Machine Learning Methods for Demosaicing and Denoising

Yu-Sheng Chen
Stanford University
Computational and Mathematical Engineering
yusheng@stanford.edu

Stephanie Sanchez
Stanford University
Computational and Mathematical Engineering
ssanche2@stanford.edu

Abstract

*We have implemented super-resolution techniques such as machine learning methods and solving a least squares problem with a prior to perform demosaicing. These techniques include k-nearest neighbors (**KNN**), **linear regression**, and alternating direction method of multipliers with a total variation prior (**ADMM TV**). For all methods, parameters were optimized to minimize the mean-squared-error (MSE) and maximize the peak signal-to-noise ratio (PSNR). The MSE and PSNR were compared to the results of traditional demosaicing methods, bilinear and Malvar interpolation.*

1. Introduction

1.1. Motivation

If we consider demosaicing and super-resolution together we notice that both methods utilize a technique to fill in pixel value information. Therefore, we have implemented a method that utilizes super-resolution techniques such as machine learning methods and solving a least squares problem to perform demosaicing.

1.2. Related Work

Traditional demosaicing methods include bilinear and Malvar interpolation. Bilinear interpolation approximates pixel values by averaging known pixel values in different directions. Malvar uses only linear kernels to perform the demosaicing by optimizing the kernel gain parameters [3].

There are many machine learning methods for image super-resolution [4], for example, k-Nearest Neighbors, Support Vector Regression and Super-Resolution Convolutional Neural Network. Machine learning techniques are used to predict the missing color/texture information. In the general image optimization problem, we model the problem as an optimization problem[5][6]. As mentioned in [5], image optimization problems contain (1) a variable which represents the target image to be reconstructed, (2) a linear operation matrix which represents the downsampling/demosaicing process, (3) a penalty measure which represents the difference of the results of downsampling/demosaicing from the measured data, and (4) the priors and constraints on the the variables. Usually we encode the prior information as a penalty (regularization) term in the objective function, to manage the ill-conditionedness of the original reconstruction problem. Once we define the constrained optimization problem, iterative update methods are applied to solve the best estimate of the higher resolution image.

2. The Method

2.1. Bayer Pattern Extraction and Noise

The first step in our procedure is to extract the bayer pattern from the image. We also note that the standard methods for demosaicing, as well as ADMM+TV, kNN, and linear regression, are also denoising methods. Therefore, we added a small amount of noise to the bayered image with parameter $\sigma = 0.01$. An example is shown in Figure 1.

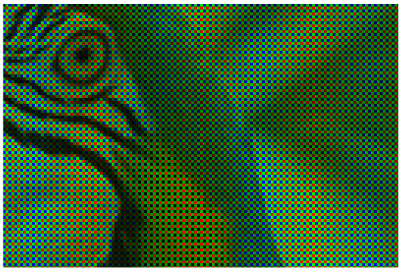

Figure 1. Bayer pattern with noise (noise parameter $\sigma = 0.01$).

2.2. Alternating Direction Method of Multipliers and Total Variation

The first method of our model solves a least squares problem to perform demosaicing. A classical approach for solving a least squares problem is to apply Alternating Direction Method of Multipliers (ADMM) with a Total Variation (TV) prior. This method promotes the usage of break-

ing apart the original problem into smaller subproblems and solving them by employing proximal operators. Figures 2 and 3 show the algorithm and proximal operators for ADMM+TV, respectively. ADMM+TV was applied to each color channel and in order to keep "matrix-free" operations, function handles for Ax and $A^T x$ were created to return the bayer pattern of the approximated color channel at each iteration.

Algorithm 1 ADMM for TV-regularized deconvolution

initialize ρ and λ
$\mathbf{x} = zeros\,(W, H)$;
$\mathbf{z} = zeros\,(W, H, 2)$;
$\mathbf{u} = zeros\,(W, H, 2)$;
for $k = 1$ **to** $maxIters$ **do**
  $\mathbf{x} = \mathbf{prox}_{\|\cdot\|_2,\rho}(\mathbf{v}) = \arg\min_{\{\mathbf{x}\}} \frac{1}{2}\|\mathbf{Ax}-\mathbf{b}\|_2^2 + \frac{\rho}{2}\|\mathbf{Kx}-\mathbf{v}\|_2^2, \quad \mathbf{v} = \mathbf{z}-\mathbf{u}$
  $\mathbf{z} = \mathbf{prox}_{\Gamma,\rho}(\mathbf{v}) = \arg\min_{\{\mathbf{z}\}} \lambda\Gamma(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{v}-\mathbf{z}\|_2^2, \quad \mathbf{v} = \mathbf{Kx}+\mathbf{u}$
  $\mathbf{u} = \mathbf{u} + \mathbf{Kx} - \mathbf{z}$
end for

Figure 2. ADMM with TV prior algorithm.

$$\underset{\{x\}}{\text{minimize}} \underbrace{\frac{1}{2}\|\mathbf{Ax}-\mathbf{b}\|_2^2}_{f(x)} + \underbrace{\lambda\Gamma(\mathbf{z})}_{g(z)}$$

subject to $\mathbf{Kx} - \mathbf{z} = 0$

$$\mathbf{x} \leftarrow \mathbf{prox}_{\|\cdot\|_2,\rho}(\mathbf{v}) = \arg\min_{\{\mathbf{x}\}} L_\rho(\mathbf{x},\mathbf{z},\mathbf{y}) = \arg\min_{\{\mathbf{x}\}} \frac{1}{2}\|\mathbf{Ax}-\mathbf{b}\|_2^2 + \frac{\rho}{2}\|\mathbf{Kx}-\mathbf{v}\|_2^2, \quad \mathbf{v} = \mathbf{z}-\mathbf{u}$$

$$\mathbf{z} \leftarrow \mathbf{prox}_{\Gamma,\rho}(\mathbf{v}) = \arg\min_{\{\mathbf{z}\}} L_\rho(\mathbf{x},\mathbf{z},\mathbf{y}) = \arg\min_{\{\mathbf{z}\}} \lambda\Gamma(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{v}-\mathbf{z}\|_2^2, \quad \mathbf{v} = \mathbf{Kx}+\mathbf{u}$$

$$\mathbf{u} \leftarrow \mathbf{u} + \mathbf{Kx} - \mathbf{z}$$

Afun = @(colorChannel, bayer, x) bayer(:, :, colorChannel).*x;

Figure 3. Proximal operators for ADMM with TV prior.

2.3. k-Nearest Neighbors

The machine learning (ML) methods can be applied to predict the missing details after the bilinear interpolation result of bayer-patterned image is complete. As shown in Fig.4, the bayer-patterned image is first split into 3 color channels, for each color channel, we use bilinear interpolation method to fill in the missing pixel values, and then the interpolation result is combined with the detail information predicted by k-Nearest Neighbors (kNN) model. The final estimated image is formed by concatenating the resulting 3 color channels into a single RGB color image. In this problem, the training sample of kNN is defined as an input/output pair consisting of the bayered image patch and its corresponding detail image patch. In the training image set, the detail image patch is obtained by the following process:

$$Detail = Source - BilinearInterp(Bayer(Source))$$

For each test bayered image patch P_i, the k closest bayered image patches are retrieved from the training set and the prediction is defined as a weighted average of the corresponding detail image patches of the k closest bayered image patch neighbors:

$$\hat{D}_i = \frac{\sum_{j=1}^{k} W_i^{(j)} D_i^{(j)}}{k}$$

The weight $W_i^{(j)}$ for the neighbor patch $N_i^{(j)}$ of the given test patch P_i is defined based on the geometric distance between P_i and $N_i^{(j)}$, as shown below:

$$W_i^{(j)} := e^{-\alpha \cdot dist(P_i, N_i^{(j)})}$$

Instead of normalizing by $\sum_{j=1}^{k} W_i^{(j)}$, we normalize by k, which yields better results. We do not normalize by the sum of weights because that as an example-based approach, detail estimation can be very inaccurate if the closest neighbors are still far from the test sample. By normalizing by k we derive a more robust model which adds detail information $D_i^{(j)}$ with distance-based confidence $W_i^{(j)}$.

For the stability concern of ML-based prediction model, we have to apply a range-limiter function on the prediction result $\hat{D}_i$ and apply another range-limiter after adding to the interpolation image, to ensure the pixel values are within valid range.

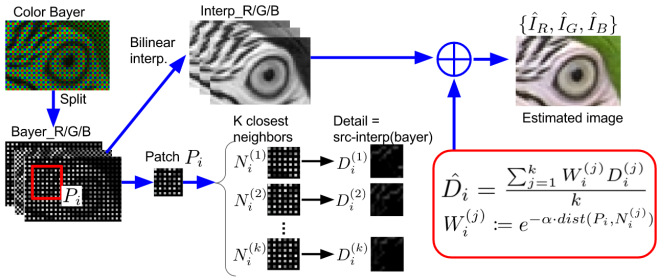

Figure 4. kNN illustration.

2.4. Linear Regression

Another detail prediction model we developed was linear regression. As shown in Fig.5, kNN detail predictor is now replaced with linear regression predictor for each color channel. Unlike kNN predicting an image patch at a time, in regression model, for each missing color pixel value, we use the linear regression predictor to predict its detail value using the local image patch centered at the pixel location as the feature input. The detailed value is defined as in kNN model. As stated in Algorithm2, linear regression algorithm optimizes the least square error over the entire training set using a linear predictor model. In our problem, according to the geometric structure of bayer pattern, 7 linear regression models are used (3 for red estimation, 3 for blue estimation, 1 for green estimation).

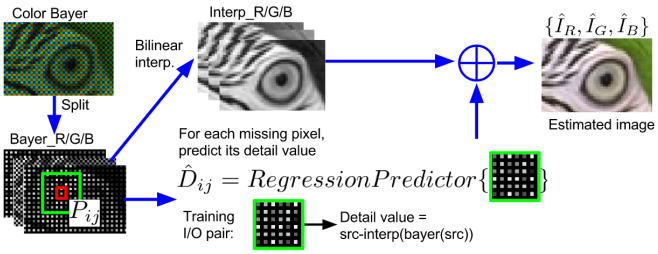

Figure 5. Regression illustration.

Data: training set $\{x^{(i)}, y^{(i)}\}$ for i=1,...,N, where N is the number of training samples, $x^{(i)} \in R^k$ and $y \in R$

Result: optimal linear predictor $h_\theta(x) := \theta^T x$

begin

minimize the cost function to obtain the optimal $\hat{\theta}$:

$$\hat{\theta} := \arg\min_\theta \frac{1}{2} \sum_{i=1}^{N} (h_\theta(x^{(i)}) - y^{(i)})^2$$

A. Iterative solution (Stochastic Gradient Descent):

1 **while** *until the cost converges* **do**

2 **while** $i = 1, ..., N$ **do**

$\theta := \theta + \alpha(y^{(i)} - h_\theta(x^{(i)}))x^{(i)}$

end

end

B. Batch solution (closed form):

$\hat{\theta} = (X^T X)^{-1} X^T Y$

where

$$X := \begin{bmatrix} \dots (x^{(1)})^T \dots \\ \dots (x^{(2)})^T \dots \\ . \\ \dots (x^{(N)})^T \dots \end{bmatrix}, Y := \begin{bmatrix} y^{(1)} \\ x^{(2)} \\ . \\ x^{(N)} \end{bmatrix}$$

end

Algorithm 1: Linear Regression Algorithm

3. Parameter Search

3.1. Parameter Optimization and Time Complexity

ADMM+TV, kNN, and linear regression are parameter methods. This means for ADMM+TV λ varies, and we vary patch sizes in kNN and regression. We optimized the parameters for the 24 images in the Kodak set so that the mean-squared error (MSE) would be minimized and the peak signal-to-noise ratio (PSNR) would be maximized.

A grid search was performed over 6 images of the Kodak set to find an optimal λ that would minimize the MSE over all 24 images. The time complexity to perform the gird search with 5 λ's over 6 images was approximately 6 hours. Even with performing grid search over 6 images, we found that the smallest λ provided the smallest MSE. Figures 6, 7, and 8 display this observation.

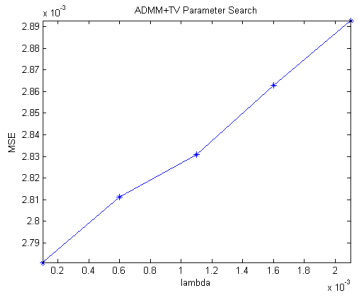

Figure 6. λ value search for Kodak image 4.

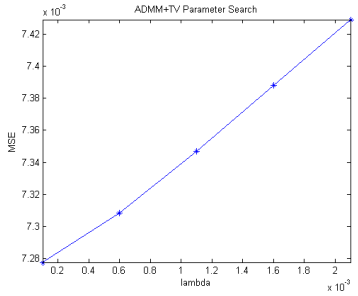

Figure 7. λ value search for Kodak image 19.

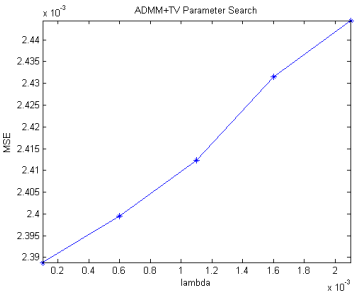

Figure 8. λ value search for Kodak image 23.

For the kNN method, the parameter k (number of closest neighbors for prediction) and image patch size are determined by minimizing the MSE. As we can see in Fig.9, as k value increases, the MSE of test set decreases, because the kNN model becomes more generalized to future data with larger k value. To the contrary, the MSE is even higher than the bilinear interpolation result when $k = 3$, which implies incorporating kNN detail prediction actually degrades the image quality. Additionally, the kNN computational complexity increases when k increases. This

results in a quality-complexity trade-off. For patch size search, we found that a smaller patch size results in a lower MSE Figure10. The reasonable explanation is that when the patch size gets smaller, the probability that there's a closely matched neighbor for a given test patch increases, therefore it yields better detail prediction for smaller patch size.

For linear regression, there's no clear relationship between the patch size and the MSE performance Figure 11, which implies pixels ranged over a distance are not correlated in this linear prediction model. To optimize the time complexity, the smallest patch size (8-by-8) is chosen.

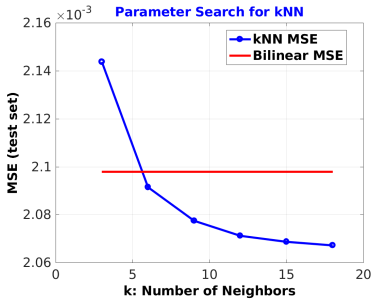

Figure 9. kNN k value search.

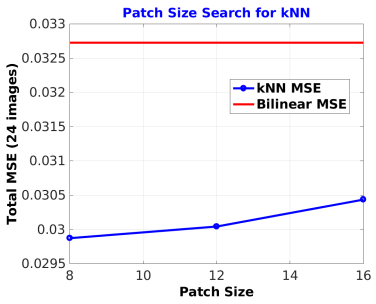

Figure 10. kNN patch size search.

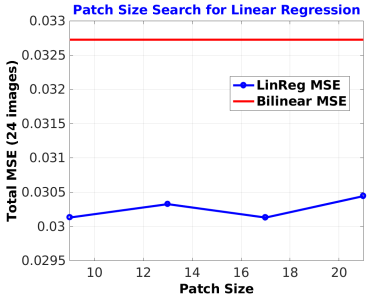

Figure 11. Regression patch size search.

4. Numerical Results

Figure 12 displays the PSNR results of our method along with bilinear and Malvar interpolation. We have observed that ADMM+TV performs best for optimizing λ for a smaller set of images or per image. We show the results for one optimized global λ for all 24 images and one λ optimized for 3 images. Table 1 shows the resulting sum of MSE across all 24 images for all methods.

Method	MSE (sum)
ADMM+TV	0.2527
k-Nearest	0.0299
Linear Regression	0.0301
Bilinear	0.0327
Malvar	0.0584

Table 1. Sum of MSE for all 24 images of for each method.

Method	training set size
k-Nearest (each model)	86016
Linear Regression (each model)	181977

Table 2. Training set size of ML methods.

Method	MSE(training set)	MSE(test set)
Bilinear	0.001216	0.002098
k-Nearest	0.001080	0.002067
Linear Regression	0.001110	0.001970

Table 3. Training/Test errors of ML methods.

Table 2 shows the amount of training data we used for kNN and linear regression methods. For the kNN model, prediction time complexity depends on the size of the training data set, while linear regression prediction just depends on feature dimension (here, it means patch size) regardless of the amount of training data. This implies our linear regression method can be efficiently applied to most portable camera devices with limited computing power.

In table 3, we see training and test errors for kNN and linear regression. Comparing with the MSE from bilinear method, we can see kNN has a slight over-fitting problem with low training error but high test error. Linear regression, on the other hand, generalizes the training data well and gives low MSE for both the training set and test set. Overall, both methods produce higher results than the simple bilinear method.

5. Conclusion

Bilinear and Malvar methods yield satisfactory results with the lowest time complexity. While ADMM+TV can be used to perform demosaicing, lambda is quite sensitive to the image content and the run time is unpractical. The kNN method (example-based) can predict the image details well, while linear regression method can also improve image quality based on bilinears result. Linear regression can be applied to most portable camera devices with limited

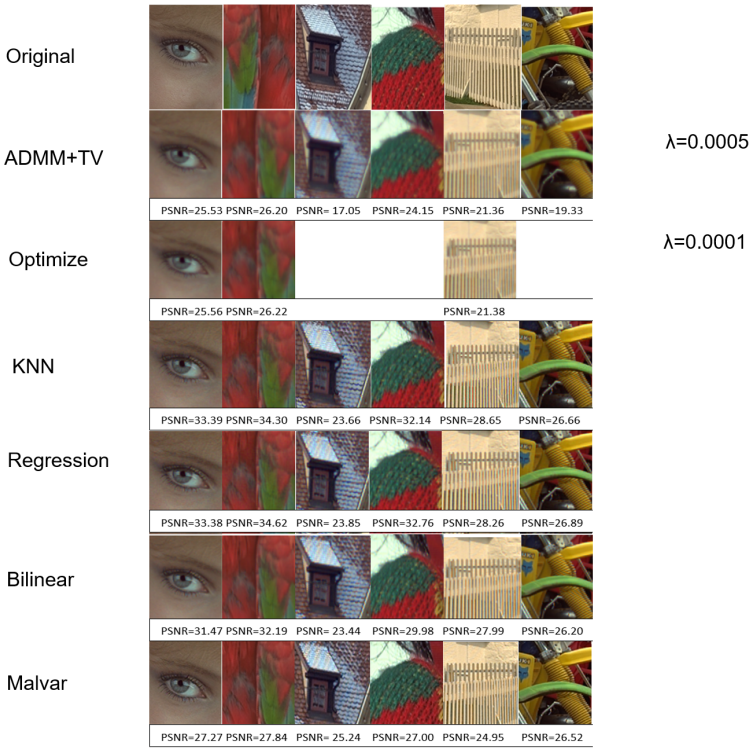

Figure 12. Method results from 6 images of Kodak data set (image 4,5,8,15,19, and 23.

computing power because of its low prediction complexity and its linearity nature. Finally, we conclude that our image detail prediction framework can incorporate most machine learning models into any image interpolation techniques to perform image demosaicing.

6. Future Work

Future work entails implementing Deep learning (CNN), another machine learning technique from super-resolution. We would also like to test the robustness of denoising in all methods, as well as use ADMM+TV for entire bayer pattern (all color channels) with other prior function such as Non-local mean prior and find the unique minimum λ that minimizes the MSE for all images.

References

[1] ACM SIGGRAPH 2001 *Image-based modeling and photo editing*.

[2] Bahadir K. Gunturk *Demosaicking: Color Filter Array Interpolation* 2014

[3] Henrique S. Malvar et al *Quality Linear Interpolation for Demosaicing of Bayer-patterned color images* ICASSP 2004

[4] Mark Sabini and Gili Rusak *Based Image Super-Resolution Techniques* CS229 course project 2016

[5] Alexey Lukin et al *Image Interpolation by Super-Resolution* International Conference Graphicon 2006

[6] Felix Heide et al *ProxImaL: Efficient Image Optimization using Proximal Algorithms* ACM SIGGRAPH 2016