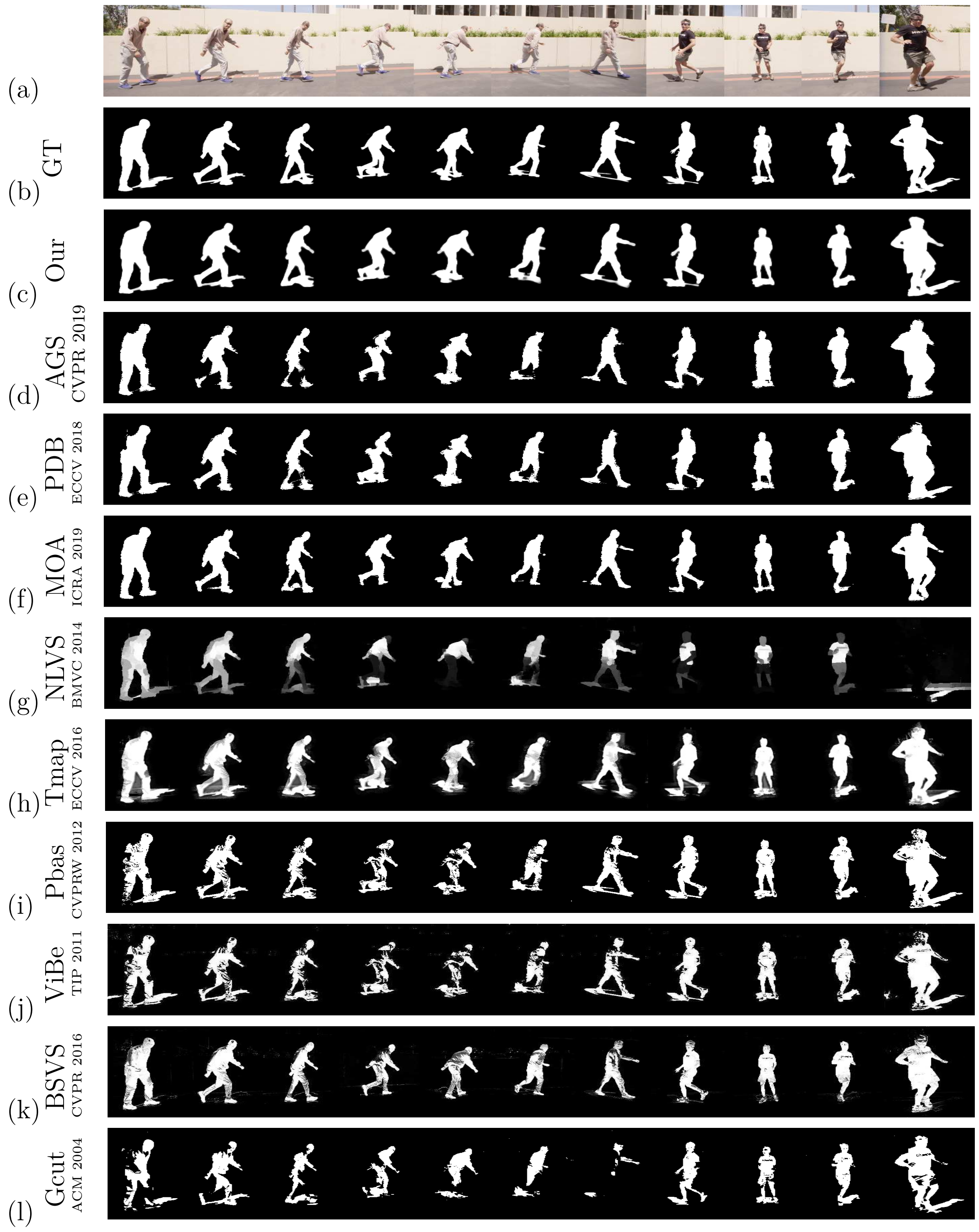


Fig. 5.12.: Office sequence results. (a) Input. (b) Ground truth. (c) Ours. (d) AGS [105]. (e) PDB [136]. (f)MOA [137] (g)NLVS [133]. (h) Trimap + DCNN [103]. (i) Pbas [104]. (j) ViBe [127]. (k)BSVS [117]. (l)Gcut [121].

as their results have many small regions of false alarms rendering untrackable amount of points on the polygon contours used in calculating T measure.

The matting agent F_1 has the most impact on the performance, followed by background estimator and denoiser. The drop in performance is most significant for hard sequences such as `Book` as it contains moving background, and `Road` as it contains strong color similarity between foreground and background. On average, we observe significant drop in IoU from 0.93 to 0.72 when the matting agent is absent. The F measure decreases from 0.94 to 0.76 as the boundaries are more erroneous without the matting agent. The structure measure also degrades from 0.97 to 0.85. The amount of error in the results also cause the T measure to become untrackable.

In this ablation study, we also observe spikes of error for some scenes when F_2 is absent. This is because, without the $\boldsymbol{\alpha}^T(1-\boldsymbol{\alpha})$ term in F_2, the result will look grayish instead of close-to-binary. This behavior leads to the error spikes. One thing worth noting is that the results obtained without F_2 do not drop significantly for S, F and T metric. This is due to the fact that IoU and MAE are pixel based metrics, whereas F, S and T are structural similarity. Therefore, even though the foreground becomes greyish without F_2, the structure of the labelled foreground is mostly intact.

For F_3, we observe that the total variation denoiser leads to the best performance for MACE. In a visual comparison shown in Figure 5.8, we observe that IRCNN [143] produces more detailed boundaries but fails to remove false alarms near the feet. BM3D [91] removes false alarms better but produces less detailed boundaries. TV on the other hand produces a more balanced result. As shown in Table 5.5, BM3D performs similarly as IRCNN scoring similar values for most metrics except that IrCNN scores 0.93 in F measure with BM3D only scoring 0.77 meaning more accurate contours. In general, even with different denoisers, the proposed method still outperforms most competing methods.

5.5 Limitations and Discussion

While the proposed method demonstrates superior performance than the state-of-the-art methods, it also has several limitations.

- **Quality of Plate Image**. The plate assumption may not hold when the background is moving substantially. When this happens, a more complex background model that includes dynamic information is needed. However, if the background is non-stationary, additional designs are needed to handle the local error and temporal consistency.

- **Loss of Fine Details**. In our proposed method, fine details such as hairs are compromised for robustness. Figure 5.13 illustrates an example. In some videos, the color difference between foreground and background is similar. This creates holes in the initial estimate $\boldsymbol{r}_0$, can be filled by a strong denoiser such as total variation. However, total variation is known to oversmooth fine details. To mitigate this issue, an additional post-processing step using alpha matting could bring back the details around the boundary.

- **Strong Shadows**. Strong shadows are sometimes treated as foreground, as shown in Figure 5.14. This is caused by the lack of shadow modeling in the problem formulation. The edge based initial estimate $\boldsymbol{r}_e$ can resolve the shadow issue to some extent, but not when the shadow is very strong. We tested a few off-the-shelf shadow removal algorithms [147–149], but generally they do not help because the shadow in our dataset can cast on the foreground object which should not be removed.

An open question here is whether our problem can be solved using deep neural networks since we have the plate. While this is certainly a feasible task because we can use the plate to replace the guided inputs (e.g., optical flow in [137] or visual attention in [105]), an appropriate training dataset is needed. In contrast, the proposed method has the advantage that it is training-free. Therefore, it is less susceptible to issues

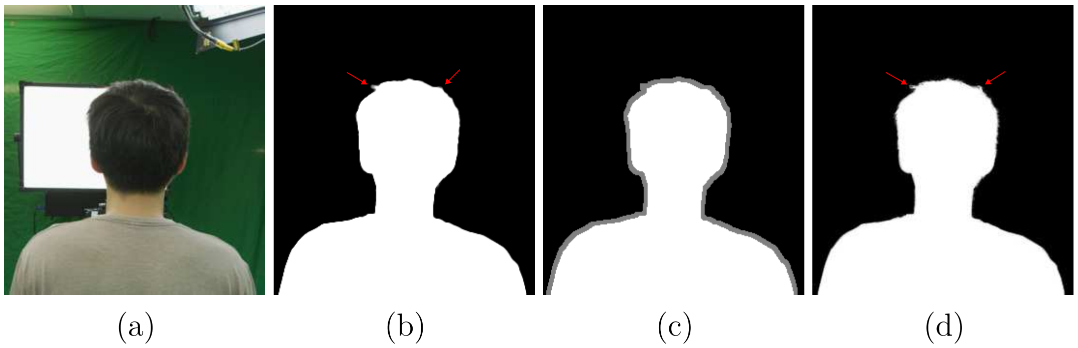


Fig. 5.13.: Limitation 1: Loss of fine details. (a) Color input. (b) Our result. (c) Improving our result by generating a trimap from (b). (d) post-processed result by alpha matting using (b).

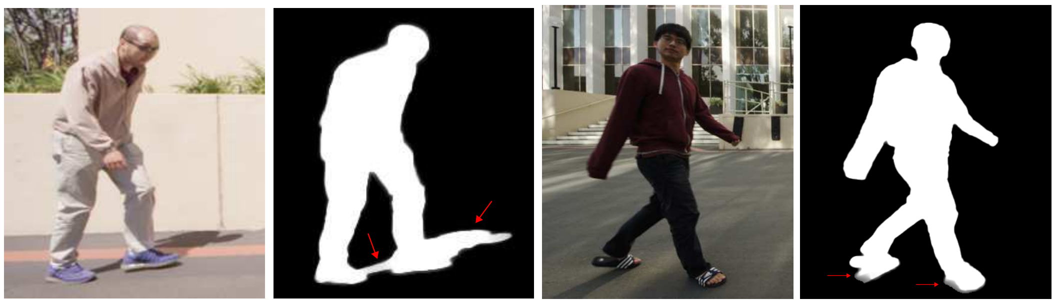


Fig. 5.14.: Limitation 2: Strong shadows. When shadows are strong, they are easily misclassified as foreground.

such as overfit. We should also comment that the MACE framework allows us to use deep neural network solutions. For example, one can replace F_1 with a deep neural network, and F_2 with another deep neural network. MACE is guaranteed to find a fixed point of these two agents if they do not agree.

5.6 Conclusion

This chapter presents a new foreground extraction algorithm based on the multi-agent consensus equilibrium (MACE) framework. MACE is an information fusion framework which integrates multiple weak experts to produce a strong estimator. Equipped with three customized agents: a dual-layer closed form matting agent, a background estimation agent and a total variation denoising agent, MACE offers substantially better foreground masks than state-of-the-art algorithms. MACE is a fully automatic algorithm, meaning that human interventions are not required. This provides significant advantage over semi-supervised methods which require trimaps or scribbles. In the current form, MACE is able to handle minor variations in the background plate image, illumination changes and weak shadows. Extreme cases can still cause MACE to fail, e.g., background movement or strong shadows. However, these could potentially be overcome by improving the background and shadow models.

5.7 Acknowledgement

The authors thank the anonymous reviewers for valuable feedback. This work is supported, in part, by the National Science Foundation under grants CCF-1763896 and CCF1718007.

5.8 Appendix

5.8.1 Proof of Theorem 2

Proof We start by writing (5.18) in the matrix form

$$\widetilde{J}(\boldsymbol{\alpha}^{\boldsymbol{I}}, \boldsymbol{\alpha}^{\boldsymbol{P}}, \boldsymbol{a}, \boldsymbol{b}) = \sum_{k \in \boldsymbol{I}} \left\| \begin{bmatrix} \boldsymbol{H}_k & \mathbf{1} \\ \sqrt{\eta}\boldsymbol{G}_k & \sqrt{\eta}\mathbf{1} \\ \sqrt{\epsilon}\boldsymbol{I}_{3\times 3} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_k \\ b_k \end{bmatrix} - \begin{bmatrix} \boldsymbol{\alpha}_k^{\boldsymbol{I}} \\ \boldsymbol{\alpha}_k^{\boldsymbol{P}} \\ \mathbf{0} \end{bmatrix} \right\|^2$$

where

$$\boldsymbol{H}_k = \begin{bmatrix} \vdots & \vdots & \vdots \\ I_i^r & I_i^g & I_i^b \\ \vdots & \vdots & \vdots \end{bmatrix}, \quad \boldsymbol{G}_k = \begin{bmatrix} \vdots & \vdots & \vdots \\ P_i^r & P_i^g & P_i^b \\ \vdots & \vdots & \vdots \end{bmatrix},$$

$$\boldsymbol{a}_k = \begin{bmatrix} a_k^r \\ a_k^g \\ a_k^b \end{bmatrix}, \quad \boldsymbol{\alpha}_k^{\boldsymbol{I}} = \begin{bmatrix} \vdots \\ \alpha_i^{\boldsymbol{I}} \\ \vdots \end{bmatrix}, \quad \boldsymbol{\alpha}_k^{\boldsymbol{P}} = \begin{bmatrix} \vdots \\ \alpha_i^{\boldsymbol{P}} \\ \vdots \end{bmatrix},$$

and i denotes the index of the i-th pixel in the neighborhood w_k. The difference with the classic closed-form matting [36] is the new terms $\boldsymbol{G}_k$, $\mathbf{1}$ and $\boldsymbol{\alpha}_k^{\boldsymbol{P}}$ (i.e., the second row of the quadratic function above.)

Denote

$$\boldsymbol{B}_k \overset{\text{def}}{=} \begin{bmatrix} \boldsymbol{H}_k & \mathbf{1} \\ \sqrt{\eta}\boldsymbol{G}_k & \sqrt{\eta}\mathbf{1} \\ \sqrt{\epsilon}\boldsymbol{I}_{3\times 3} & \mathbf{0} \end{bmatrix}, \tag{5.37}$$

and use the fact that $\boldsymbol{\alpha}^{\boldsymbol{P}} = \mathbf{0}$, we can find out the solution of the least-squares optimization:

$$\begin{bmatrix} \boldsymbol{a}_k \\ b_k \end{bmatrix} = (\boldsymbol{B}_k^T \boldsymbol{B}_k)^{-1} \boldsymbol{B}_k^T \begin{bmatrix} \boldsymbol{\alpha}_k^{\boldsymbol{I}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{5.38}$$

We now need to simplify the term $\boldsymbol{B}_k^T \boldsymbol{B}_k$. First, observe that

$$\begin{aligned} \boldsymbol{B}_k^T \boldsymbol{B}_k &= \begin{bmatrix} \boldsymbol{H}_k^T \boldsymbol{H}_k + \eta \boldsymbol{G}_k^T \boldsymbol{G}_k + \epsilon \boldsymbol{I}_{3\times 3} & \boldsymbol{H}_k^T \mathbf{1} + \eta \boldsymbol{G}_k^T \mathbf{1} \\ (\boldsymbol{H}_k \mathbf{1} + \eta \boldsymbol{G}_k^T \mathbf{1})^T & n(1+\eta) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{\Sigma}_k & \boldsymbol{\mu}_k \\ \boldsymbol{\mu}_k^T & c \end{bmatrix} \end{aligned}$$

where we define the terms $\boldsymbol{\Sigma}_k \overset{\text{def}}{=} \boldsymbol{H}_k^T\boldsymbol{H}_k + \eta\boldsymbol{G}_k^T\boldsymbol{G}_k + \epsilon\boldsymbol{I}$, $\boldsymbol{\mu}_k \overset{\text{def}}{=} \boldsymbol{H}_k^T\mathbf{1} + \eta\boldsymbol{G}_k^T\mathbf{1}$ and $c \overset{\text{def}}{=} n(1+\eta)$. Then, by applying the block inverse identity, we have

$$(\boldsymbol{B}_k^T\boldsymbol{B}_k)^{-1} = \begin{bmatrix} \boldsymbol{T}_k^{-1} & -\boldsymbol{T}_k^{-1}\widehat{\boldsymbol{\mu}}_k \\ -(\boldsymbol{T}_k^{-1}\widehat{\boldsymbol{\mu}}_k)^T & \frac{1}{c} + \widehat{\boldsymbol{\mu}}_k\boldsymbol{T}_k^T\widehat{\boldsymbol{\mu}}_k \end{bmatrix} \tag{5.39}$$

where we further define $\boldsymbol{T}_k = \boldsymbol{\Sigma}_k - \frac{\boldsymbol{\mu}_k\boldsymbol{\mu}_k^T}{c}$ and $\widehat{\boldsymbol{\mu}}_k = \frac{\boldsymbol{\mu}_k}{c}$.

Substituting (5.38) back to , and using (5.39), we have

$$\begin{aligned}(\boldsymbol{\alpha}^{\boldsymbol{I}}) &= \sum_k \left\| (\boldsymbol{I}_{3\times 3} - \boldsymbol{B}_k(\boldsymbol{B}_k^T\boldsymbol{B}_k)^{-1}\boldsymbol{B}_k^T) \begin{bmatrix} \boldsymbol{\alpha}_k^{\boldsymbol{I}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \right\|^2 \\ &= (\boldsymbol{\alpha}_k^{\boldsymbol{I}})^T \boldsymbol{L}_k \boldsymbol{\alpha}_k^{\boldsymbol{I}},\end{aligned}$$

where

$$\begin{aligned}\boldsymbol{L}_k = \boldsymbol{I}_{3\times 3} - \Big(&\boldsymbol{H}_k\boldsymbol{T}_K^{-1}\boldsymbol{H}_k^T - \boldsymbol{H}_k\boldsymbol{T}_k^{-1}\widehat{\boldsymbol{\mu}}_k\mathbf{1}^T \\ &- \mathbf{1}^T(\boldsymbol{T}_k^{-1}\widehat{\boldsymbol{\mu}}_k)^T\boldsymbol{H}_k + \frac{1}{c}\mathbf{1}^T\widehat{\boldsymbol{\mu}}_k\boldsymbol{T}_k^{-1}\widehat{\boldsymbol{\mu}}_k\mathbf{1}\Big)\end{aligned} \tag{5.40}$$

The (i,j)-th element of $\boldsymbol{L}_k$ is therefore

$$\begin{aligned}\boldsymbol{L}_k(i,j) =&\delta_{ij} - (\boldsymbol{I}_{ki}^T\boldsymbol{T}_k^{-1}\boldsymbol{I}_{kj} - \boldsymbol{I}_{ki}^T\boldsymbol{T}_k^{-1}\widehat{\boldsymbol{\mu}}_k \\ &- \widehat{\boldsymbol{\mu}}_k^T\boldsymbol{T}_k^{-1}\boldsymbol{I}_{kj} + \frac{1}{c} + \widehat{\boldsymbol{\mu}}_k^T\boldsymbol{T}_k^{-1}\widehat{\boldsymbol{\mu}}^k) \\ =&\delta_{ij} - (\frac{1}{c} + (\boldsymbol{I}_{ki} - \widehat{\boldsymbol{\mu}}_k)^T\boldsymbol{T}_k^{-1}(\boldsymbol{I}_{kj} - \widehat{\boldsymbol{\mu}}_k))\end{aligned} \tag{5.41}$$

Adding terms in each w_k, we finally obtain

$$\widetilde{L}_{i,j} = \sum_{k|(i,j)\in w_k} \Big\{\delta_{ij} - (\frac{1}{c} + (\boldsymbol{I}_{ki} - \widehat{\boldsymbol{\mu}}_k)^T\boldsymbol{T}_k^{-1}(\boldsymbol{I}_{kj} - \widehat{\boldsymbol{\mu}}_k))\Big\}.$$

■

5.8.2 Proof: $\tilde{\boldsymbol{L}}$ is positive definite

Proof Recall the definition of $\widetilde{J}(\boldsymbol{\alpha}^I, \boldsymbol{\alpha}^P, \boldsymbol{a}, \boldsymbol{b})$:

$$\widetilde{J}(\boldsymbol{\alpha}^I, \boldsymbol{\alpha}^P, \boldsymbol{a}, \boldsymbol{b}) = \sum_{j\in I}\left\{\sum_{i\in w_j}\left(\alpha_i^I - \sum_c a_j^c I_i^c - b_j\right)^2 \right.$$
$$\left. +\eta\sum_{i\in w_j}\left(\alpha_i^P - \sum_c a_j^c P_i^c - b_j\right)^2 + \epsilon\sum_c (a_j^c)^2\right\}$$

Based on Theorem 2 we have,

$$\widetilde{J}(\boldsymbol{\alpha}) \overset{\text{def}}{=} \min_{\boldsymbol{a},\boldsymbol{b}} \; \widetilde{J}(\boldsymbol{\alpha}, \boldsymbol{0}, \boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{\alpha}^T\widetilde{\boldsymbol{L}}\boldsymbol{\alpha}. \tag{5.42}$$

We consider two cases: (i) $a_j^c = 0 \ \forall j$ and $\forall c$, (ii) there exists some j and c such that $a_j^c \neq 0$. For the second case, is larger than 0. For the first case, can be reduced into

$$\widetilde{J}(\boldsymbol{\alpha}, 0, \boldsymbol{a}, \boldsymbol{b}) = \sum_{j\in I}\left\{\sum_{i\in w_j}\left((\alpha_i - b_j)^2 + \eta\,(-b_j)^2\right)\right\} \tag{5.43}$$

For any vector $\alpha \neq 0$, there exists at least one $\alpha_i \neq 0$. Then by completing squares we can show that

$$\begin{aligned}
&(\alpha_i - b_j)^2 + \eta b_j^2 \\
&= \alpha_i^2 - 2\alpha_i b_j + (1+\eta)b_j^2 \\
&= \left(\sqrt{\frac{1}{1+\eta}}\alpha_i - \sqrt{1+\eta}\,b_j\right)^2 + \frac{\eta}{1+\eta}\alpha_i^2 > 0
\end{aligned}$$

Therefore, $\widetilde{J}(\boldsymbol{\alpha}, 0, \boldsymbol{a}, \boldsymbol{b}) > 0$ for any non-zero vector α. As a result, $\widetilde{J}(\boldsymbol{\alpha}, 0, \boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{\alpha}^T\widetilde{\boldsymbol{L}}\boldsymbol{\alpha} > 0$ for both cases, and $\widetilde{\boldsymbol{L}}$ is positive definite. ■

5.8.3 Proof of Proposition 1

Proof Let $\underline{\boldsymbol{x}} \in \mathbb{R}^{nN}$ and $\underline{\boldsymbol{y}} \in \mathbb{R}^{nN}$ be two super-vectors.

(i). If the F_i's are non-expansive, then

$$\begin{aligned}
&\|\mathcal{F}(\underline{\boldsymbol{x}}) - \mathcal{F}(\underline{\boldsymbol{y}})\|^2 + \|\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}} - (\mathcal{F}(\underline{\boldsymbol{x}}) - \mathcal{F}(\underline{\boldsymbol{y}}))\|^2 \\
&= \sum_{i=1}^{N} \left(\|F_i(\boldsymbol{x}_i) - F_i(\boldsymbol{x}_i)\|^2 + \|\boldsymbol{x}_i - \boldsymbol{y}_i - (F_i(\boldsymbol{x}_i) - F_i(\boldsymbol{y}_i))\|^2\right) \\
&\overset{(c)}{\leq} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{y}_i\|^2 = \|\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}}\|^2
\end{aligned}$$

where (c) holds because each F_i is firmly non-expansive. As a result, $\mathcal{F}$ is also firmly non-expansive.

(ii). To prove that $\mathcal{G}$ is firmly non-expansive, we recall from Theorem 1 that $2\mathcal{G} - \mathcal{I}$ is self-inverse. Since $\mathcal{G}$ is linear, it has a matrix representation. Thus, $\|(2\mathcal{G} - \mathcal{I})\boldsymbol{x}\|^2 = \boldsymbol{x}^T(2\mathcal{G} - \mathcal{I})^T(2\mathcal{G} - \mathcal{I})\boldsymbol{x}$. Because $\mathcal{G}$ is an averaging operator, it has to be symmetric, and hence $\mathcal{G}^T = \mathcal{G}$. As a result, we have $\|(2\mathcal{G} - \mathcal{I})\boldsymbol{x}\|^2 = \|\boldsymbol{x}\|^2$ for any $\boldsymbol{x}$, which implies non-expansiveness.

(iii). If $\mathcal{F}$ and $\mathcal{G}$ are both firmly non-expansive, we have

$$\begin{aligned}
&\|(2\mathcal{G} - \mathcal{I})[(2\mathcal{F} - \mathcal{I})(\underline{\boldsymbol{x}})] - (2\mathcal{G} - \mathcal{I})[(2\mathcal{F} - \mathcal{I})(\underline{\boldsymbol{y}})]\|^2 \\
&\overset{(a)}{\leq} \|(2\mathcal{F} - \mathcal{I})(\underline{\boldsymbol{x}}) - (2\mathcal{F} - \mathcal{I})(\underline{\boldsymbol{y}})\|^2 \overset{(b)}{\leq} \|\underline{\boldsymbol{x}} - \underline{\boldsymbol{y}}\|^2
\end{aligned}$$

where (a) is true due to the firmly non-expansiveness of $\mathcal{G}$ and (b) is true due to the non-expansiveness of $\mathcal{G}$. Thus, $\mathcal{T} \overset{\text{def}}{=} (2\mathcal{G} - \mathcal{I})(2\mathcal{F} - \mathcal{I})$ is non-expansive. This result also implies convergence of the MACE algorithm, due to [35]. ■

6. SUMMARY

In this work, we proposed multiple applications of the Plug-and-Play ADMM on image deblurring, image inpainting/interpolation, image super-resolution and single-photon imaging and discussed their efficient implimentations. For the case of super-resolution we proposed a closed form solution for the x subproblem of the ADMM algorithm that beats alternative methods like conjugate gradients on both performance and runtime.

We introduced a modofied version of the Plug-and-Play ADMM with a continuation scheme that updates the internal parameter ρ of the ADMM automatically and proved fixed point convergence for the proposed alsogirhtm when the denoiser falls under a class called ”bounded denoisers”. Our definition for the ”bounded denoiser” is weaker than the known ”proper denoiser” but excludes trivial ones like $\mathcal{D}_\sigma(\boldsymbol{x}) = 0$. Compared to existing convergence analysis for Plug-and-Play ADMM that guarantees global convergence with symmetric smoothing filter, we compromise the convergence type to a fixed-point convergence but broaden the range of denoisers to ”bounded denoisers”.

We studied the relation between ADMM and the generalized approximate message passing. A parameter-free version of the Plug-and-play ADMM is proposed. Instead of using a vectorized weight norm as in generalized approximate message passing, we adopt a scalar parameter for the data term. The scalar is derived using the divergence of the previous subproblem so that the current subproblem will adapt based on the reliability of the previous subproblem. When calculating the divergence of the off-the-shelf denoiser, we use the Monte Carlo scheme that performs denoising twice.

We further extend our study to a variant of the Plug-and-Play ADMM called consensus equiblirum, and developed a system for video foreground extraction used in the virtual reality content creation pipeline. We design one dual-layer alpha matting agent, one background estimator using both color and edge cues, and adaopt the off-the-shelf total variation denoiser. Through experimental results, we show that the proposed multi-agent concensus equilibrium algorihtm outperforms other state-of-the-art competing background subtraction and video segmentation algorithms.

In the future, we will explore more operators for the MACE framework. One potential operator we would like to try is to include the NLSV [133] as one of the operator of our system. Our current system does not exploit any information regarding the temporal consistency in the image sequence. Although this might enable us to process frames in a parallel manner, but the quality of our results could suffer from temporal inconsistency such as flickering, etc. The NLVS method associates super pixels accross frames based on their similarities. At the same time, since NLVS does not require a stable background image, we can also potentially lift the requirement for a stationary camera so that our algorithm can be used for the more general cases.

Another solution for improving temporal consistency is to calculate optical flow across frames and apply warping followed by a 3D denoiser that remove both spatial and temporal noise. However, we suspect that optical flow could do more bad than good when its results are not accurate enough. Another approach is to use a camera that can capture videos with high frame rate so that direct application of 3D denois ers is reasonable due to the high temporal resolution, however this might limit the application of our algorithms for videos. However, we suspect that optical flow could do more bad than good when its results are not accurate enough. Another approach is to use a camera that can capture videos with high frame rate so that direct application of 3D denoisers is reasonable due to the high temporal resolution, however this might limit the application of our algorithms for videos.

Our current assumption is that the alpha matte should be locally constant so that the total variational denoiser makes sense. However in reality the alpha mattes can sometimes be spatially inconsistent, especially when the object has unclear boundaries such as hair or fur. In these cases, our algorithm can overly smooth the the fuzzy boundaries of the object. However, removing the denoiser will cause the results to suffer from noise. One solution is to keep the denoiser and add an additional stage of post processing after the CE algorithm to enhance the details along the boundaries. Once the we know the locations of boundaries based on CE solutions we can target specifically on these area.

Our current foreground probability operator requires a user defined parameter to convert the difference between the plate and target frames to a probability with the range [0,1]. Based on experiments, the parameter that can achieve the best results can vary among different scenes. As a result, it is highly preferable that we design an automatic scheme for determine the value if this parameter based on each video. One solution is to reference the results of a background subtraction algorithm. Background subtraction algorithms build a background model based on the scene, often times these algorithms have consistent performance for difference scenes. We think it might be reasonable to determine the value of the parameter in our foreground probability operator based on the difference between the result from a background subtraction algorithm and the probability map we can obtain with a specific parameter value. In this way, we can even find the best parameter per frame in case the scene are complex and varies largely at difference sections.

REFERENCES

REFERENCES

[1] P. Campisi and K. Egiazarian, *Blind image deconvolution: theory and applcations.* CRC press, Apr 2016.

[2] M. Jiang, “Blind deblurring of spiral ct images,” *IEEE Trans. on Medical Imaging*, vol. 22, no. 7, pp. 837–845, Jul 2003.

[3] R. Dahl, M. Norouzi, and J. Shlens, “Pixel recursive super resolution,” in *Proc. IEEE International Conference on Computer Vision. (ICCV)*, 2017, pp. 5439–5448.

[4] S. H. Chan, X. Wang, and O. A. Elgendy, “Plug-and-play ADMM for image restoration,” *IEEE Trans. on Computational Imaging*, vol. 3, no. 1, pp. 84–98, Mar. 2017.

[5] M. Duarte, D. Takhar, J. Laska, T. Sun, K. Kelly, and R. Baraniuk, “Single-pixel imaging via compressive sampling,” *IEEE signal processing magazine*, vol. 25, no. 2, pp. 83–91, 2008.

[6] R. Glowinski and A. Marrocco, “Sur l pproximation, par elements finis d rdre un, et la resolution, par penalisation-dualit e, d ne classe de problems de dirichlet non lineares,” *Revue Fran caise d utomatique, Informatique, et Recherche Op erationelle*, vol. 9, pp. 41–76, 1975.

[7] D. Gabay and B. Mercier, “A dual algorithm for the solution of nonlinear variational problems via finite element approximations,” *Computers and Mathematics with Applications*, vol. 2, pp. 17–40, 1976.

[8] M. Fortin and R. Glowinski, *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems.* North-Holland: Amsterdam: Elsevier, Jan 1983, vol. 15.

[9] R. Glowinski and P. L. Tallec, “Augmented lagrangian methods for the solution of variational problems,” WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER, No. MRC-TSR-2965, Jan 1987.

[10] M. Fukushima, “Application of the alternating direction method of multipliers to separable convex programming problems,” *Computational Optimization and Applications*, vol. 1, no. 1, pp. 93–111, Oct 1992.

[11] J. Eckstein and M. Fukushima, *Large Scale Optimization: State of the Art.* Boston, MA: Springer, 1994, ch. Some reformulations and applications of the alternating direction method of multipliers, pp. 119–138.

[12] G. Chen and M. Teboulle, “A proximal-based decomposition method for convex minimization problems,” *Mathematical Programming*, vol. 64, no. 1-3, pp. 81–101, Mar 1994.

[13] D. Gabay, *Studies in mathematics and its applications.* North-Holland: Amsterdam: Elsevier, Jan 1983, vol. 15, ch. Applications of the method of multipliers to variational inequalities, pp. 299–331.

[14] J. Eckstein and D. Bertsekas, “On the douglas achford splitting method and the proximal point algorithm for maximal monotone operators,” *Mathematical Programming*, vol. 55, no. 1-3, pp. 293–318, Apr 1992.

[15] M. Fadili and J. Starck, “Monotone operator splitting for optimization problems in sparse recovery,” in *Proc. IEEE International Conference on Image Processing (ICIP).* IEEE, Nov 2009, pp. 1461–1464.

[16] M. A. T. Figueiredo and J. M. Bioucas-Dias, “Restoration of Poissonian images using alternating direction optimization,” *IEEE Trans. Image Process.*, vol. 19, no. 12, pp. 3133–3145, Dec. 2010.

[17] G. Steidl and T. Teuber, “Removing multiplicative noise by douglas-rachford splitting methods,” *Journal of Mathematical Imaging and Vision*, vol. 36, no. 2, pp. 168–184, Feb 2010.

[18] S. H. Chan, R. Khoshabeh, K. B. Gibson, P. E. Gill, and T. Q. Nguyen, “An augmented Lagrangian method for total variation video restoration,” *IEEE Trans. on Image Processing*, vol. 20, no. 11, pp. 3097–3111, May 2011.

[19] P. Combettes and J. Pesquet, *Fixed-point algorithms for inverse problems in science and engineering.* New York, NY: Springer, 2011, ch. Proximal splitting methods in signal processing, pp. 185–212.

[20] I. Schizas and G. Giannakis, “Consensus in ad hoc wsns with noisy links art ii: Distributed estimation and smoothing of random signals,” *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1650–1666, Apr 2008.

[21] Wright, Nowak, and Figueiredo, “Sparse reconstruction by separable approximation,” *IEEE Trans. Signal Processing*, vol. 57, no. 9, pp. 2479–2493, Jul 2009.

[22] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM J. Imaging Sciences*, vol. 2, no. 1, pp. 181–202, Mar. 2009.

[23] Bioucas-Dias and Figueiredo, “A new twist: Two-step iterative shrinkage/thresholding algorithms for image restoration,” *IEEE Trans. Image Process.*, vol. 16, no. 12, pp. 2992–3004, Dec 2007.

[24] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine learning*, vol. 3, no. 1, pp. 1–22, Jul. 2011.

[25] S. Venkatakrishnan, C. Bouman, and B. Wohlberg, “Plug-and-play priors for model based reconstruction,” in *Proc. IEEE Global Conference on Signal and Information Processing*, Dec. 2013, pp. 945–948.

[26] S. Sreehari, S. V. Venkatakrishnan, B. Wohlberg, G. T. Buzzard, L. F. Drummy, J. P. Simmons, and C. A. Bouman, “Plug-and-play priors for bright field electron tomography and sparse interpolation,” *IEEE Trans. on Computational Imaging*, vol. 2, no. 4, pp. 408–423, Dec. 2016.

[27] A. Teodoro, J. Bioucas-Dias, and M. Figueiredo, "Image restoration and reconstruction using targeted plug-and-play priors," *IEEE Transactions on Computational Imaging*, May 2019.

[28] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (red)," *SIAM Journal on Imaging Sciences*, vol. 10, no. 4, pp. 1804–1844, Oct. 2017.

[29] E. Reehorst and P. Schniter, "Regularization by denoising: Clarifications and new interpretations," *IEEE Transactions on Computational Imaging*, vol. 5, no. 1, pp. 52–67, Nov. 2018.

[30] Y. Sun, S. X. andY. Li, L. Tian, B. Wohlberg, and U. Kamilov, "Regularized fourier ptychography using an online plug-and-play algorithm," in *InICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE. IEEE, May 2019, pp. 7665–7669.

[31] S. Chan, "Performance analysis of plug-and-play admm: A graph signal processing perspective," *IEEE Transactions on Computational Imaging*, vol. 5, no. 2, pp. 274–286, Jan. 2019.

[32] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," Nov. 2015, available online: http://arxiv.org/abs/1410.1390.

[33] P. Milanfar, "Symmetrizing smoothing filters," *SIAM Journal on Imaging Sciences*, vol. 6, no. 1, pp. 263–284, 2013.

[34] M. Borgerding and P. Schniter, "Generalized approximate message passing for the cosparse analysis model," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process. (ICASSP)*, 2015, pp. 3756–3760.

[35] G. Buzzard, S. H. Chan, S. Sreehari, and C. A. Bouman, "Plug-and-play unplugged: optimization free reconstruction ufing consensus equilibrium," 2017, accepted to SIAM J. Imaging Sciences. Available online http://arxiv.org/abs/1705.08983.

[36] A. Levin, D. Lischinski, and Y. Weiss, "A closed-form solution to natural image matting," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 228–242, Feb. 2008.

[37] A. Levin, A. Rav-Acha, and D. Lischinski, "Spectral matting," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1699–1712, Oct. 2008.

[38] J. Sun, J. Jia, C. K. Tang, and H. Shum, "Poisson matting," *ACM Trans. on Graphics (ToG)*, vol. 23, no. 3, pp. 315–321, Aug. 2004.

[39] J. Wang and M. F. Cohen, "Optimized color sampling for robust matting," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2007, pp. 1–8.

[40] Achanta, Shaji, Smith, Lucchi, Fua, and Susstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, Nov 2012.

[41] J. Dahl, P. C. Hansen, S. H. Jensen, and T. L. Jensen, "Algorithms and software for total variation image reconstruction via first-order methods," *Numerical Algorithms*, vol. 53, no. 1, pp. 67–92, Jan. 2010.

[42] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "BM3D-AMP: A new image recovery algorithm based on BM3D denoising," in *Proc. IEEE Intl. Conf. Image Process.*, Sep. 2015, pp. 3116–3120.

[43] H. V. Poor, *An Introduction to Signal Detection and Estimation*, 2nd ed. Springer, 1998.

[44] C. A. Bouman. (2015) Model-based image processing. Available online: https://engineering.purdue.edu/~bouman/publications/pdf/MBIP-book.pdf.

[45] M. Afonso, J. Bioucas-Dias, and M. Figueiredo, "Fast image recovery using variable splitting and constrained optimization," *IEEE Trans. Image Process.*, vol. 19, no. 9, pp. 2345–2356, Apr. 2010.

[46] J. Yang, Y. Zhang, and W. Yin, "An efficient TVL1 algorithm for deblurring multichannel images corrupted by impulsive noise," *SIAM J. on Sci. Comput.*, vol. 31, no. 4, pp. 2842–2865, Jul. 2009.

[47] D. Garcia, "Robust smoothing of gridded data in one and higher dimensions with missing values," *Comput. Statist. Data Anal.*, vol. 54, no. 4, pp. 1167–1178, Apr. 2010.

[48] M. Zhou, H. Chen, L. Ren, G. Sapiro, L. Carin, and J. W. Paisley, "Non-parametric Bayesian dictionary learning for sparse image representations," in *Advances in Neural Information Processing Systems*, vol. 22, 2009, pp. 2295–2303.

[49] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. European Conference on Computer Vision (ECCV)*, Sep. 2014, pp. 184–199.

[50] T. Peleg and M. Elad, "A statistical prediction model based on sparse representations for single image super-resolution," *IEEE Trans. Image Process.*, vol. 23, no. 6, pp. 2569–2582, Jun. 2014.

[51] H. He and W. Siu, "Single image super-resolution using Gaussian process regression," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Jun. 2011, pp. 449–456.

[52] J. Yang, J. Wright, T. Huang, and Y. Ma, "Image super-resolution as sparse representation of raw image patches," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Jun. 2008, pp. 1–8.

[53] Y. Dar, A. M. Bruckstein, M. Elad, and R. Giryes, "Postprocessing of compressed images via sequential denoising," Oct. 2015, available online: http://arxiv.org/abs/1510.09041.

[54] A. Rond, R. Giryes, and M. Elad, "Poisson inverse problems by the plug-and-play scheme," *Journal of Visual Communication and Image Representation*, vol. 41, pp. 96–108, Nov. 2015.

[55] A. Brifman, Y. Romano, and M. Elad, "Turning a denoiser into a super-resolver using plug and play priors," in *Proc IEEE Intl. Conf. Image Process.*, Sep. 2016, pp. 1404–1408.

[56] P. Milanfar, "A tour of modern image filtering," *IEEE Signal Processing Magazine*, vol. 30, pp. 106–128, Jan. 2013. [Online]. Available: http://cs.ucsc.edu/~milanfar/publications/journal/ModernTour_FinalSubmission.pdf

[57] S. H. Chan, T. Zickler, and Y. M. Lu, "Fast non-local filtering by random sampling: it works, especially for large images," in *Proc. IEEE Intl. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2013, pp. 1603–1607.

[58] S. H. Chan, "Algorithm-induced prior for image restoration," Feb. 2016, available online at http://arxiv.org/abs/1602.00715.

[59] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "From denoising to compressed sensing," Jul. 2014, available online: http://arxiv.org/abs/1406.4175.

[60] J. Tan, Y. Ma, and D. Baron, "Compressive imaging via approximate message passing with wavelet-based image denoising," in *Proc. IEEE Global Conf. Signal Information Process.*, Dec. 2014, pp. 424–428.

[61] ——, "Compressive imaging via approximate message passing with image denoising," *IEEE Trans. Signal Process.*, vol. 63, no. 8, pp. 2085–2092, Apr. 2015.

[62] R. Neelamani, H. Choi, and R. Baraniuk, "ForWaRD: Fourier-Wavelet regularized deconvolution for ill-conditioned systems," *IEEE Trans. Signal Process.*, vol. 52, no. 2, pp. 418–433, Feb. 2004.

[63] A. Danielyan, V. Katkovnik, and K. Egiazarian, "BM3D frames and variational image deblurring," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1715–1728, Aug 2012.

[64] J. Zhang, D. Zhao, and W. Gao, "Group-based sparse representation for image restoration," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3336–3351, Aug. 2014.

[65] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdu, and M. J. Weinberger, "Universal discrete denoising: Known channel," *IEEE Trans. Information Theory*, vol. 51, no. 1, pp. 5–28, Jan. 2005.

[66] K. Sivaramakrishnan and T. Weissman, "A context quantization approach to universal denoising," *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2110–2129, Jun 2009.

[67] A. Levin and B. Nadler, "Natural image denoising: Optimality and inherent bounds," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Jun. 2011, pp. 2833–2840.

[68] A. Levin, B. Nadler, F. Durand, and W. Freeman, "Patch complexity, finite pixel correlations and optimal denoising," in *Proc. 12th European Conf. Computer Vision (ECCV)*, vol. 7576, Oct. 2012, pp. 73–86.

[69] A. Buades, B. Coll, and J. Morel, "A review of image denoising algorithms, with a new one," *SIAM Multiscale Model and Simulation*, vol. 4, no. 2, pp. 490–530, 2005. [Online]. Available: http://epubs.siam.org/doi/pdf/10.1137/040616024

[70] Y. Ma, J. Zhu, and D. Baron, "Approximate message passing with universal denoising and Gaussian mixture learning," *IEEE Trans. Image Process.*, vol. 64, no. 21, pp. 5611–5622, Nov. 2016.

[71] K.-M. Ng, "A continuation approach for solving nonlinear optimization problems with discrete variables," Ph.D. dissertation, Stanford University, 2002.

[72] Z. T. Harmany, R. F. Marcia, and R. M. Willet, "This is SPIRAL-TAP: sparse Poisson intensity reconstruction algorithms: Theory and practice," *IEEE Trans. Image Process.*, vol. 21, no. 3, pp. 1084–1096, Sep. 2011.

[73] Y. Wang, Y. Yang, W. Yin, and Y. Zhang, "A new alternating minimization algorithm for total variation image reconstruction," *SIAM J. Imaging Science*, vol. 1, no. 3, pp. 248–272, Aug. 2008.

[74] T. Goldstein, B. O. Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," UCLA, Tech. Rep., 2012, available online: ftp://ftp.math.ucla.edu/pub/camreport/cam12-35.pdf.

[75] N. Nguyen, P. Milanfar, and G. Golub, "Efficient generalized cross-validation with applications to parametric image restoration and resolution enhancement," *IEEE Trans. Image Process.*, vol. 10, no. 9, pp. 1299–1308, Sep. 2001.

[76] S. Ramani, T. Blu, and M. Unser, "Monte-Carlo SURE: A black-box optimization of regularization parameters for general denoising algorithms," *IEEE Trans. Image Process.*, vol. 17, no. 9, pp. 1540–1554, 2008.

[77] J.-J. Moreau, "Proximité et dualité dans un espace Hibertien," *Bulletin de la Société Mathématique de France*, vol. 93, pp. 273–299, 1965.

[78] S. Wiggins, *Introduction to Applied Nonlinear Dynamical Systems and Chaos.* New York: Springer-Verlag, 1990.

[79] L. Liu, S. H. Chan, and T. Q. Nguyen, "Depth reconstruction from sparse samples: representation, algorithm, and sampling," *IEEE Trans. Image Process.*, vol. 24, no. 6, pp. 1983–1996, Jun. 2015.

[80] M. S. C. Almeida and M. Figueiredo, "Deconvolving images with unknown boundaries using the alternating direction method of multipliers," *IEEE Trans. Image Process.*, vol. 22, no. 8, pp. 3074–3086, Aug. 2013.

[81] ——, "Deconvolving images with unknown boundaries using the alternating direction method of multipliers," *IEEE Trans. Image Process.*, vol. 22, no. 8, pp. 3074–3086, Aug. 2013.

[82] P. P. Vaidyanathan, *Multirate Systems and Filter Banks.* Prentice Hall, 1992.

[83] A. Matakos, S. Ramani, and J. A. Fessler, "Accelerated edge-preserving image restoration without boundary artifacts," *IEEE Trans. Image Process.*, vol. 22, no. 5, pp. 2019–2029, Jan 2013.

[84] E. R. Fossum, "The Quanta Image Sensor (QIS): Concepts and Challenges," in *Proc OSA Topical Mtg Computational Optical Sensing and Imaging*, Jul. 2011, paper JTuE1. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=ISA-2011-JTuE1

[85] S. H. Chan and Y. M. Lu, "Efficient image reconstruction for gigapixel quantum image sensors," in *Proc IEEE Global Conf. on Signal and Information Processing (GlobalSIP'14)*, Dec. 2014, pp. 312–316.

[86] J. Huang, A. Singh, and N. Ahuia, "Single image super-resolution using transformed self-exemplars," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 5197–5206.

[87] F. Yang, Y. M. Lu, L. Sbaiz, and M. Vetterli, "Bits from photons: Oversampled image acquisition using binary poisson statistics," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1421–1436, Apr. 2012.

[88] O. A. Elgendy and S. H. Chan, "Image reconstruction and threshold design for quanta image sensors," in *Proc. IEEE Intl' Conf. Image Process. (ICIP)*, Sep. 2016, pp. 978–982.

[89] R. Sinkhorn, "A relationship between arbitrary positive matrices and doubly stochastic matrices," *The Annals of Mathematical Statistics*, vol. 35, pp. 876–879, 1964.

[90] F. Heide, M. Steinberger, Y.-T. Tsai, M. Rouf, D. Pajak, D. Reddy, O. Gallo, J. L. abd W. Heidrich, K. Egiazarian, J. Kautz, and K. Pulli, "FlexISP: A flexible camera image processing framework," *ACM Transactions on Graphics (Proceedings SIGGRAPH Asia 2014)*, vol. 33, no. 6, Dec. 2014.

[91] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3D transform-domain collaborative filtering," *IEEE Trans. on Image Processing*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[92] S. Rangan, P. Schniter, E. Riegler, A. Fletcher, and V. Cevher, "Fixed points of generalized approximate message passing with arbitrary matrices," in *Proc. IEEE Int. Symp. Information Theory*, 2013, pp. 664–668.

[93] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *Proc. IEEE Int. Symp. Information Theory*, 2011, pp. 2168–2172.

[94] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," *Proc. Nat. Acad. Sci.*, vol. 106, no. 45, pp. 18 914–18 919, 2009.

[95] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: nonlinear phenomena*, vol. 60, no. 1-4, pp. 259–268, Nov. 1992.

[96] J. Vila, P. Schniter, S. Rangan, F. Krzakala, and L. Zdeborova, "Adaptive damping and mean removal for the generalized approximate message passing algorithm," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process. (ICASSP)*, 2015, pp. 2021–2025.

[97] S. Rangan, P. Schniter, and A. K. Fletcher, "On the convergence of approximate message passing with arbitrary matrices," in *Proc. IEEE Int. Symp. Information Theory*, 2014, pp. 236–240.

[98] M. Zyda, "From visual simulation to virtual reality to games," *Computer*, vol. 38, no. 9, pp. 25–32, Sep. 2005.

[99] F. Biocca and M. R. Levy, Eds., *Communication in the age of virtual reality.* Routledge, Feb. 2013.

[100] G. C. Burdea and P. Coiffet, *Virtual reality technology.* John Wiley & Sons, Jun. 2003, vol. 1.

[101] R. P. McMahan, D. A. Bowman, D. J. Zielinski, and R. B. Brady, "Evaluating display fidelity and interaction fidelity in a virtual reality game," *IEEE Trans. on Visualization and Computer Graphics*, vol. 18, no. 4, pp. 626–633, Apr. 2012.

[102] J. Chen, D. Bautembach, and S. Izadi, "Scalable real-time volumetric surface reconstruction," *ACM Trans. on Graphics (ToG)*, vol. 32, no. 4, Jul 2013, article 113.

[103] D. Cho, Y. Tai, and I. Kweon, "Natural image matting using deep convolutioonal neural networks," in *Proc. European Conference on Computer Vision (ECCV).* Springer, Oct. 2016, pp. 626–643.

[104] M. Hofmann, P. Tiefenbacher, and G. Rigoll, "Background segmentation with feedback: The pixel-based adaptive segmenter," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2012, pp. 38–43.

[105] W. Wang, H. Song, S. Zhao, J. Shen, S. Zhao, S. Hoi, and H. Ling, "Learning unsupervised video object segmentation through visual attention," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE. IEEE, Jun. 2019, pp. 3064–3074.

[106] S. Shimoda, M. Hayashi, and Y. Kanatsugu, "New chroma-key imagining technique with hi-vision background," *IEEE Transactions on broadcasting*, vol. 35, no. 4, pp. 357–361, Dec. 1989.

[107] Y. Zheng and C. Kambhamettu, "Learning based digital matting," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, Sep. 2009, pp. 889–896.

[108] Q. Chen, D. Li, and C. Tang, "KNN matting," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 9, pp. 2175–2188, Sep. 2013.

[109] E. Shahrian, D. Rajan, B. Price, and S. Cohen, "Improving image matting using comprehensive sampling sets," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2013, pp. 636–643.

[110] M. Camplani, L. Maddalena, G. M. Alcover, A. Petrosino, and L. Salgado, "A benchmarking framework for background subtraction in RGBD videos," in *New Trends in Image Analysis and Processing-ICIAP 2017 Workshops.* Springer, Sep. 2017, pp. 219–229.

[111] Y. Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski, "A Bayesian approach to digital matting," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, Dec 2001, pp. 11–18.

[112] E. Gastal and M. Oliveira, "Shared sampling for real-time alpha matting," *Euro Graphics*, vol. 29, no. 2, pp. 575–584, 2010.

[113] K. He, C. Rhemann, X. Tang, and J. Sun, "A global sampling method for alpha matting," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE. IEEE, Jun. 2011, pp. 2049–2056.

[114] N. Xu, B. Price, and T. Huang, "Deep image matting," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 311–320.

[115] J. Tang, Y. Aksoy, C. Oztireli, M. Gross, and T. Aydin, "Learning-based sampling for natural image matting," in *InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE. IEEE, 2019, pp. 3055–3065.

[116] W. Xi, J. Chen, L. Qian, and J. Allebach, "High-accuracy automatic person segmentation with novel spatial saliency map," in *IEEE International Conference on Image Processing*, IEEE. IEEE, Sep. 2019.

[117] N. Marki, F. Perazzi, O. Wang, and A. Sorkine-Hornung, "Bilateral space video segmentation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 743–751.

[118] S. A. Ramakanth and R. V. Babu, "Seamseg: Video segmentation using patch seams," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, Jun. 2014, pp. 376–383.

[119] S. D. Jain and K. Grauman, "Supervoxel-consistent foreground propagation in video," in *Proc. European Conference on Computer Vision (ECCV)*. Springer, Sep. 2014, pp. 656–671.

[120] C. Hsieh and M. Lee, "Automatic trimap generation for digital image matting," in *Proc. IEEE Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, Oct. 2013, pp. 1–5.

[121] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interatcive foreground extraction using iterated graph cuts," *ACM Trans. on Graphics (ToG)*, vol. 23, no. 3, pp. 309–314, Aug. 2004.

[122] J. Cho, T. Yamasaki, K. Aizawa, and K. H. Lee, "Depth video camera based temporal alpha matting for natural 3d scene generation," in *3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*, May 2011, pp. 1–4.

[123] O. Wang, J. Finger, Q. Yang, J. Davis, and R. Yang, "Automatic natural video matting with depth," in *15th Pacific Conference On Computer Graphics and Applications*, Oct. 2007, pp. 469–472.

[124] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proc. IEEE International Conference on Pattern Recognition*, vol. 2, Aug. 2004, pp. 28–31.

[125] J. W. Davis and V. Sharma, “Fushion-based background-subtraction using contour saliency,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2005, pp. 11–19.

[126] V. Mahadevan and N. Vasconcelos, “Background subtraction in highly dynamic scenes,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2008, pp. 1–6.

[127] O. Barnich and V. D. Marc, “ViBe: A universal background subtraction algorithm for video sequences,” *IEEE Trans. on Image Processing*, vol. 20, no. 6, pp. 1709–1724, Jun. 2011.

[128] Y. Lee and K. Grauman, “Key-segments for video object segmentation,” in *In 2011 International conference on computer vision*, IEEE. IEEE, Nov. 2011, pp. 1995–2002.

[129] P. Krahenbuhl and V. Koltun, “Efficient inference in fully connected crfs with gaussian edge potentials,” in *InAdvances in neural information processing systems*, 2011, pp. 109–117.

[130] T. Ma and L. Latecki, “Maximum weight cliques with mutex constraints for video object segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE. IEEE, Jun. 2012, pp. 670–677.

[131] F. Perazzi, P. Krahenbuhl, Y. Pritch, and A. Mornung, “Saliency filters: Contrast based filtering for salient region detection,” in *In2012 IEEE conference on computer vision and pattern recognition (CVPR)*, IEEE. IEEE, Jun. 2012, pp. 733–740.

[132] A. Papazouglou and V. Ferrari, “Fast object segmentation in unconstrained video,” in *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, IEEE. IEEE, 2013, pp. 1777–1784.

[133] A. Faktor and M. Irani, “Video segmentation by non-local consensus voting,” in *Proc. British Machine Vision Association (BMVC)*, vol. 2, Jun. 2014, p. 8.

[134] W. Wang, J. Shen, X. Li, and F. Porikli, “Robust video object cosegmentation,” *IEEE Transactions on Image Processing (TIP)*, vol. 24, no. 10, pp. 3137–3148, Jun. 2015.

[135] W. Jang, C. Lee, and C. Kim, “Primary object segmentation in videos via alternate convex optimization of foreground and background distributions,” in *InProceedings of the IEEE conference on computer vision and pattern recognition*, IEEE. IEEE, 2016, pp. 696–704.

[136] H. Song, W. Wang, S. Zhao, J. Shen, and K. Lam, “Pyramid dilated deeper convlstm for video salient object detection,” in *InProceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[137] M. Dehghan, Z. Zhang, M. Siam, J. Jin, L. Petrich, and M. Jagersand, “Online tool and task learning via human robot interaction,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, available online https://arxiv.org/pdf/1809.08722.pdf.

[138] W. Wang, J. Shen, F. Porikli, and R. Yang, "Semi-supervised video object segmentation with super-trajectories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 4, pp. 985–998, Mar. 2018.

[139] W. Wang, Q. Lai, H. Fu, J. Shen, and H. Ling, "Salient object detection in the deep learning era: An in-depth survey," *Preprint*, Apr. 2019, available on arXiv: https://arxiv.org/pdf/1904.09146.pdf.

[140] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127–239, Jan. 2014.

[141] X. Wang and S. H. Chan, "Parameter-free-plug-and-play ADMM for image restoration," in *Proc. IEEE International Conference on Acoustic, Speech, Signal Process. (ICASSP)*, Mar. 2017, pp. 1323–1327.

[142] S. V. Burtsev and Y. P. Kuzmin, "An efficient flood-fill algorithm," *Computers & Graphics*, vol. 17, no. 5, pp. 549–561, Sept 1993.

[143] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep cnn denoiser for image restoration," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2808–2817.

[144] D. Cho, S. Kim, and Y. W. Tai, "Automatic trimap generation and consistent matting for light-field images," *IEEE Trans. on pattern Analysis and Machine Intelligence*, vol. 39, no. 8, pp. 1504–1517, Aug. 2017.

[145] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE. IEEE, 2016, pp. 724–732.

[146] D. Fan, M. Chen, Y. Liu, T. Li, and A. Borji, "Structure-measure: A new way to evaluate foreground maps," in *InProceedings of the IEEE international conference on computer vision (CVPR)*, IEEE. IEEE, 2017, pp. 4548–4557.

[147] R. Guo, Q. Dai, and D. Hoiem, "Single-image shadow detection and removal using paired regions," in *proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2011, pp. 2033–2040.

[148] E. Arbel and H. Hel-Or, "Shadow removal using intensity surfaces and texture anchor points," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, no. 6, pp. 1202–1216, Jun. 2011.

[149] F. Liu and M. Gleicher, "Texture-consistent shadow removal," in *Proc. European Conference on Computer Vision (ECCV)*. Springer, Oct. 2008, pp. 437–450.

VITA

VITA

Xiran Wang received the B.Sc. degree (cum laude) in Electrical Engineering in 2013 from Rice University, Houston, TX. He is currently a PhD student in the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN. He has been working in the Statistical Signal and Image Processing (SSIP) lab since February 2016 under the supervision of Prof. Stanley H. Chan. His current research interests include image and video restoration, large-scale optimization, image denoising, and computational photography. In addition, he also has experience in vision-based robot control, photography quality enhancement and machine learning.

Mr. Wang is a recipient of the Ross fellowship of Purdue University. He has served as a reviewer for IEEE Transactions on Image Processing. He is also granted five US patents.