Pattern-independent Demosaicing;

Presented by Daniel Khashabi
Joint work with Sebastian Nowozin, Jeremy Jancsary, Andrew W. Fitzgibbon and Bruce Lindbloom
Outline

- Demosaicing problem
- Creating input-output pairs
- Our demosaicing model
- Experiments
A simple camera pipeline

1. Scene
2. Optical operations
3. Color Filter Array (CFA)
4. Analog/Digital Conversion
5. Linear Transformations: black-level adjustment and color scaling
6. Camera output (sRGB)
7. Image correction: 3x3 color transform and Gamma correction (nonlinear)
8. Demosaicing
A simple camera pipeline

1. Scene
2. Optical operations
3. Color Filter Array (CFA)
4. Analog/Digital Conversion
5. Linear Transformations: black-level adjustment and color scaling
6. Demosaicing

Image correction: 3x3 color transform and Gamma correction (nonlinear)

Camera output (sRGB)
A simple camera pipeline

- Scene
- Optical operations
- Color Filter Array (CFA)
- Analog/Digital Conversion
- Linear Transformations: black-level adjustment and color scaling
- Demosaicing
- Image correction: 3x3 color transform and Gamma correction (nonlinear)
- Camera output (sRGB)

Diagram:
- Incoming light
- Filter layer
- Sensor array
- Resulting pattern
A simple camera pipeline

1. Scene
2. Optical operations
3. Color Filter Array (CFA)
4. Analog/Digital Conversion
5. Linear Transformations: black-level adjustment and color scaling
6. Image correction: 3x3 color transform and Gamma correction (nonlinear)
7. Demosaicing
8. Camera output (sRGB)

Diagram:
- Scene -> Optical operations -> Color Filter Array (CFA)
- Camera output -> Image correction: 3x3 color transform and Gamma correction (nonlinear) -> Demosaicing
- Demosaicing -> Analog/Digital Conversion
- Analog/Digital Conversion -> Linear Transformations: black-level adjustment and color scaling

Output Code:
- 000 -> 001 -> 010 -> 011 -> 100 -> 101 -> 110 -> 111
- Input: 0/8, 1/8, 2/8, 3/8, 4/8, 5/8, 6/8, 7/8, FS
- Output: 0, 1, 2, 3, 4, 5, 6, 7, 8
A simple camera pipeline

- Scene
- Optical operations
- Color Filter Array (CFA)
- Analog/Digital Conversion
- Linear Transformations: black-level adjustment and color scaling
- Demosaicing
- Image correction: 3×3 color transform and Gamma correction (nonlinear)
- Camera output (sRGB)

Diagram showing the process:
- Scene → Optical operations → Color Filter Array (CFA) → Analog/Digital Conversion → Linear Transformations: black-level adjustment and color scaling → Demosaicing → Image correction: 3×3 color transform and Gamma correction (nonlinear) → Camera output (sRGB)
A simple camera pipeline

Scene → Optical operations → Color Filter Array (CFA) → Analog/Digital Conversion → Linear Transformations: black-level adjustment and color scaling → Camera output (sRGB)

Image correction: 3x3 color transform and Gamma correction (nonlinear) → Demosaicing

Scene Optics

Raw Space

Mosaiced Linear Space
A simple camera pipeline

- Scene
  - Optical operations
  - Color Filter Array (CFA)
  - Analog/Digital Conversion
    - Linear Transformations: black-level adjustment and color scaling
- Camera output (sRGB)
  - Image correction: 3x3 color transform and Gamma correction (nonlinear)
  - Demosaicing

![Diagram of camera pipeline with green boxes representing demosaicing process and black and blue boxes representing linear transformations.]
A simple camera pipeline

Scene → Optical operations → Color Filter Array (CFA) → Analog/Digital Conversion → Linear Transformations: black-level adjustment and color scaling

Camera output (sRGB) → Image correction: 3x3 color transform and Gamma correction (nonlinear) → Demosaicing

Linear Transformations:

\[
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix}
= \begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{bmatrix}
\begin{bmatrix}
R \\
G \\
B
\end{bmatrix}
\]

Graph: Input value vs. Output value

Raw Space → Mosaiced Linear Space → Demosaiced Linear Space → sRGB Space
A simple camera pipeline

Scene → Optical operations → Color Filter Array (CFA) → Analog/Digital Conversion

Image correction: 3x3 color transform and Gamma correction (nonlinear) → Demosaicing → Linear Transformations: black-level adjustment and color scaling

Camera output (sRGB)
A simple camera pipeline

1. Scene
2. Optical operations
3. Color Filter Array (CFA)
4. Analog/Digital Conversion
5. Image correction: 3x3 color transform and Gamma correction (nonlinear)
6. Demosaicing
7. Linear Transformations: black-level adjustment and color scaling

Demosaicing

- Interpolating color-filter-array (CFA) samples to create full-resolution color images

Image from: [http://i1.cbsi.com/cnwk1d/i/tim/2012/02/06/Adobe-r...o-demosaic-diagram.jpg](http://i1.cbsi.com/cnwk1d/i/tim/2012/02/06/Adobe-r...o-demosaic-diagram.jpg)
Demosaicing

- Interpolating color-filter-array (CFA) samples to create full-resolution color images
- Isn’t a solved problem?! NO!

Image from: http://i.cbsi.com.cnwk.1d/i/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg
Demosaicing

- Interpolating color-filter-array (CFA) samples to create full-resolution color images
- Isn’t a solved problem?! NO!
- The current methods are can be improved in terms of performance
  - Zippering effect still present in the output of many current method

Image from: http://i.i.cbsi.com/cnwk.1d/i/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg
Demosaicing

- Interpolating color-filter-array (CFA) samples to create full-resolution color images
- Isn’t a solved problem?! NO!
- The current methods are can be improved in terms of performance
  - Zippering effect still present in the output of many current method

Image from: http://i.i.cbsi.com/cnwk.1d/ititim/2012/02/06/Adobe-raw-demosaic-diagram.jpg
Demosaicing

- Interpolating color-filter-array (CFA) samples to create full-resolution color images
- Isn’t a solved problem?! **NO!**
- The current methods are can be improved in terms of performance
  - Zippering effect still present in the output of many current method
- The current methods are CFA-specific (mostly Bayer pattern)
And not easily generalizable to new CFAs

Image from: [http://i.i.cbsi.com/cnwk.1d/i/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg](http://i.i.cbsi.com/cnwk.1d/i/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg)
Demosaicing

- Interpolating color-filter-array (CFA) samples to create full-resolution color images
- Isn’t a solved problem?! NO!
- The current methods are can be improved in terms of performance
  - Zippering effect still present in the output of many current method
- The current methods are CFA-specific (mostly Bayer pattern)

And not easily generalizable to new CFAs

Image from: http://i.i.cbsi.com/cnwk.1d/i/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg
Demosaicing

- Interpolating color-filter-array (CFA) samples to create full-resolution color images
- Isn’t a solved problem?! NO!

- The current methods are can be improved in terms of performance
  - Zippering effect still present in the output of many current method
- The current methods are CFA-specific (mostly Bayer pattern)

And not easily generalizable to new CFAs

Image from: [http://I.cbsi.com/cnwk.1d/i/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg](http://I.cbsi.com/cnwk.1d/i/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg)
Demosaicing

- Interpolating color-filter-array (CFA) samples to create full-resolution color images
- Isn’t a solved problem?! NO!
- The current methods are can be improved in terms of performance
  - Zippering effect still present in the output of many current method
- The current methods are CFA-specific (mostly Bayer pattern)
- And not easily generalizable to new CFAs
- All method are engineered for sRGB images
  - Demosaicing is lacking a good dataset

Image from: [http://i.i.cbsi.com/cnwk.1d/i/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg](http://i.i.cbsi.com/cnwk.1d/i/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg)
Demosaicing

- Interpolating color-filter-array (CFA) samples to create full-resolution color images
- Isn’t a solved problem?! **NO!**
- The current methods are can be improved in terms of performance
  - Zippering effect still present in the output of many current method
- The current methods are CFA-specific (mostly Bayer pattern)
  And not easily generalizable to new CFAs
- All method are engineered for sRGB images
  - Demosaicing is lacking a good dataset

Image from: http://i.i.cbsi.com/cnwk.1d/i/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg
Demosaicing

- Interpolating color-filter-array (CFA) samples to create full-resolution color images
- Isn’t a solved problem? NO!
- The current methods are can be improved in terms of performance
  - Zippering effect still present in the output of many current method
- The current methods are CFA-specific (mostly Bayer pattern)
  - And not easily generalizable to new CFAs
- All method are engineered for sRGB images
  - Demosaicing is lacking a good dataset
- Many of low-level task could be done before/joint with demosaicing
  - Noise behaviour changes after demosaicing

Image from: http://iti.cbsi.com.cnwk.1d/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg
Overview of our approach
Overview of our approach

- We design a *supervised model*. 
Overview of our approach

- We design a **supervised model**.
- For training the model we propose a procedure to create the ground truth dataset from **light-space** images
Overview of our approach

- We design a **supervised model**.
- For training the model we propose a procedure to create the ground dataset from **light-space** images
Overview of our approach

- We design a **supervised model**.
- For training the model we propose a procedure to create the ground truth dataset from **light-space** images
  - Our dataset is will be published with our work.
Overview of our approach

- We design a **supervised model**.
- For training the model we propose a procedure to create the ground truth dataset from **light-space** images
  - Our dataset is will be published with our work.
- Our model is **easily generalizable** to different CFA patterns.
Overview of our approach

- We design a **supervised model**.
- For training the model we propose a procedure to create the ground truth dataset from **light-space** images
  - Our dataset is will be published with our work.
- Our model is **easily generalizable** to different CFA patterns.
- Our model can perform **denoising jointly with demosaicing**.
Creating ground-truth images
Creating ground-truth images

- We want to design set of **input-output** pairs in **linear light-space**
Creating ground-truth images

- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant
  - regardless of some properties like noise distribution
Creating ground-truth images

- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant
  - regardless of some properties like noise distribution
Creating ground-truth images

- We want to design a set of input-output pairs in linear light-space.
- We use the fact that images are scale-invariant regardless of some properties like noise distribution.
Creating ground-truth images

- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant
  - regardless of some properties like noise distribution
Creating ground-truth images

- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant regardless of some properties like noise distribution
Creating ground-truth images

- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant
  - regardless of some properties like noise distribution
Creating ground-truth images

- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant
  - regardless of some properties like noise distribution
Creating ground-truth images

- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant
  - regardless of some properties like noise distribution
Creating ground-truth images

- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant
  - regardless of some properties like noise distribution
down-scaling strategies

- There is a systematic bias between red and blue, when the size of the square is **even**
down-scaling strategies

- There is a systematic bias between red and blue, when the size of the square is even
down-scaling strategies

- There is a systematic bias between red and blue, when the size of the square is **even**
down-scaling strategies

- There is a systematic bias between red and blue, when the size of the square is even.

- When the size of the block is odd, this systematic bias vanishes, but the ratio of color sensors are different. This ratio goes to one, when the size of the block increases.
down-scaling strategies

- There is a systematic bias between red and blue, when the size of the square is **even**

- When the size of the block is **odd**, this systematic bias vanishes, but the ratio of color sensors are different
  - This ratio goes to **one**, when the size of the block increases
down-scaling strategies

- There is a systematic bias between red and blue, when the size of the square is even.

- When the size of the block is odd, this systematic bias vanishes, but the ratio of color sensors are different.
  - This ratio goes to one, when the size of the block increases.
down-scaling strategies

- There is a systematic bias between red and blue, when the size of the square is **even**

- When the size of the block is **odd**, this systematic bias vanishes, but the ratio of color sensors are different
  - This ratio goes to **one**, when the size of the block increases

- We devised a strategy to compensate for the systematic bias when the size of the window is even
Max-Ent down-scaling

- **Goal:**
  - Want to bring the **center of the mass**, for each color, to the center of each block
Max-Ent down-scaling

Goal:
- Want to bring the center of the mass, for each color, to the center of each block
Max-Ent down-scaling

- Goal:
  - Want to bring the center of the mass, for each color, to the center of each block

\[
\begin{align*}
\max_p \{ H(p) &= \sum_{x,y} p(x,y) \log p(x,y) \} \\
\sum_{x,y} p(x,y) &= 1 \\
p(x,y) &\geq 0, \forall x, y \\
\sum_{x,y} x p(x,y) &= W + 0.5, \forall y \\
\sum_{x,y} y p(x,y) &= W + 0.5, \forall x
\end{align*}
\]
**Max-Ent down-scaling**

- **Goal:**
  - Want to bring the center of the mass, for each color, to the center of each block.

\[
\max_p \{ \mathcal{H}(p) = \sum_{x,y} p(x,y) \log p(x,y) \}
\]

\[
\begin{align*}
\sum_{x,y} p(x,y) &= 1 \\
p(x,y) &\geq 0, \forall x, y \\
\sum_{x,y} x p(x,y) &= W + 0.5, \forall y \\
\sum_{x,y} y p(x,y) &= W + 0.5, \forall x
\end{align*}
\]

\[
p_r(x,y) = \frac{1}{16} \begin{bmatrix}
1 & 0 & 3 & 0 \\
0 & 0 & 0 & 0 \\
3 & 0 & 9 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}, \quad
p_b(x,y) = \frac{1}{16} \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 9 & 0 & 3 \\
0 & 0 & 0 & 0 \\
0 & 3 & 0 & 1
\end{bmatrix}, \quad
p_g(x,y) = \frac{1}{8} \begin{bmatrix}
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0
\end{bmatrix}
\]
Max-Ent down-scaling

Goal:

- Want to bring the center of the mass, for each color, to the center of each block

\[
\begin{align*}
\max_p \{ H(p) &= \sum_{x,y} p(x,y) \log p(x,y) \} \\
\sum_{x,y} p(x,y) &= 1 \\
p(x,y) &\geq 0, \forall x, y \\
\sum_{x,y} xp(x,y) &= W + 0.5, \forall y \\
\sum_{x,y} yp(x,y) &= W + 0.5, \forall x
\end{align*}
\]

\[
p_r(x, y) = \frac{1}{16} \begin{bmatrix} 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 3 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad p_b(x, y) = \frac{1}{16} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 3 \\ 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 1 \end{bmatrix}, \quad p_y(x, y) = \frac{1}{8} \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}
\]
Down-scaling, a real example
Down-scaling, a real example

<table>
<thead>
<tr>
<th>Uniform Averaging</th>
<th>MaxEnt (Even Block) Averaging</th>
<th>Odd Block Averaging</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Uniform Averaging" /></td>
<td><img src="image2" alt="MaxEnt (Even Block) Averaging" /></td>
<td><img src="image3" alt="Odd Block Averaging" /></td>
</tr>
</tbody>
</table>
Down-scaling, a real example
Down-scaling, a real example
Down-scaling, a real example

<table>
<thead>
<tr>
<th>Uniform Averaging</th>
<th>MaxEnt (Even Block) Averaging</th>
<th>Odd Block Averaging</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Uniform Averaging" /></td>
<td><img src="image2" alt="MaxEnt (Even Block) Averaging" /></td>
<td><img src="image3" alt="Odd Block Averaging" /></td>
</tr>
</tbody>
</table>
Down-scaling, a real example
Creating ground-truth images
Creating ground-truth images

- Collected 500 raw images captured by Panasonic-Lumix LX-3
- We need to be able to go along a good camera pipeline
Creating ground-truth images

- Collected 500 raw images captured by Panasonic-Lumix LX-3
- We need to be able to go along a good camera pipeline
- We use dcraw to transfer images to different stages in camera pipeline
Creating ground-truth images

- Collected 500 raw images captured by Panasonic-Lumix LX-3
- We need to be able to go along a good camera pipeline
- We use `dcraw` to transfer images to different stages in camera pipeline

- We write out the **specifications of each step**, for each image

```
Raw images → Linear Transformations: black-level adjustment and color scaling → Demosaicing → Image correction: 3x3 color transform and Gamma correction (nonlinear) → Camera output (sRGB)
```
Creating ground-truth images

- Collected 500 raw images captured by Panasonic-Lumix LX-3
- We need to be able to go along a good camera pipeline
- We use `dcraw` to transfer images to different stages in camera pipeline

We write out the **specifications of each step**, for each image

```
Raw images → Linear Transformations: black-level adjustment and color scaling → Demosaicing → Image correction: 3x3 color transform and Gamma correction (nonlinear) → Camera output (sRGB)
```
Creating ground-truth images

- Collected 500 raw images captured by Panasonic-Lumix LX-3
- We need to be able to go along a good camera pipeline
- We use `dcraw` to transfer images to different stages in camera pipeline
- We write out the **specifications of each step**, for each image
- Then simulate our own pipeline in MATLAB, with **demosaicing** replaced with our own **down-scaling**
Creating ground-truth images

- Collected 500 raw images captured by Panasonic-Lumix LX-3
- We need to be able to go along a good camera pipeline
- We use `dcraw` to transfer images to different stages in camera pipeline
- We write out the specifications of each step, for each image
- Then simulate our own pipeline in MATLAB, with demosaicing replaced with our own down-scaling
Creating ground-truth images

- Collected 500 raw images captured by Panasonic-Lumix LX-3
- We need to be able to go along a good camera pipeline
- We use `dcraw` to transfer images to different stages in camera pipeline
- We write out the **specifications of each step**, for each image
- Then simulate our own pipeline in MATLAB, with demosaicing replaced with our own down-scaling

```
Raw images → Linear Transformations: black-level adjustment and color scaling → Demosaicing → Image correction: 3x3 color transform and Gamma correction (nonlinear) → Camera output (sRGB)
```

```
Raw images → Linear Transformations: black-level adjustment and color scaling → Down-scaling → Image correction: 3x3 color transform and Gamma correction (nonlinear) → Camera output (sRGB)
```
Creating ground-truth images

- Collected 500 raw images captured by Panasonic-Lumix LX-3
- We need to be able to go along a good camera pipeline
- We use `dcraw` to transfer images to different stages in camera pipeline
- We write out the specifications of each step, for each image
- Then simulate our own pipeline in MATLAB, with demosaicing replaced with our own down-scaling.
The noise behaviour in down-scaled images

- Averaging decreases the noise present in images.
The noise behaviour in down-scaled images

- Averaging decreases the noise present in images.
- Need to devise a way to bring back the noise into our images
  - Since the goal is to perform demosaicing on the original raw images
The noise behaviour in down-scaled images

- Averaging decreases the noise present in images.
- Need to devise a way to bring back the noise into our images
  - Since the goal is to perform demosaicing on the original raw images
- We use a recent method of noise estimation for raw images [1]
  - Models as Poisson-Gaussian distribution

\[
\begin{align*}
  z(x) &= y(x) + \eta_p(y(x)) + \eta_g(x) \\
  \chi(y(x) + \eta_p(y(x))) &\sim \mathcal{P}(\chi y(x)), \quad \eta_g(x) \sim \mathcal{N}(0, b),
\end{align*}
\]

The noise behaviour in down-scaled images

- Averaging decreases the noise present in images.
- Need to devise a way to bring back the noise into our images
  - Since the goal is to perform demosaicing on the original raw images
- We use a recent method of noise estimation for raw images [1]
  - Models as Poisson-Gaussian distribution

\[
\begin{align*}
z(x) &= y(x) + \eta_p(y(x)) + \eta_g(x) \\
\chi(y(x) + \eta_p(y(x))) &\sim \mathcal{P}(\chi y(x)), \quad \eta_g(x) \sim \mathcal{N}(0, b),
\end{align*}
\]

- We then add back the noise into images

The noise behaviour in down-scaled images

- Averaging decreases the noise present in images.
- Need to devise a way to bring back the noise into our images
  - Since the goal is to perform demosaicing on the original raw images
- We use a recent method of noise estimation for raw images [1]
  - Models as Poisson-Gaussian distribution

\[
\begin{align*}
    z(x) &= y(x) + \eta_p(y(x)) + \eta_g(x) \\
    \chi(y(x) + \eta_p(y(x))) &\sim P(\chi y(x)), \quad \eta_g(x) \sim N(0, b),
\end{align*}
\]

- We then add back the noise into images

Samples of noise-addition to image
Samples of noise-addition to image

<table>
<thead>
<tr>
<th>Linear(noise-free)</th>
<th>Linear (noisy)</th>
<th>sRGB (noise-free)</th>
<th>sRGB (Noisy)</th>
</tr>
</thead>
</table>

![Images of samples of noise-addition to image](image-url)
Samples of noise-addition to image

<table>
<thead>
<tr>
<th>Linear(noise-free)</th>
<th>Linear (noisy)</th>
<th>sRGB (noise-free)</th>
<th>sRGB (Noisy)</th>
</tr>
</thead>
</table>

---

Sample images showing the effects of noise addition to images.
Samples of noise-addition to image

<table>
<thead>
<tr>
<th>Linear (noise-free)</th>
<th>Linear (noisy)</th>
<th>sRGB (noise-free)</th>
<th>sRGB (Noisy)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Possible tasks

- We have developed these images

- Among the above tasks we are mostly interested in doing the followings:
  - Demosaicing in linear light-space
  - Deniosing + demosaicing in linear light-space
Possible tasks

- We have developed these images

- Among the above tasks we are mostly interested in doing the followings:
  - Demosaicing in linear light-space
  - Deniosing + demosaicing in linear light-space
Possible tasks

- We have developed these images

- Among the above tasks we are mostly interested in doing the followings:
  - Demosaicing in linear light-space
  - Denoising + demosaicing in linear light-space
Possible tasks

- We have developed these images

- Among the above tasks we are mostly interested in doing the followings:
  - Demosaicing in linear light-space
  - Denoising + demosaicing in linear light-space
Possible tasks

- We have developed these images

- Among the above tasks we are mostly interested in doing the followings:
  - Demosaicing in linear light-space
  - Denoisings + demosaicing in linear light-space
Possible tasks

- We have developed these images

- Among the above tasks we are mostly interested in doing the followings:
  - Demosaicing in linear light-space
  - Denoising + demosaicing in linear light-space
**Possible tasks**

- We have developed these images

- Among the above tasks we are mostly interested in doing the followings:
  - Demosaicing in linear light-space
  - Denosing + demosaicing in linear light-space
Regression tree field

- Combines regression trees (Non-parametric) and Gaussian Random Fields
Regression tree field

- Combines **regression trees (Non-parametric)** and **Gaussian Random Fields**

- Energy defined for each factor:
  \[ E(y_F | x_F) = \frac{1}{2} y_F^T Q(x_F) y_F - y^T L(x_F) b(x_F) \]

  Measure of the goodness of particular labelling given inputs, and parameters of the factor
Regression tree field

- Combines regression trees (Non-parametric) and Gaussian Random Fields

- Energy defined for each factor:
  - Measure of the goodness of particular labelling given inputs, and parameters of the factor
  - Coefficients of are determined by the regression tree, or each leave stores set of parameters.

\[ E(y_F \mid x_F) = \frac{1}{2} y_F^T Q(x_F) y_F - y_F^T L(x_F) b(x_F) \]
Regression tree field

- Combines regression trees (Non-parametric) and Gaussian Random Fields

- Energy defined for each factor:

  \[ E(y_F \mid x_F) = \frac{1}{2} y_F^T Q(x_F) y_F - y^T L(x_F) b(x_F) \]

  - Measure of the goodness of particular labelling given inputs, and parameters of the factor
  - Coefficients of are determined by the regression tree, or each leaf stores set of parameters.

- Inference:

  \[ p(y \mid x; w) \propto \exp[-E(y \mid x; w)] \quad \Rightarrow \quad \hat{y}(x) = \arg\max_y p(y \mid x) = \mu = [Q(x; w)]^{-1}l(x; w) \]
Regression tree field

- Combines regression trees (Non-parametric) and Gaussian Random Fields

- Energy defined for each factor:
  - Measure of the goodness of particular labelling given inputs, and parameters of the factor
  - Coefficients of are determined by the regression tree, or each leave stores set of parameters.

- Inference:
- Training: Jointly choosing the structure of the tree, and parameters of at leaves such that minimizes the empirical risk, in greedy way:

\[
p(y \mid x; w) \propto \exp[-E(y \mid x; w)] \quad \Rightarrow \quad \hat{y}(x) = \arg\max_y p(y \mid x) = \mu = [Q(x; w)]^{-1}l(x; w)
\]

\[
\frac{1}{N} \sum_{i}^{N} \ell(\hat{y}(x^{(i)}; w), y^{(i)}) \approx \mathbb{E}_{p(x, y)} [\ell(\hat{y}(x; w), y)]
\]
Regression tree fields, for demosaicing

Regression tree fields, for demosaicing

- **Tree Feature checks**,  
  - A preliminary bilinear interpolation

Regression tree fields, for demosaicing

- **Tree Feature checks**,  
  - A preliminary bilinear interpolation  
  - RFS filters [1] which act like derivatives in different directions, with various scales

Regression tree fields, for demosaicing

- **Tree Feature checks,**
  - A preliminary bilinear interpolation
  - RFS filters [1] which act like derivatives in different directions, with various scales
- Quadratic energy basis vectors,
  - Set of neighbouring pixels,

Regression tree fields, for demosaicing

- **Tree Feature checks**,  
  - A preliminary bilinear interpolation  
  - RFS filters [1] which act like derivatives in different directions, with various scales  
- Quadratic energy basis vectors,  
  - Set of neighbouring pixels,  
  - RFS filter responses

\[
E(y_F \mid x_F) = \frac{1}{2} y_F^T Q(x_F) y_F - y^T L(x_F) b(x_F)
\]  

Regression tree fields, for demosaicing

Regression tree fields, for demosaicing

- The generalized loss function,

Regression tree fields, for demosaicing

- The generalized loss function,
- An approximate camera pipeline, from linear-space, to sRGB space, which is differentiable
  - An analytic approximation for gamma transform

Regression tree fields, for demosaicing

- The generalized loss function,
  - An approximate camera pipeline, from linear-space, to sRGB space, which is differentiable
    - An analytic approximation for gamma transform
    - An approximate 3x3 color transform,

Regression tree fields, for demosaicing

- The generalized loss function,
  - An approximate camera pipeline, from linear-space, to sRGB space, which is differentiable
    - An analytic approximation for gamma transform
    - An approximate $3 \times 3$ color transform,

$$c_{\text{MSE}}(\mathbf{I}, \mathbf{\bar{I}}) = \sum_{i \in \{R, G, B\}} \sum_{x, y} (I_{x, y}^i - \bar{I}_{x, y}^i)^2.$$
Regression tree fields, for demosaicing

- The generalized loss function,
  - An approximate camera pipeline, from linear-space, to sRGB space, which is differentiable
    - An analytic approximation for gamma transform
    - An approximate $3 \times 3$ color transform,

\[
c_{\text{MSE}}(I, \hat{I}) = \sum_{i=\{R,G,B\}} \sum_{x,y} (I^i_{x,y} - \hat{I}^i_{x,y})^2.
\]

\[
f : L \rightarrow I \Rightarrow c_{\text{G-MSE}}(L, \hat{L}) = (c_{\text{MSE}} \circ f)(L, \hat{L}) = \sum_{i=\{R,G,B\}} \sum_{x,y} (f(L^i_{x,y}) - f(\hat{L}^i_{x,y}))^2
\]

Regression tree fields, for demosaicing

- The generalized loss function,
  - An approximate camera pipeline, from linear-space, to sRGB space, which is differentiable
    - An analytic approximation for gamma transform
    - An approximate $3 \times 3$ color transform,

\[ c_{\text{MSE}}(I, \tilde{I}) = \sum_{i=\{R,G,B\}} \sum_{x,y} (I_{x,y}^i - \tilde{I}_{x,y}^i)^2. \]

\[ f : \mathbf{L} \rightarrow \mathbf{I} \implies c_{\text{G-MSE}}(\mathbf{L}, \tilde{\mathbf{L}}) = (c_{\text{MSE}} \circ f)(\mathbf{L}, \tilde{\mathbf{L}}) = \sum_{i=\{R,G,B\}} \sum_{x,y} (f(L_{x,y}^i) - f(\tilde{L}_{x,y}^i))^2 \]

- Stacking RTFs,

Experiments (1)

PSNR (measured in linear space)

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bilinear</td>
<td>29</td>
</tr>
<tr>
<td>MATLAB</td>
<td>30</td>
</tr>
<tr>
<td>HA</td>
<td>31</td>
</tr>
<tr>
<td>AHD</td>
<td>32</td>
</tr>
<tr>
<td>nat</td>
<td>33</td>
</tr>
<tr>
<td>PSF</td>
<td>34</td>
</tr>
<tr>
<td>POCs</td>
<td>35</td>
</tr>
<tr>
<td>nlm</td>
<td>36</td>
</tr>
<tr>
<td>AP</td>
<td>37</td>
</tr>
<tr>
<td>oneStepAp</td>
<td>38</td>
</tr>
<tr>
<td>WECD</td>
<td>39</td>
</tr>
<tr>
<td>directionalMMSE</td>
<td>40</td>
</tr>
</tbody>
</table>
Experiments(1)

- Demosaicing, in linear-space, without noise:
Experiments(2)

PSNR (measured in sRGB)

- Bilinear
- MATLAB
- HA
- PSF
- AP
- POCUS
- oneStepAp
- AHD
- nat
- WECD
- directionalMMSE
- nlm
- RTF-b
- LASIP
- RTF-a
- CS

Values:
- 24
- 25
- 26
- 27
- 28
- 29
- 30
- 31
- 32
- 33
- 34
- 35
Experiments(2)

- Demosaicing, in linear-space, without noise:
Some samples

NAT: Non-Local Adaptive Thresholding:
NLM: Non-Local Means
Some samples

Ground truth  Bilinear  CS  RTF

NAT: Non-Local Adaptive Thresholding:
NLM: Non-Local Means
Some samples
Some samples

NAT: Non-Local Adaptive Thresholding:
NLM: Non-Local Means
Some samples

Ground truth  Bilinear  CS  RTF

NAT: Non-Local Adaptive Thresholding:
NLM: Non-Local Means
Image Margins
Image Margins

- Not a considerable margin in RTF outputs

NAT: Non-Local Adaptive Thresholding
NLM: Non-Local Means
Image Margins

- Not a considerable margin in RTF outputs

NAT: Non-Local Adaptive Thresholding
NLM: Non-Local Means
And more to go ....
And more to go ....

- RTF with the generalized loss function
And more to go ....

- RTF with the generalized loss function
- Full denoising-demosaicing experiments
And more to go ....

- RTF with the generalized loss function
- Full denoising-demosaicing experiments
- Other CFA patterns: Fuji-Xtrans pattern
And more to go ....

- RTF with the generalized loss function
- Full denoising-demosaicing experiments
- Other CFA patterns: Fuji-Xtrans pattern
- Further work?!