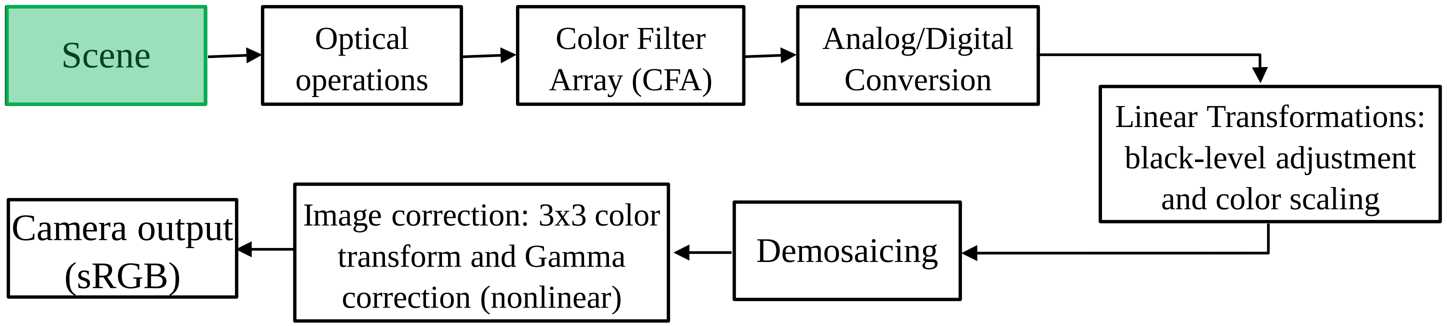
Pattern-independent Demosaicing;

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

Outline

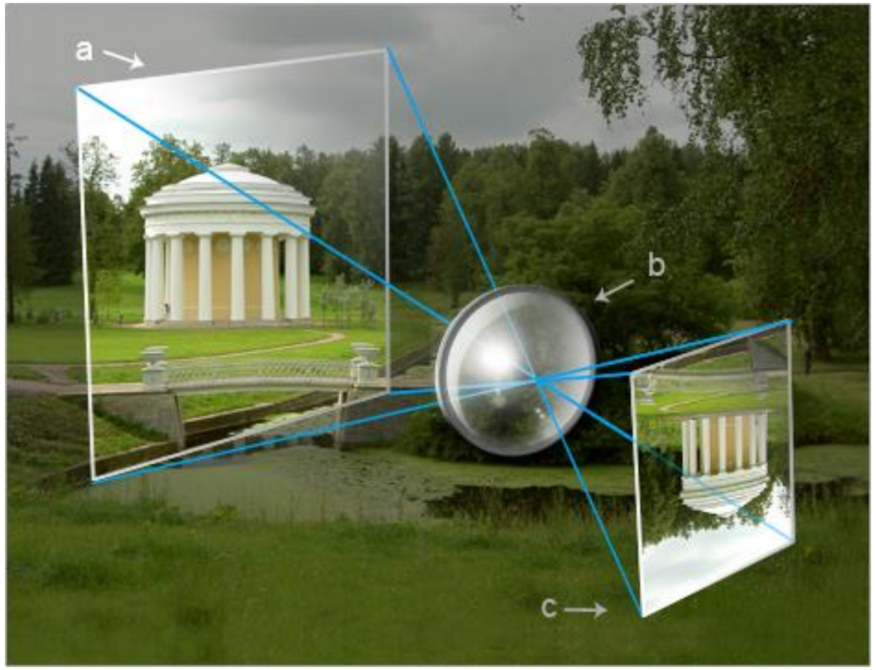
- Demosaicing problem
- Creating input-output pairs
- Our demosaicing model
- Experiments

A simple camera pipeline

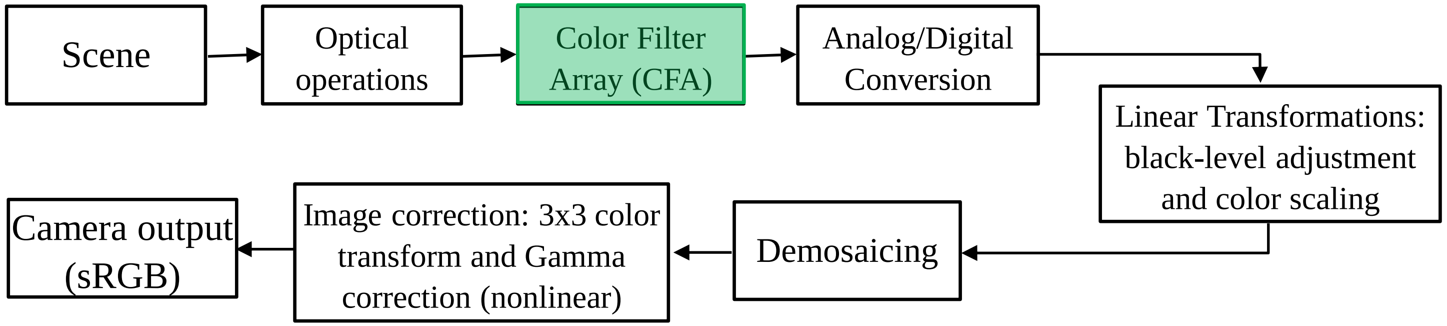


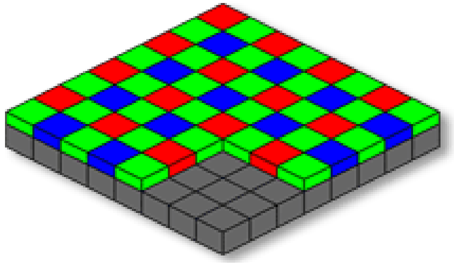
A simple camera pipeline

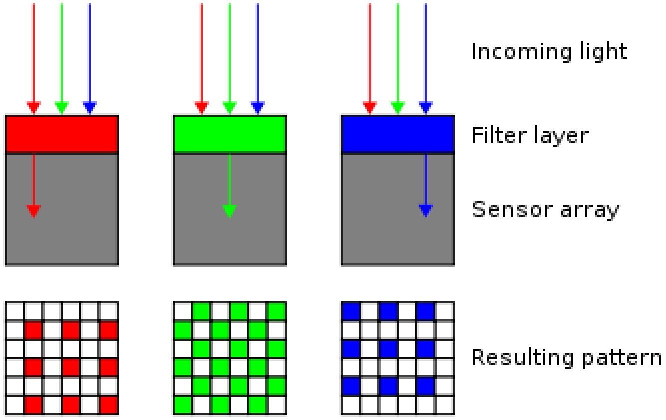
Scene → Optical operations → Color Filter Array (CFA) → Analog/Digital Conversion → Linear Transformations: black-level adjustment and color scaling → Demosaicing → Image correction: 3x3 color transform and Gamma correction (nonlinear) → Camera output (sRGB)



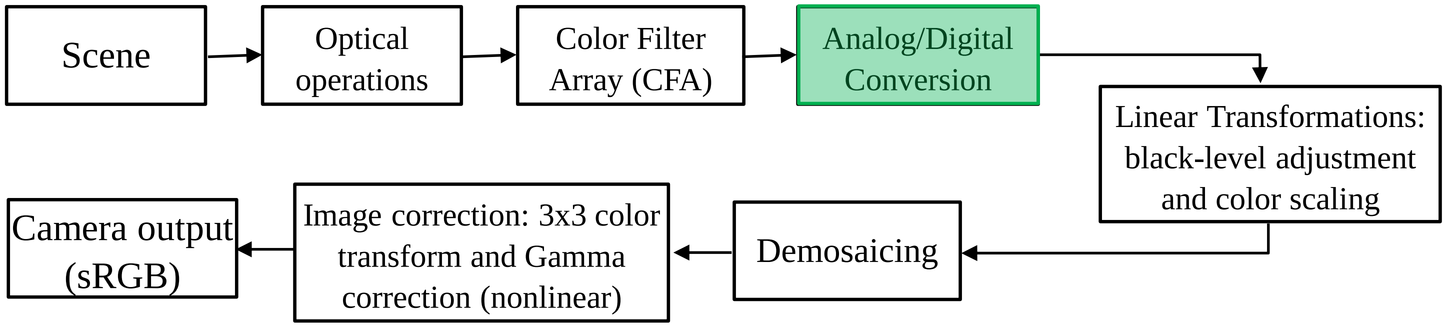
A simple camera pipeline

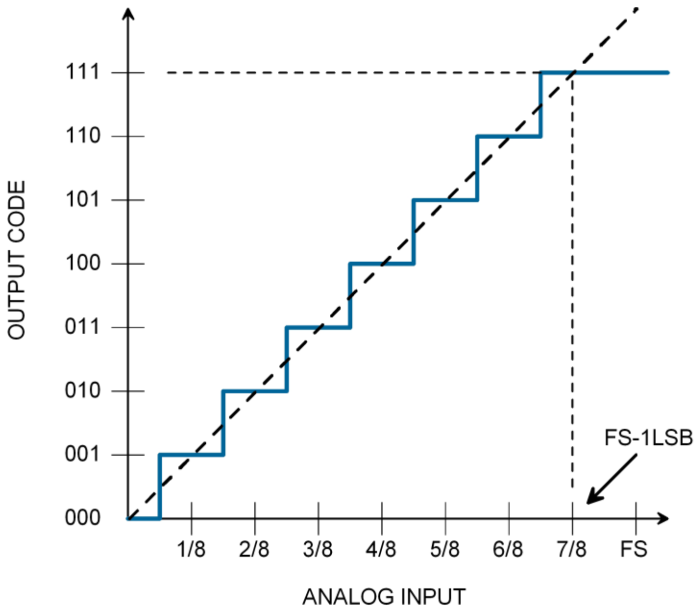




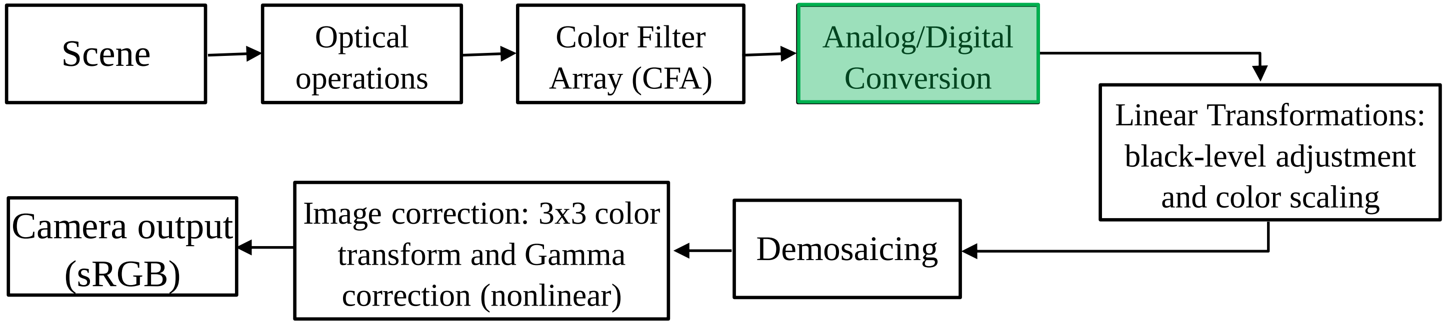


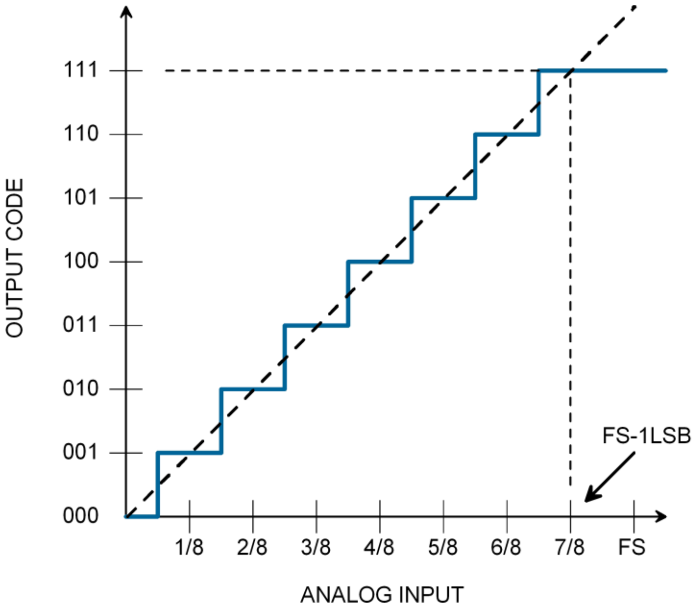
A simple camera pipeline

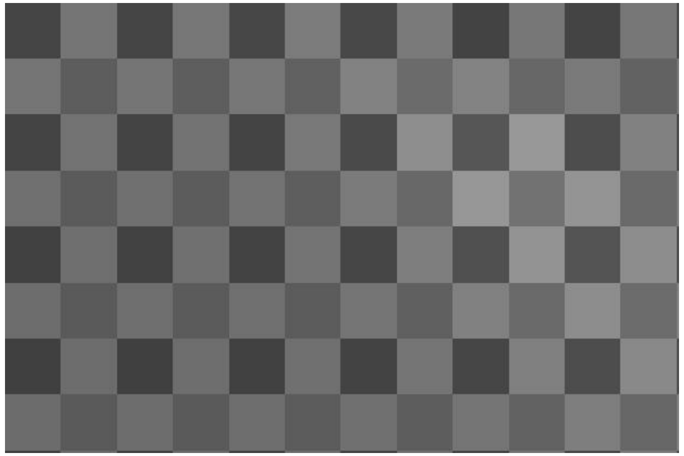




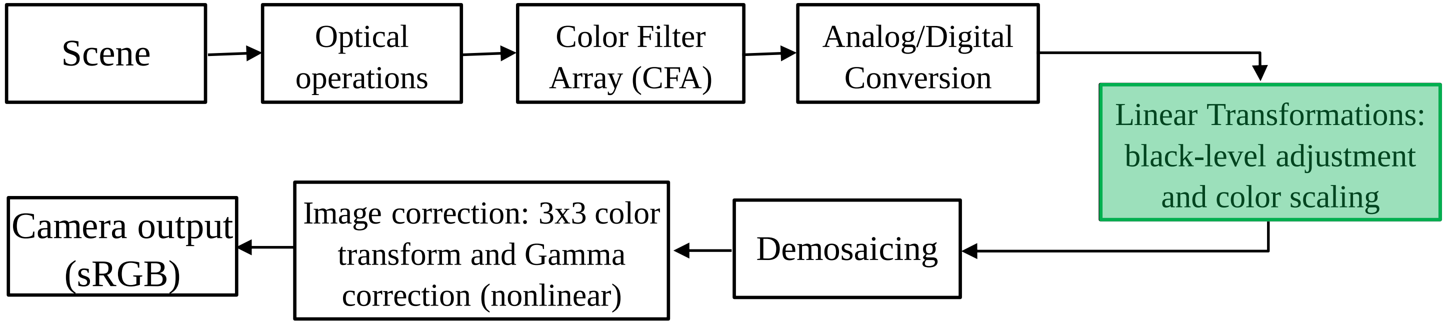
A simple camera pipeline

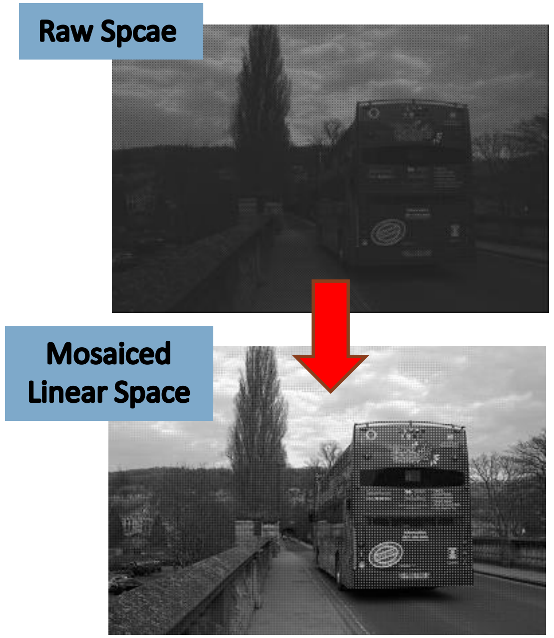




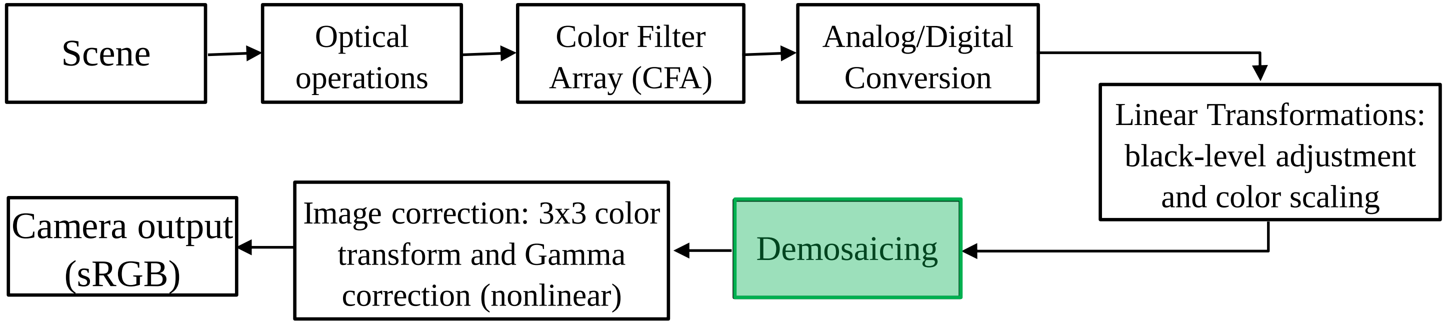


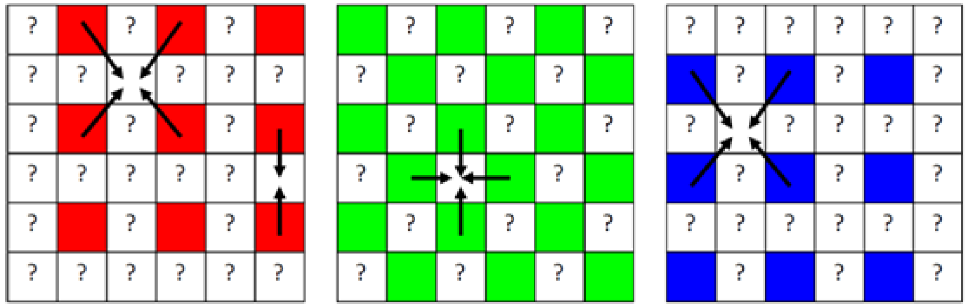
A simple camera pipeline

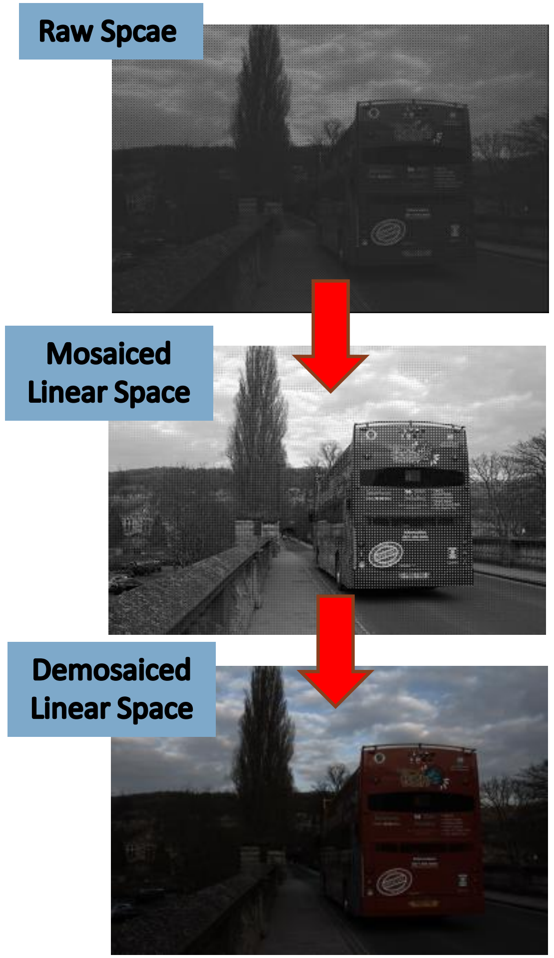




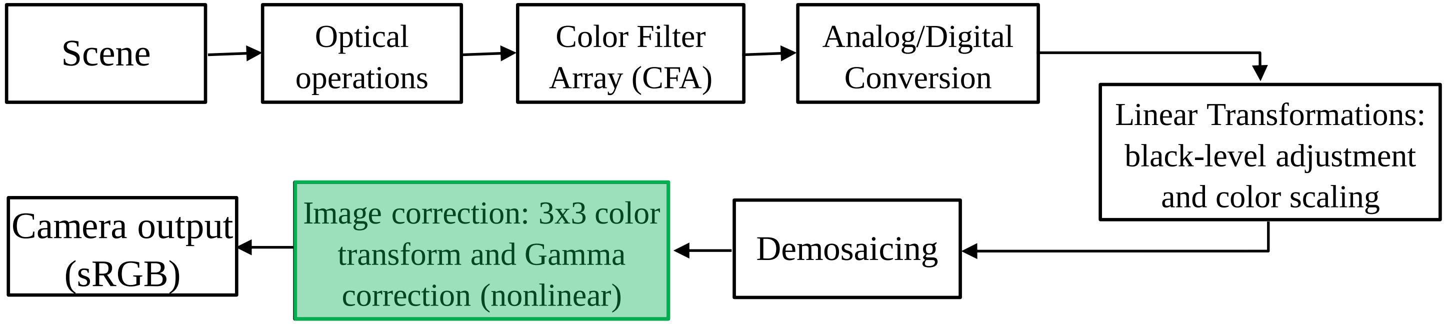
A simple camera pipeline

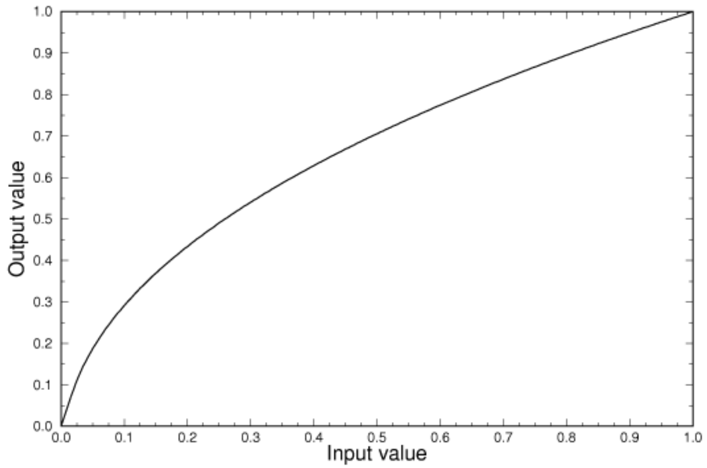




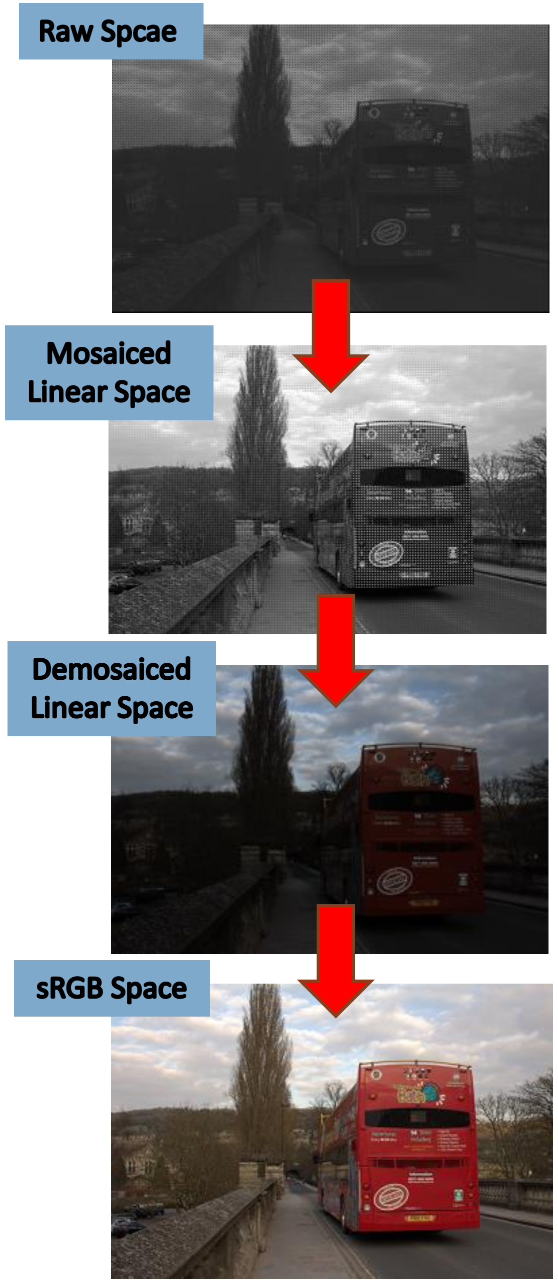


A simple camera pipeline

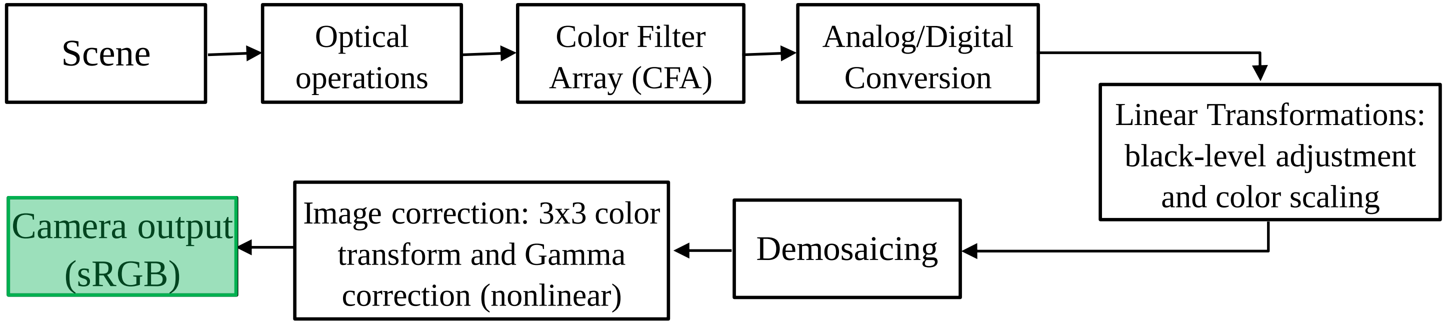


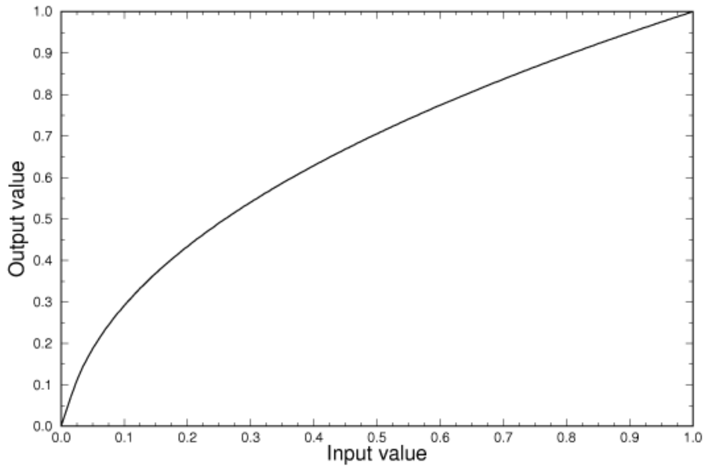


$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

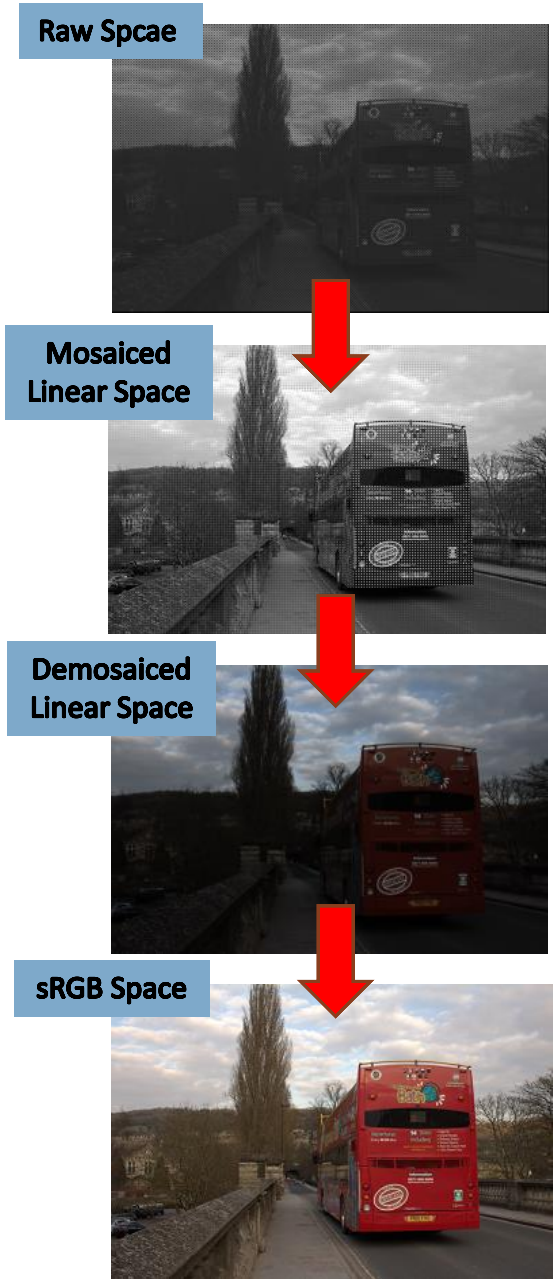


A simple camera pipeline

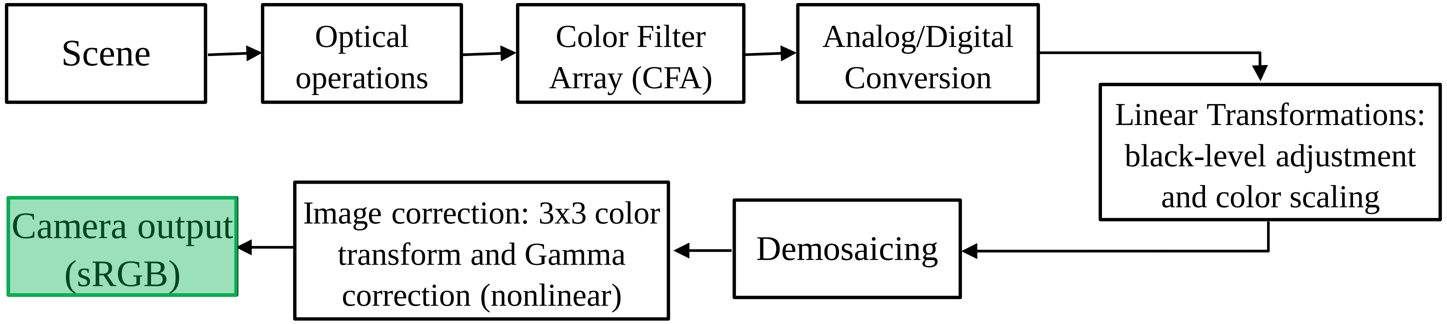


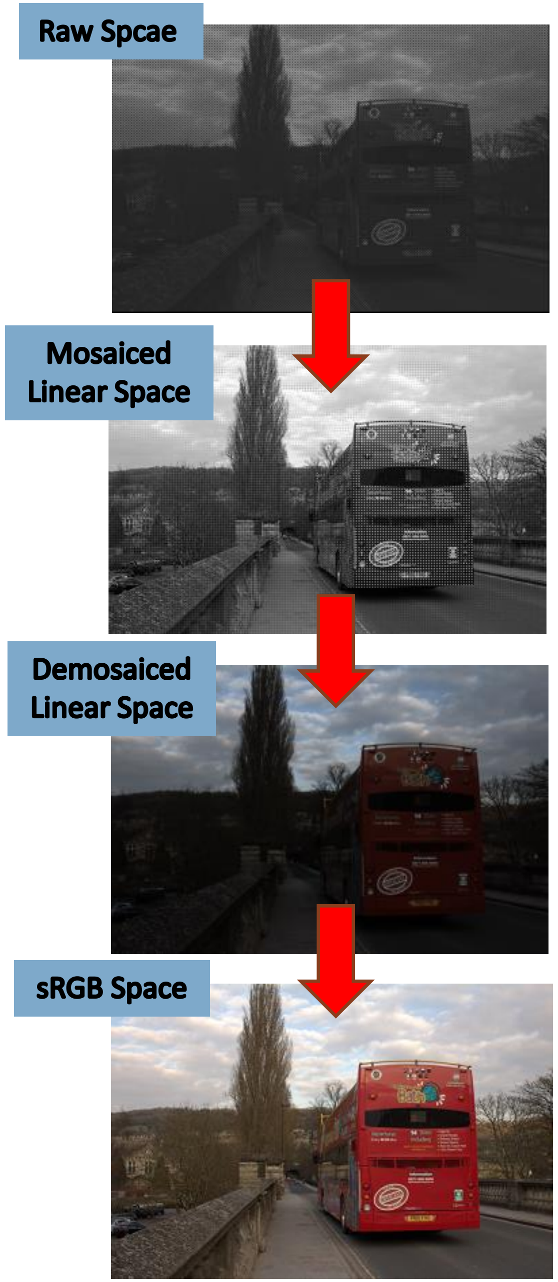


$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$



A simple camera pipeline

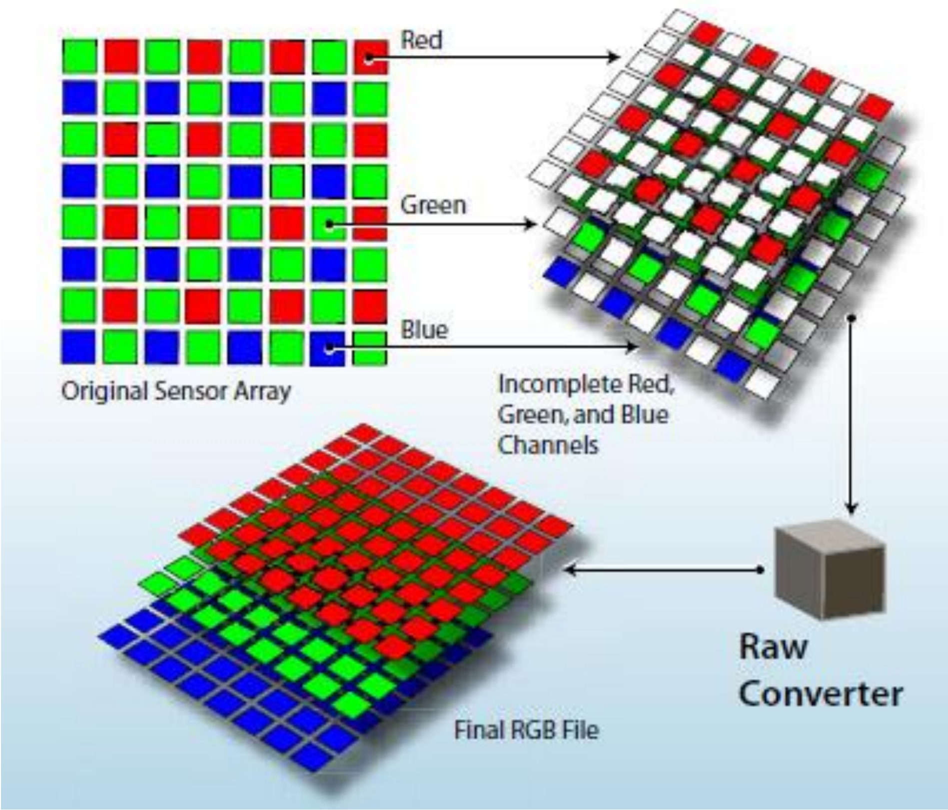




Images from: http://pho.to/media/images/digital/digital-techlology.jpg

Demosaicing

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



Demosaicing

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

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

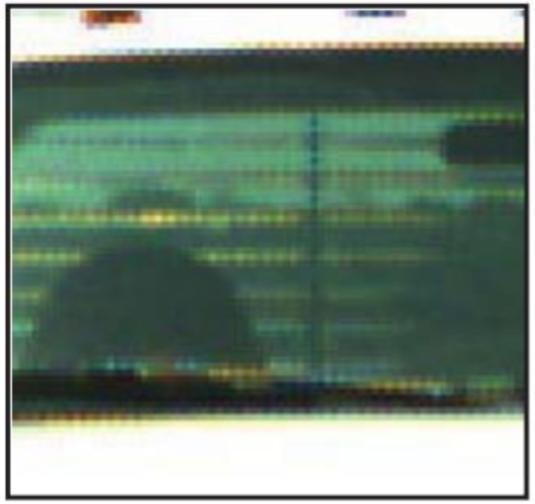


Image from: http://i.i.cbsi.com/cnwk.1d/i/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg

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/i/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg

Demosaicing

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

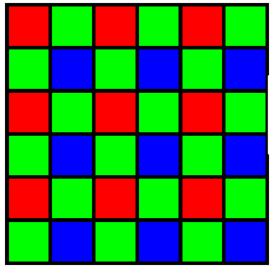
▪Isn’t a solved problem?! **NO!**

▪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



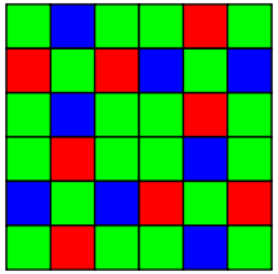
Bayer Pattern

Demosaicing

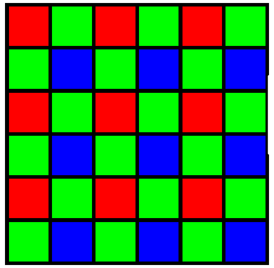
- Interpolating color-filter-array (CFA) samples to create full-resolution color images

- 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)

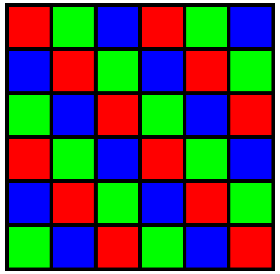
And not easily generalizable to new CFAs



Fuji X-trans



Bayer Pattern



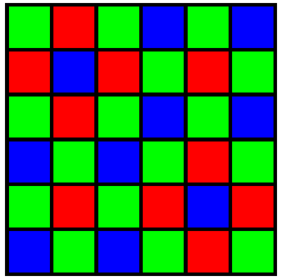


Image from: http://i.i.cbsi.com/cnwk.1d/i/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg

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

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

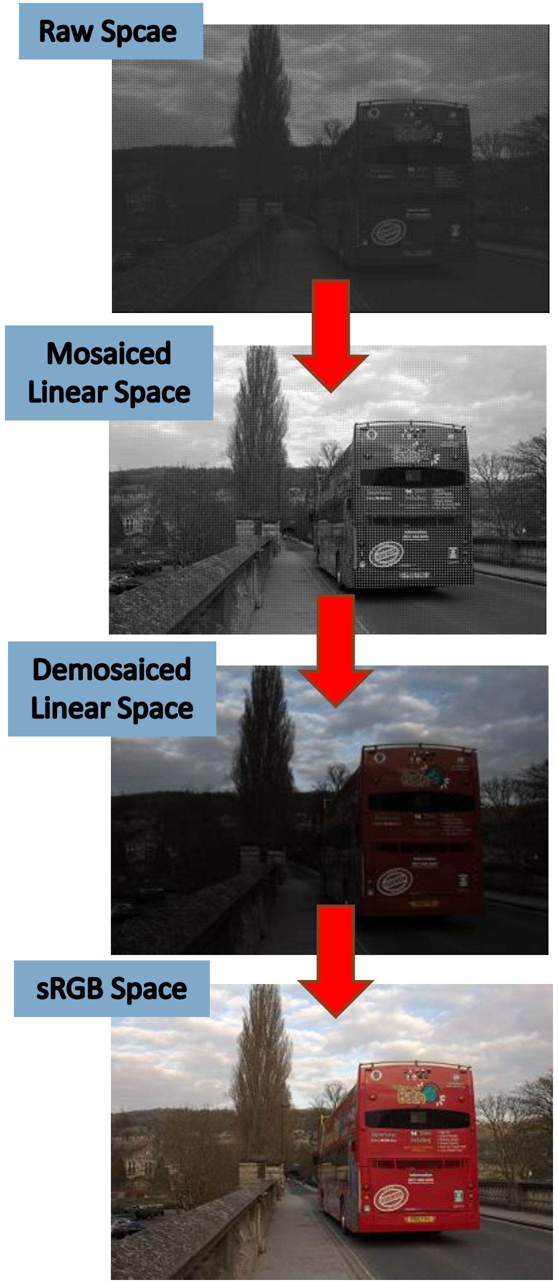


Image from: http://i.i.cbsi.com/cnwk.1d/i/tim/2012/02/06/Adobe-raw-demosaic-diagram.jpg

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

Overview of our approach

Overview of our approach

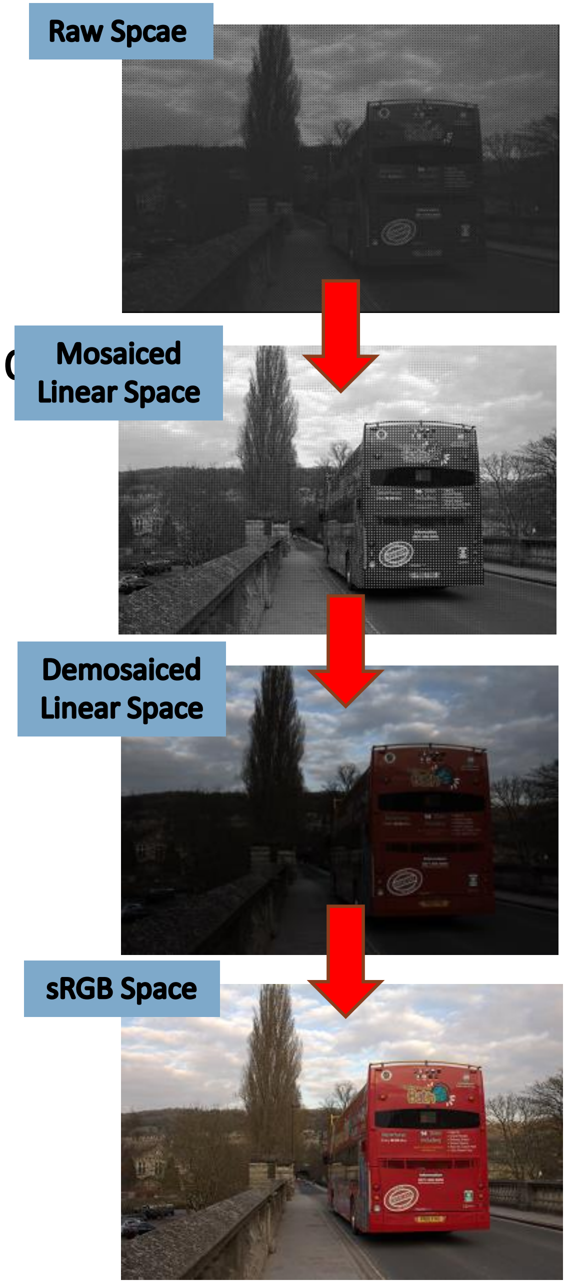
- We design a **supervised model**.

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

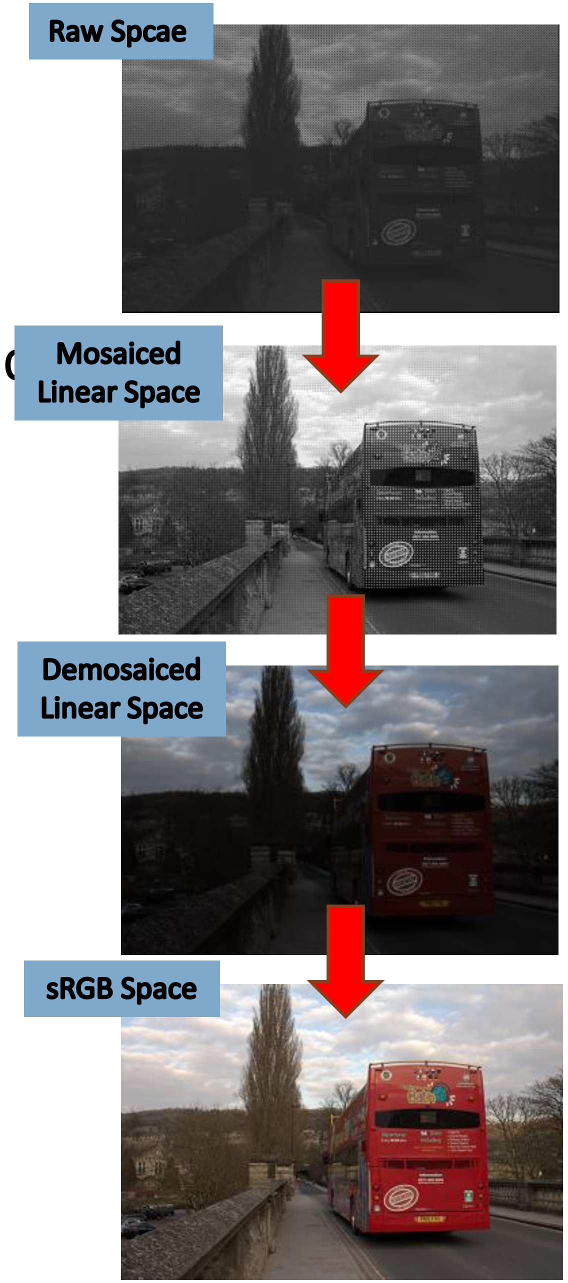
Overview of our approach

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



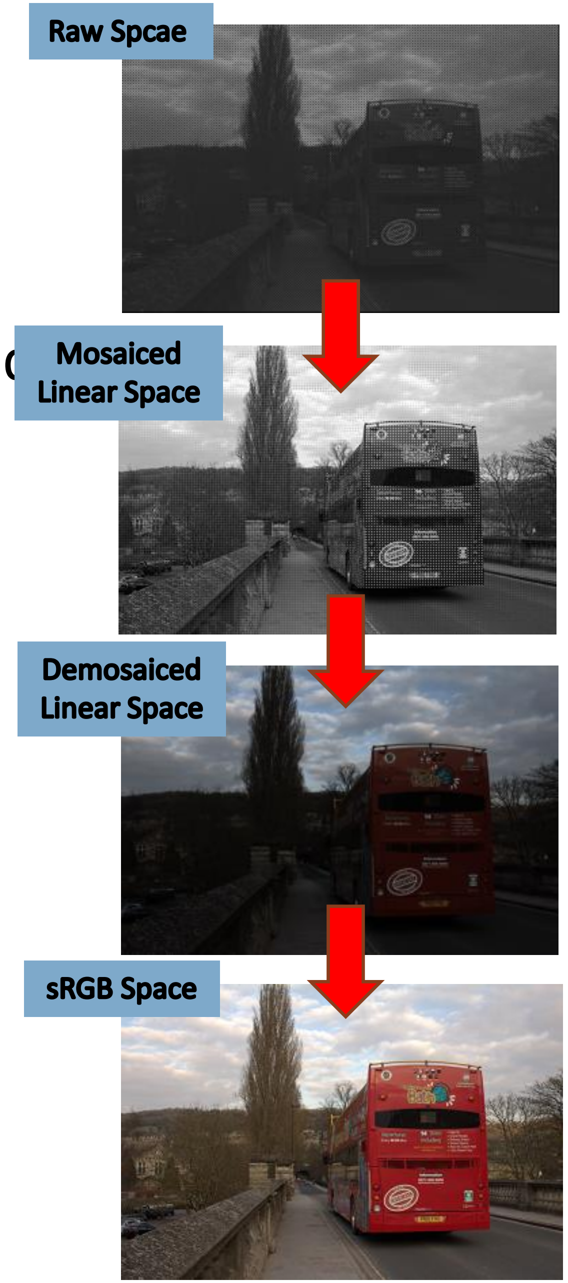
Overview of our approach

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



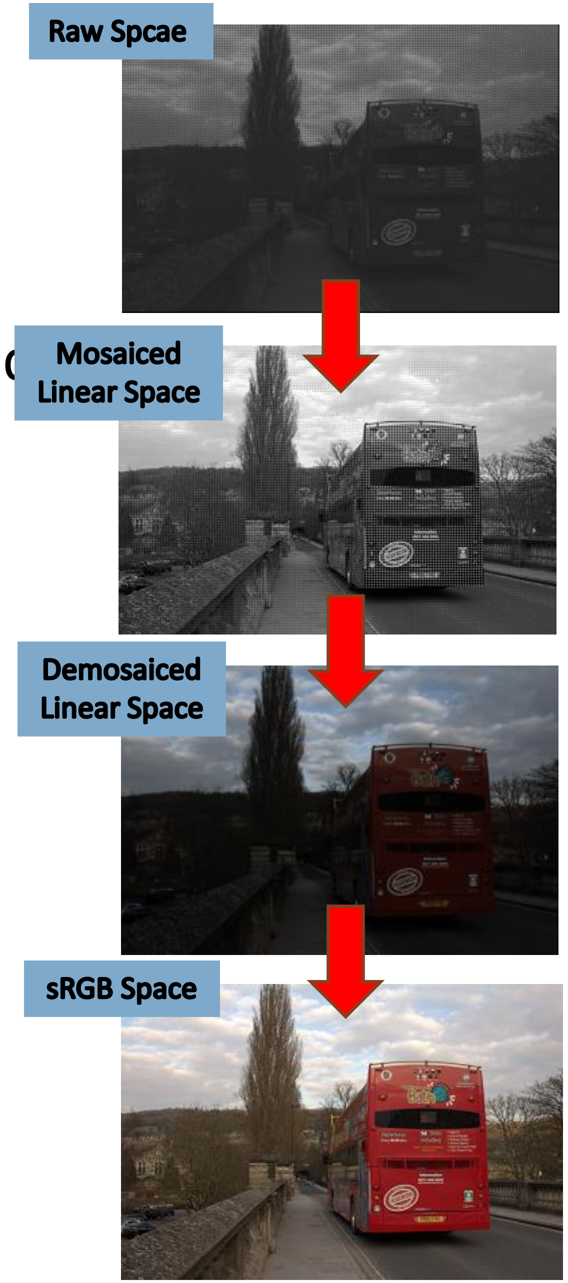
Overview of our approach

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

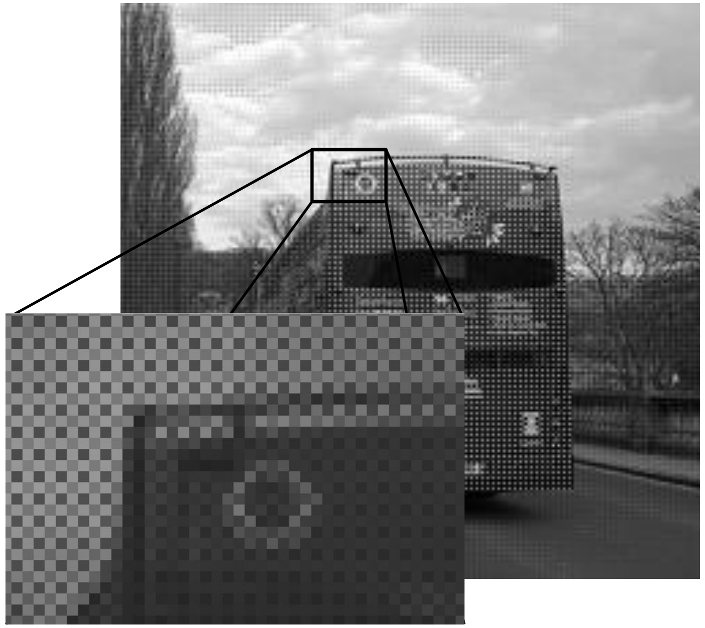


Overview of our approach

- We design a **supervised model**.
- For training the model we propose a procedure to create the ground 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**.

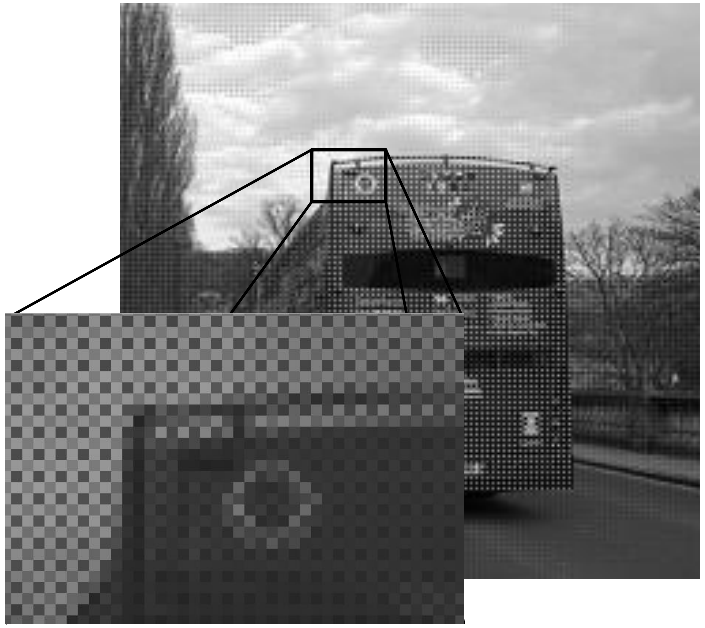


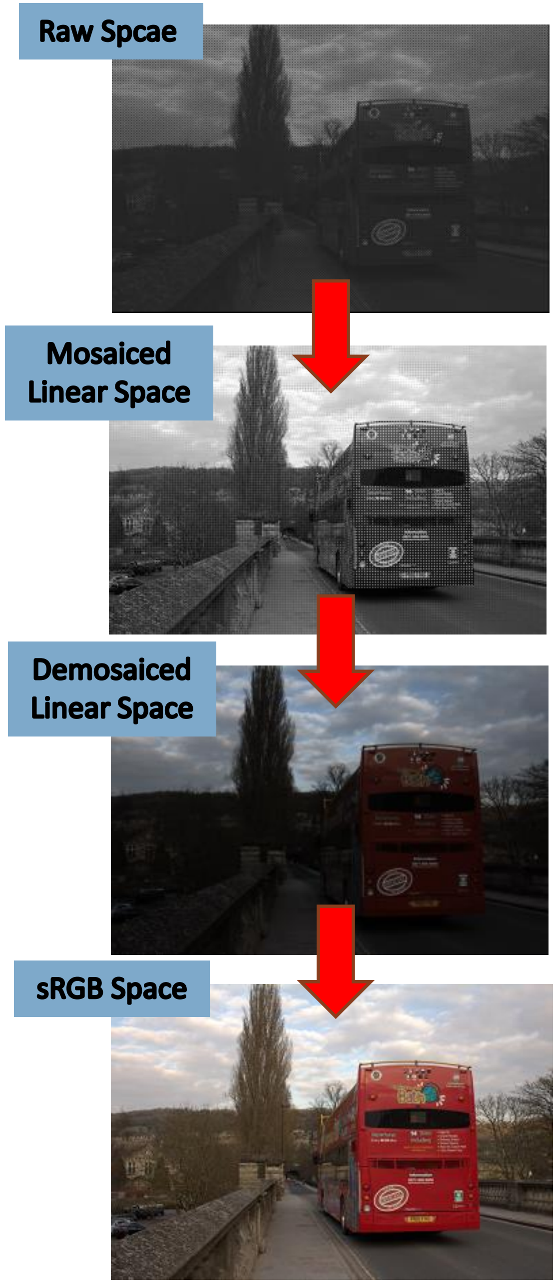
Creating ground-truth images



Creating ground-truth images

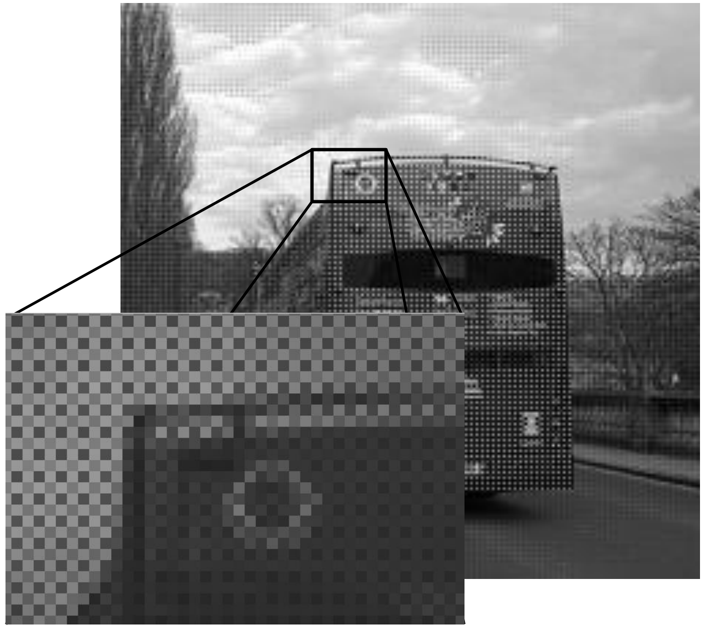
- We want to design set of **input-output** pairs in **linear light-space**

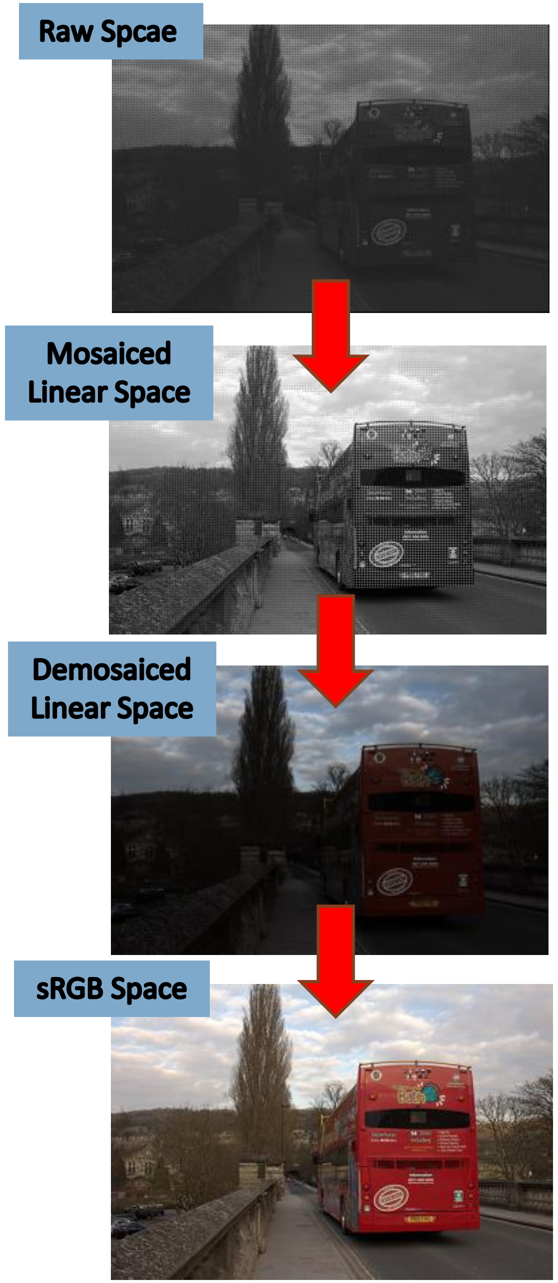




Creating ground-truth images

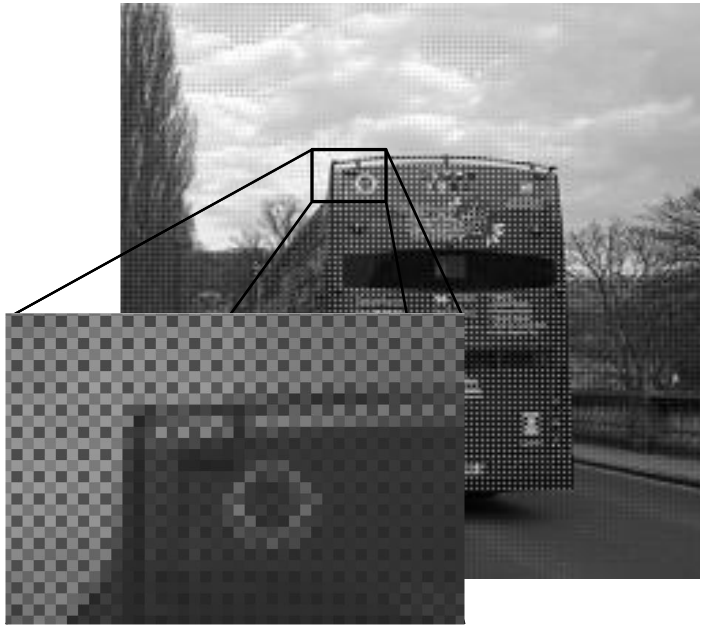
- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant
  - regardless of some properties like noise distribution

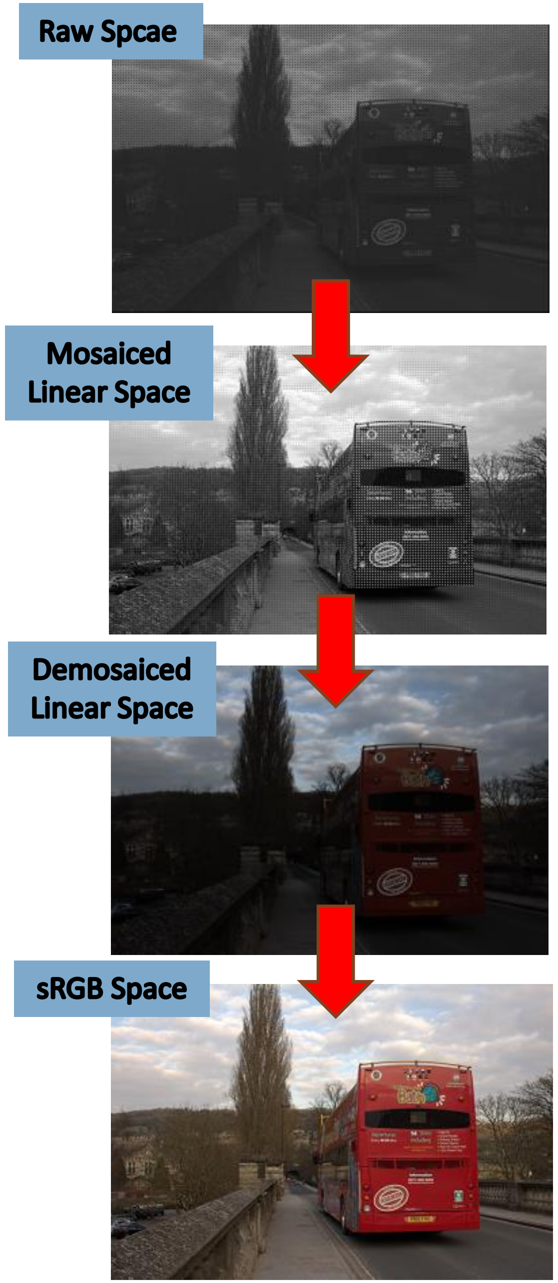




Creating ground-truth images

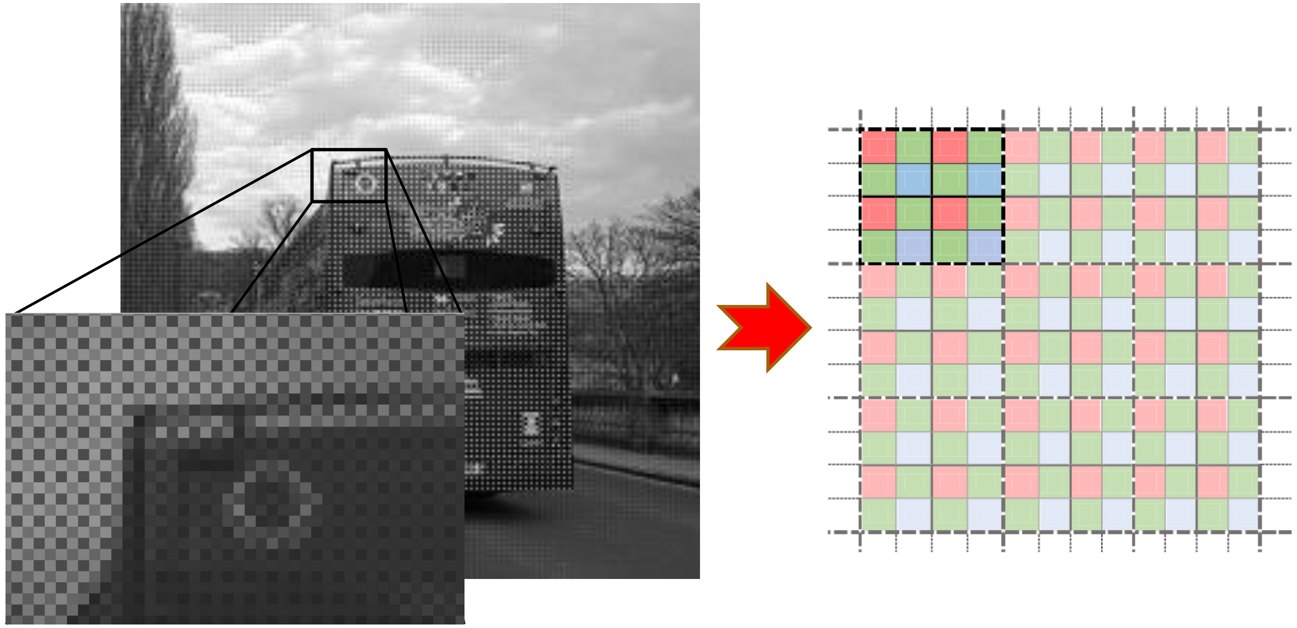
- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant
  - regardless of some properties like noise distribution

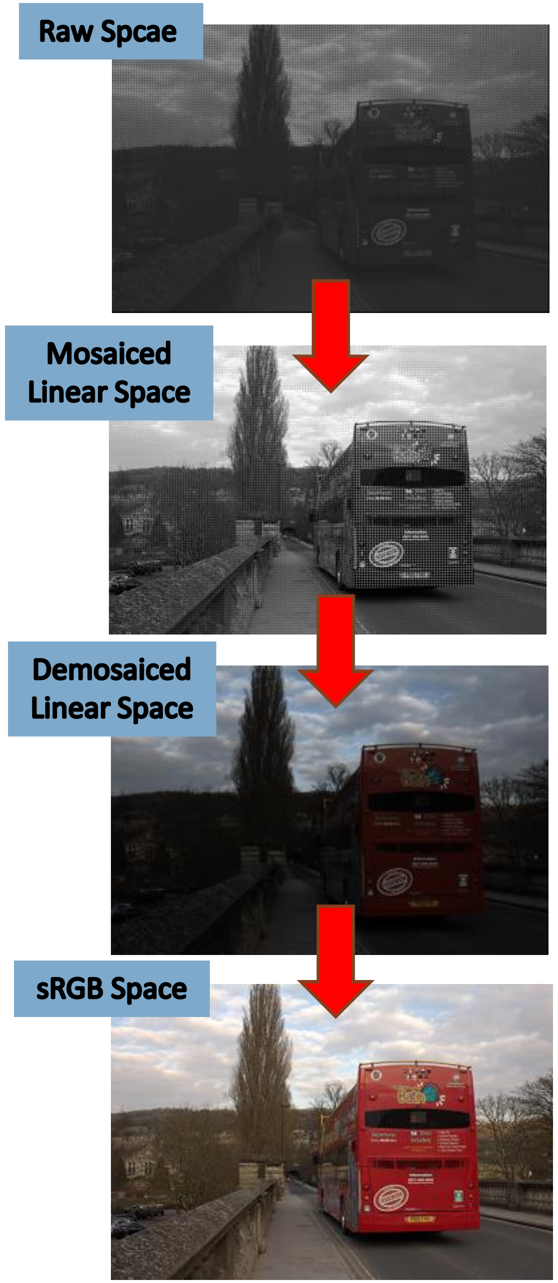




Creating ground-truth images

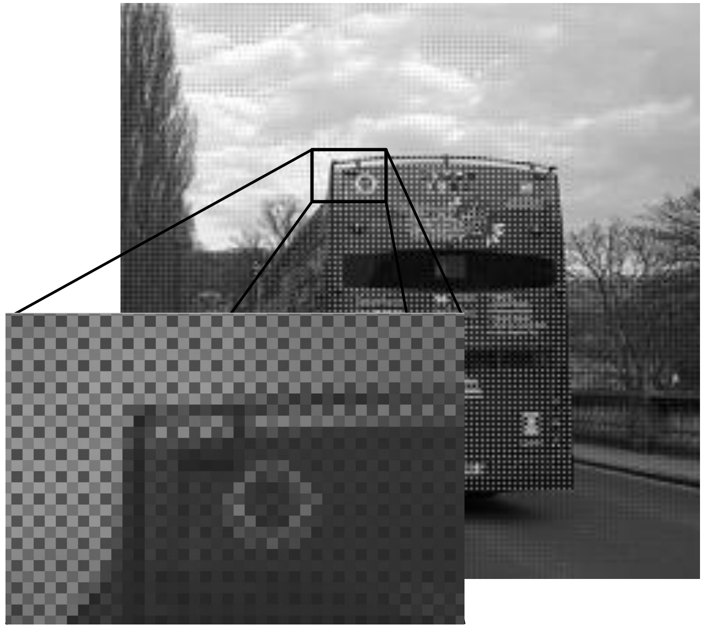
- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant
    - regardless of some properties like noise distribution

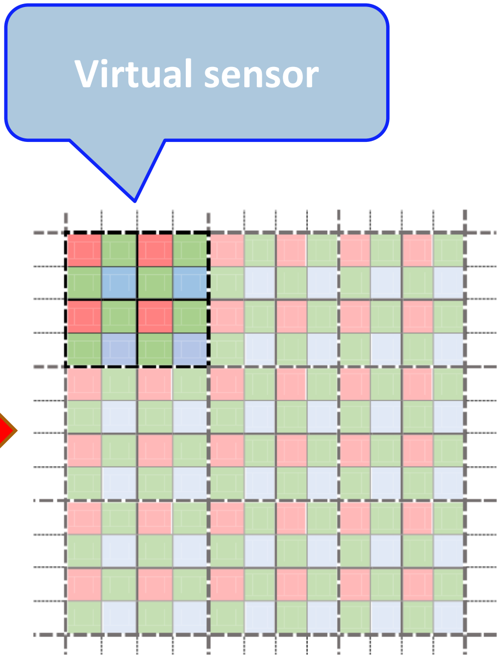


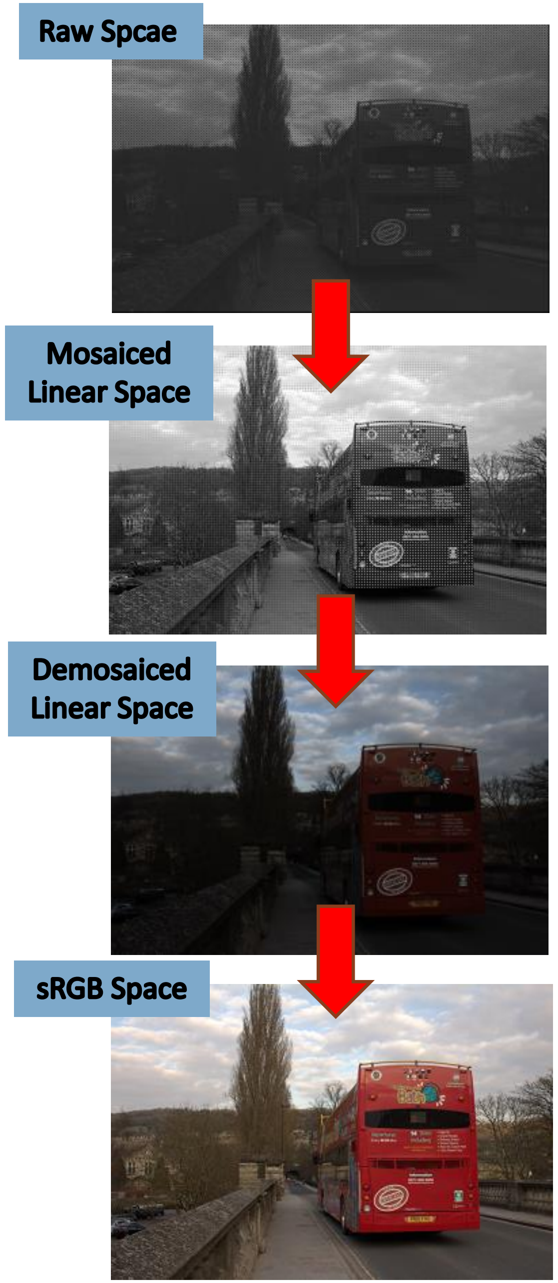


Creating ground-truth images

- 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

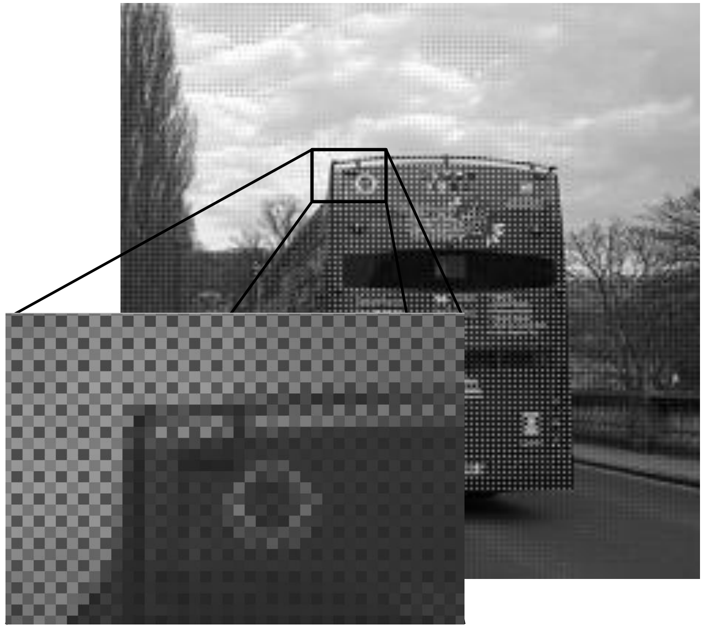


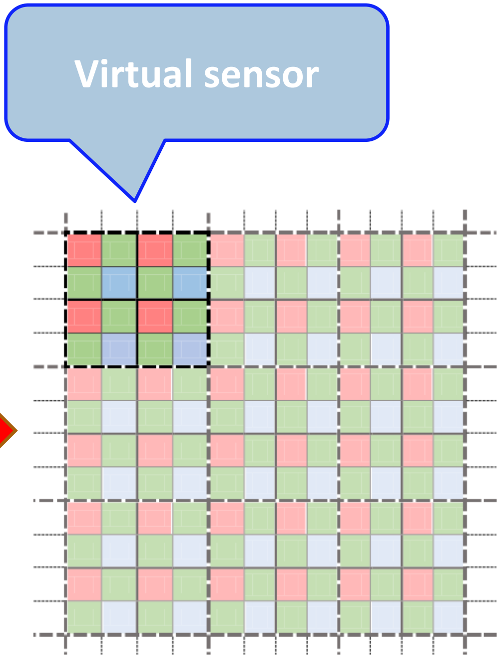


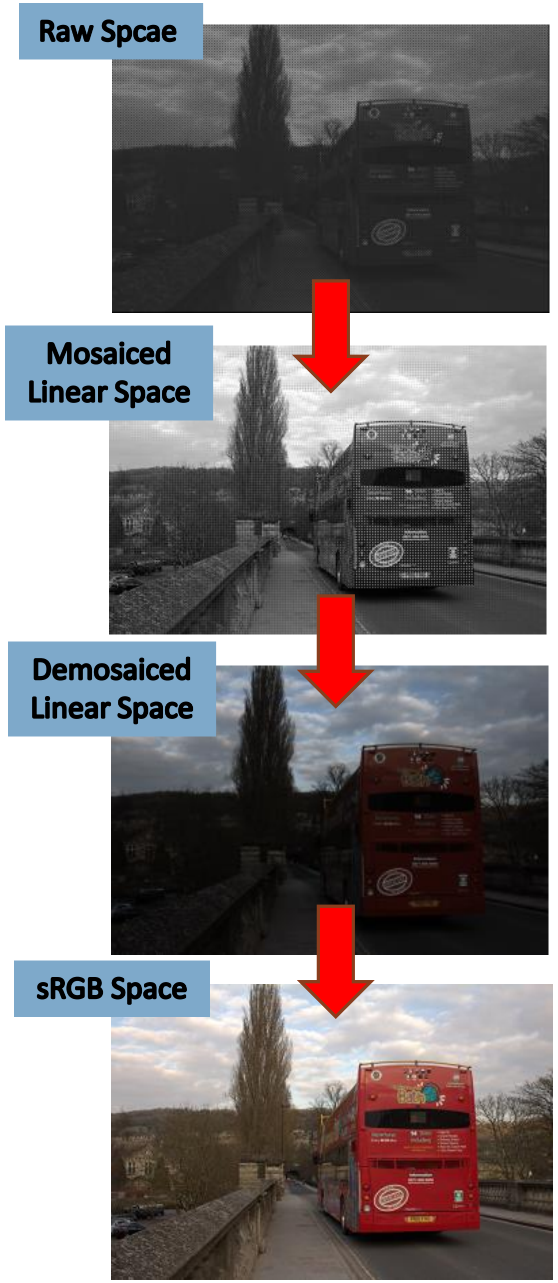


Creating ground-truth images

- 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

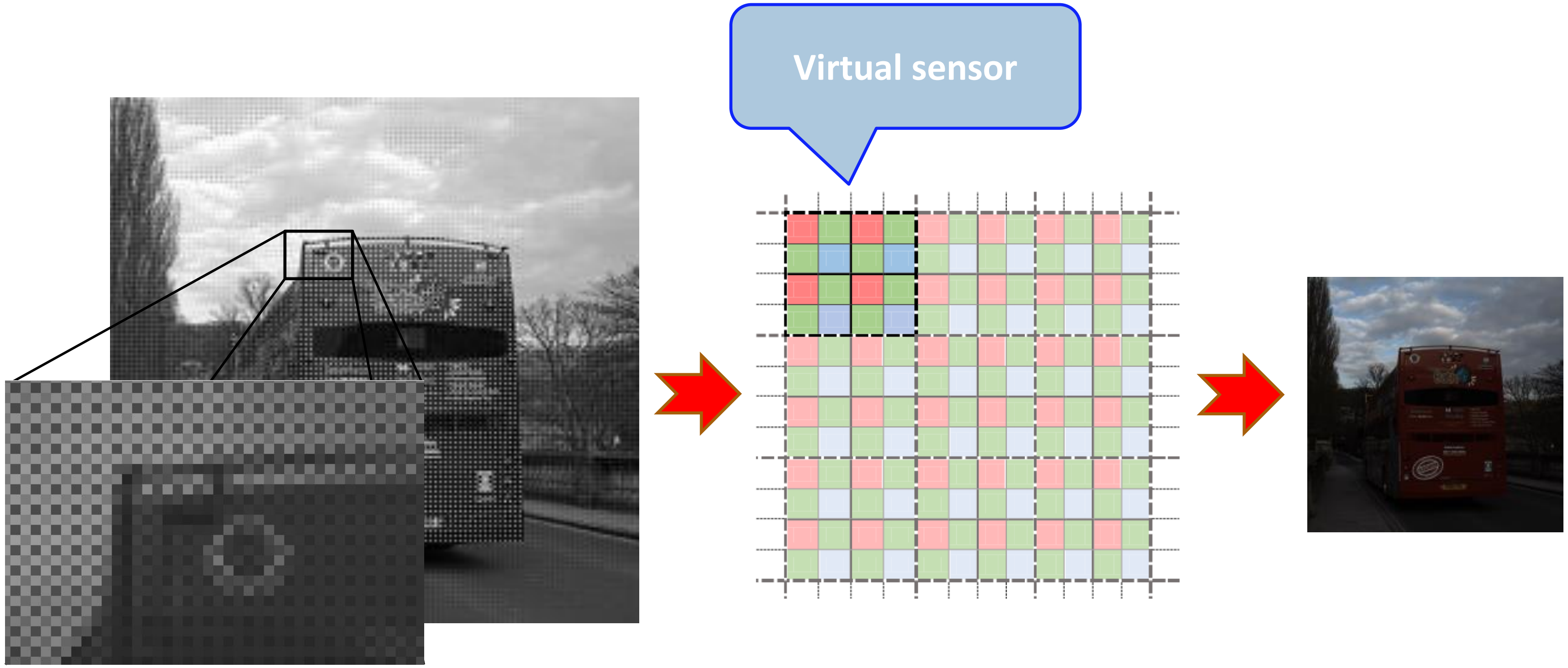


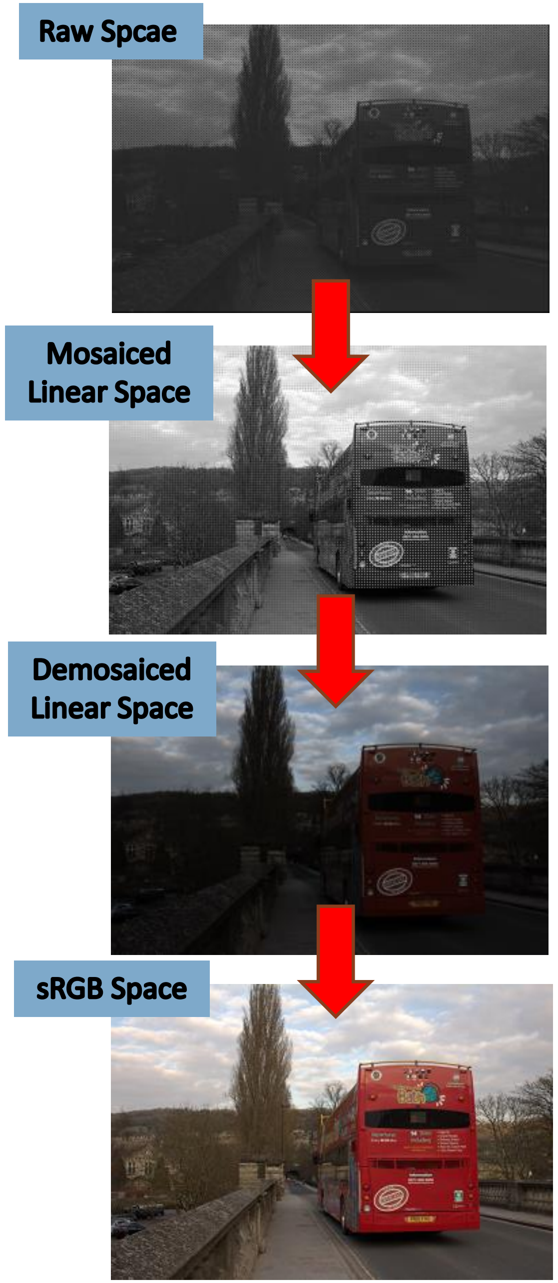




Creating ground-truth images

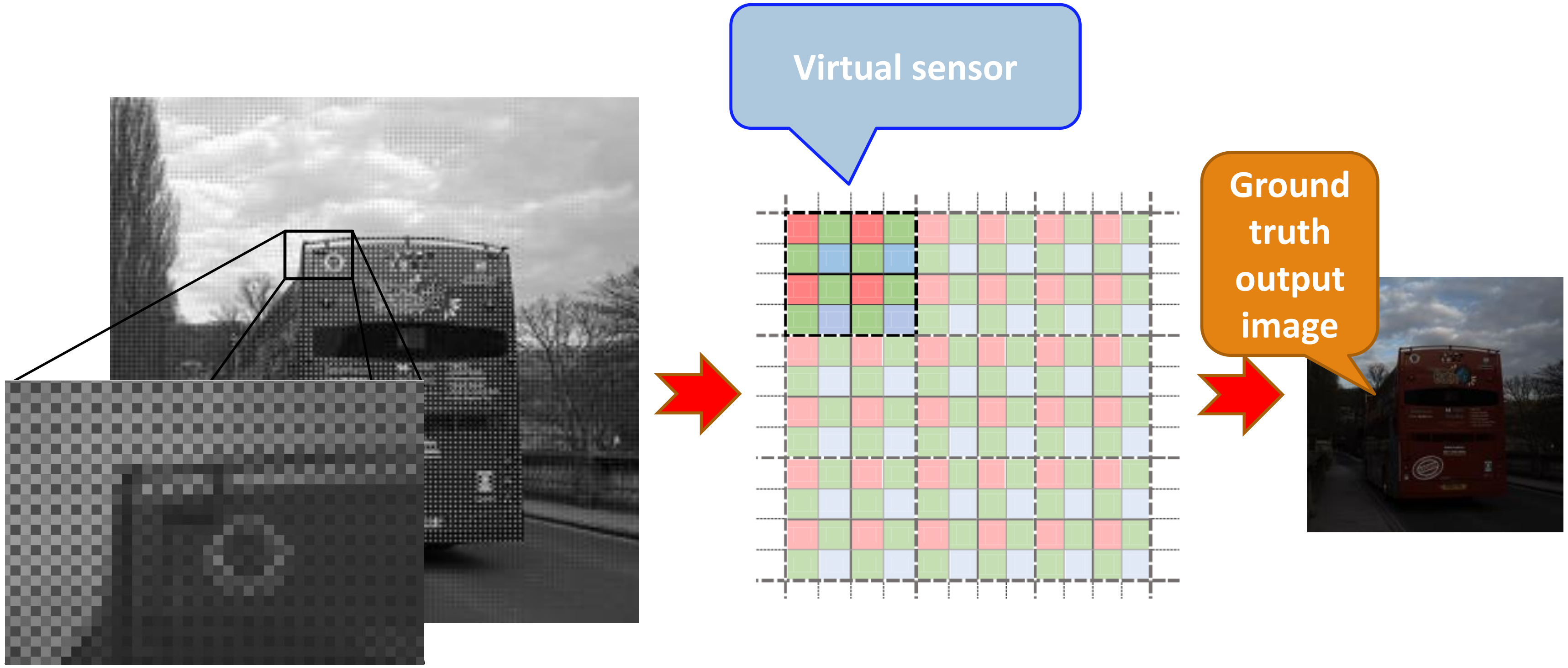
- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant
    - regardless of some properties like noise distribution

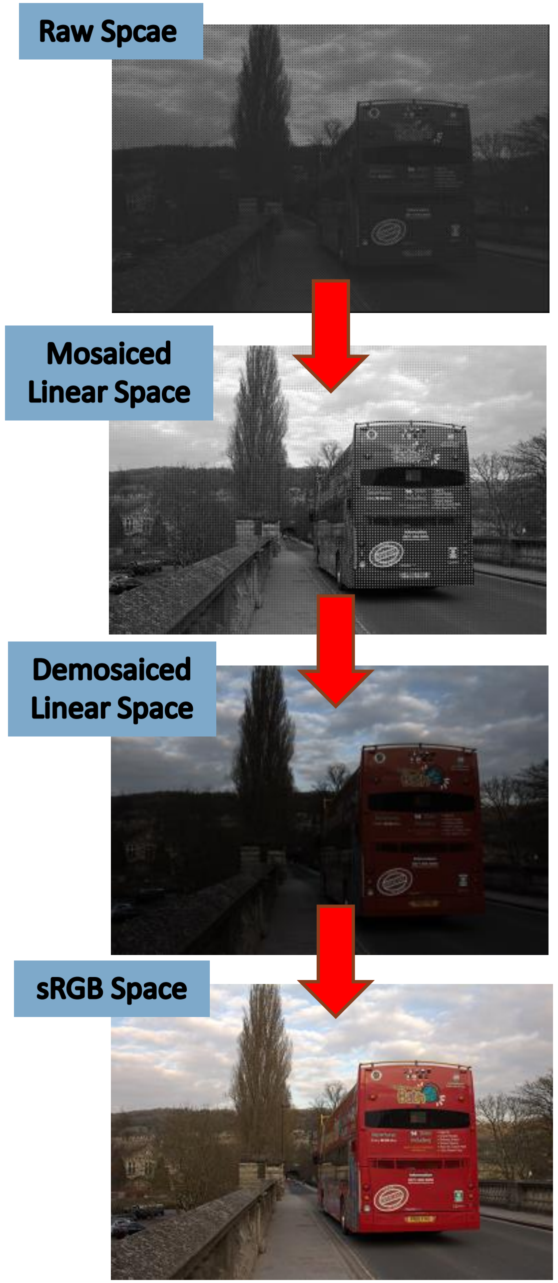




Creating ground-truth images

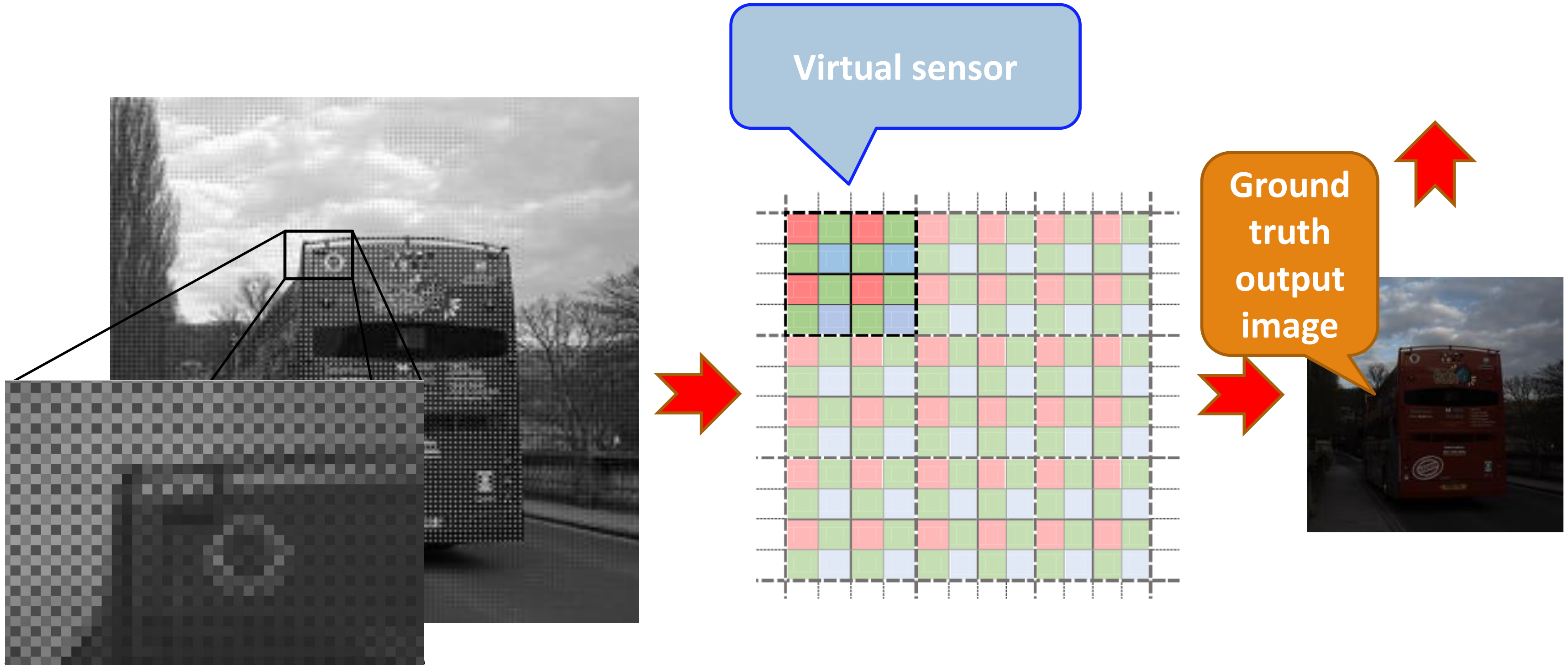
- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant
  - regardless of some properties like noise distribution

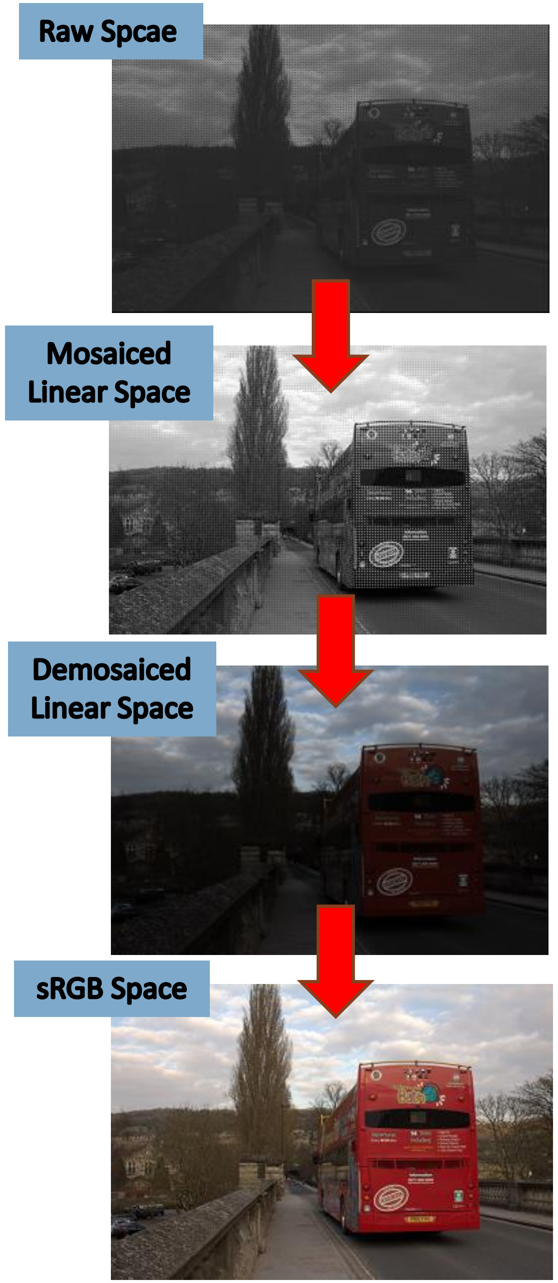




Creating ground-truth images

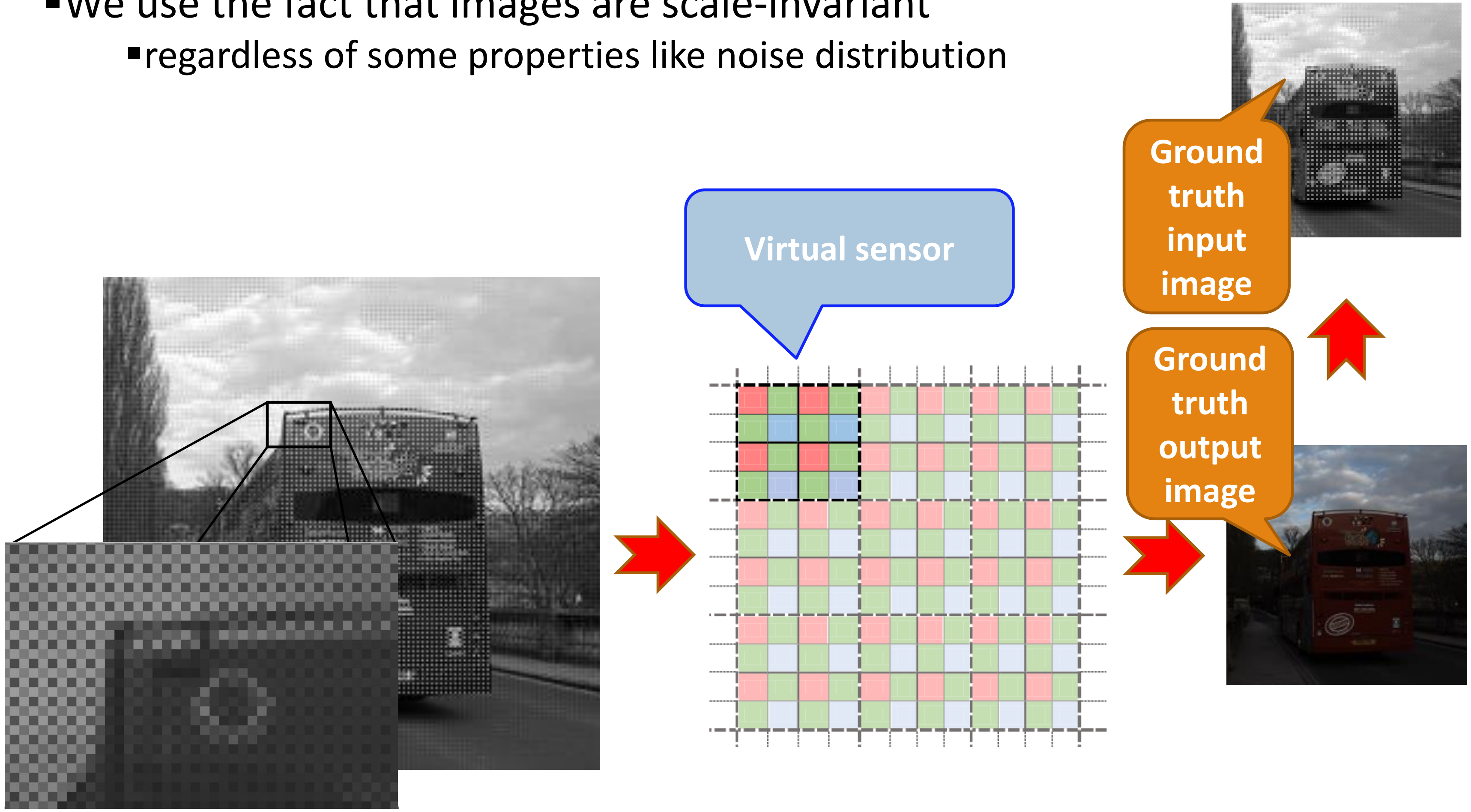
- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant
  - regardless of some properties like noise distribution

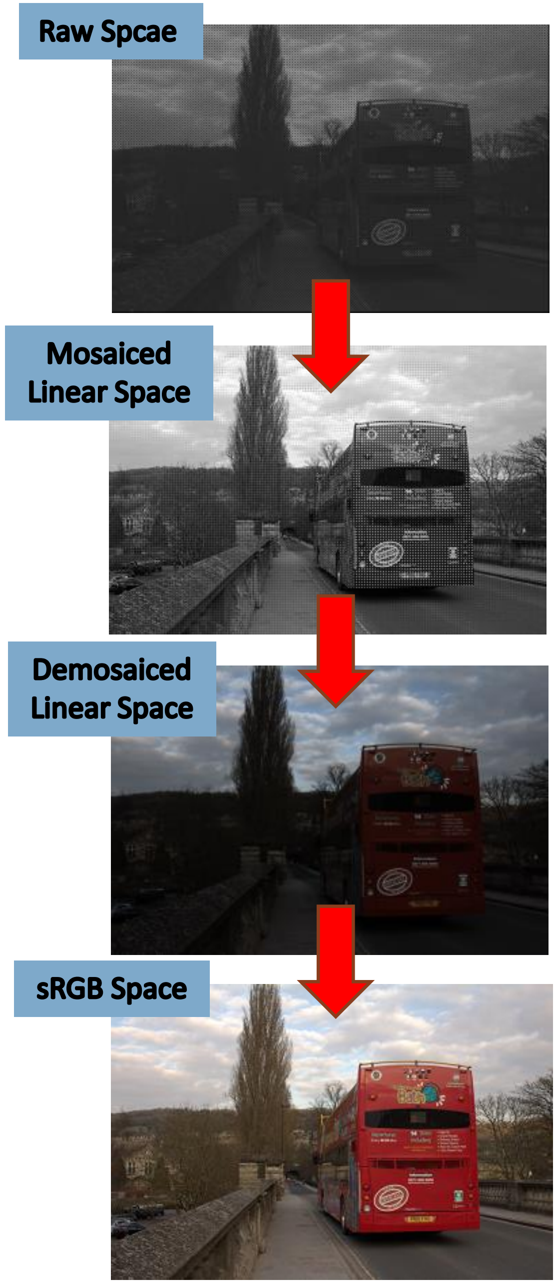




Creating ground-truth images

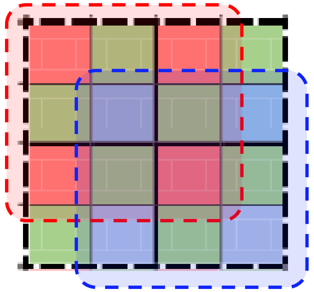
- We want to design set of **input-output** pairs in **linear light-space**
- We use the fact that images are scale-invariant
  - regardless of some properties like noise distribution





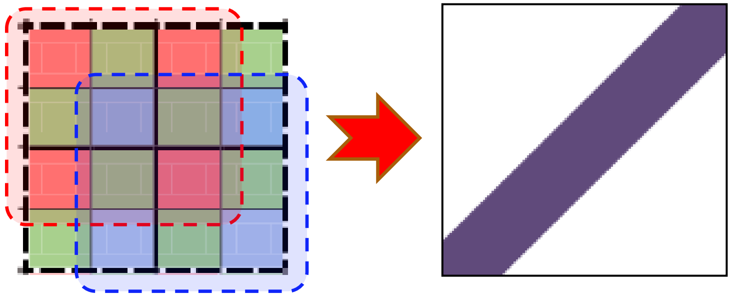
down-scaling strategies

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



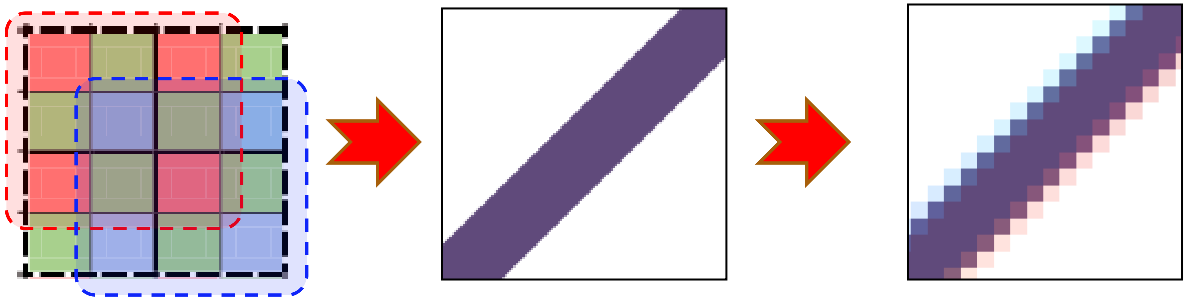
down-scaling strategies

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



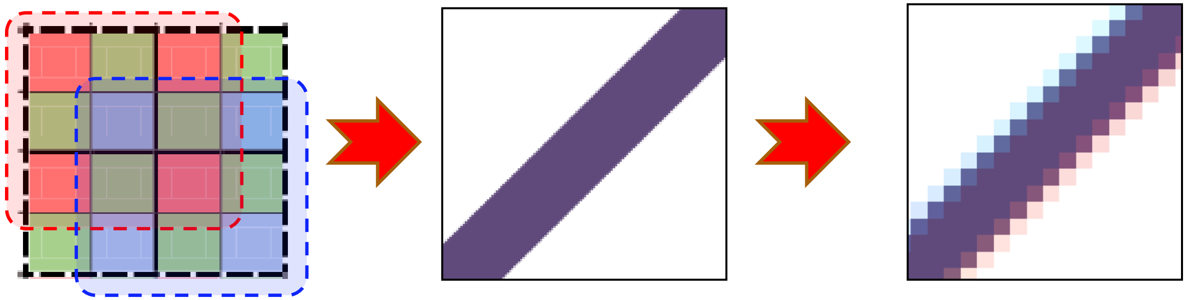
down-scaling strategies

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



down-scaling strategies

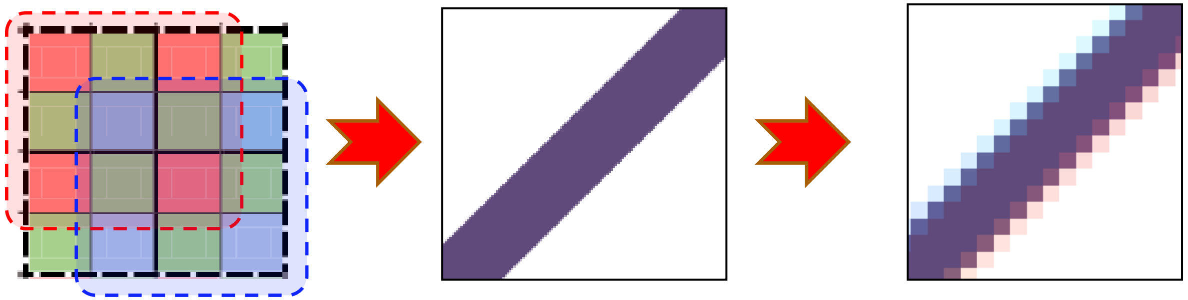
- There is a systematic bias between red and blue, when the size of the square is **even**



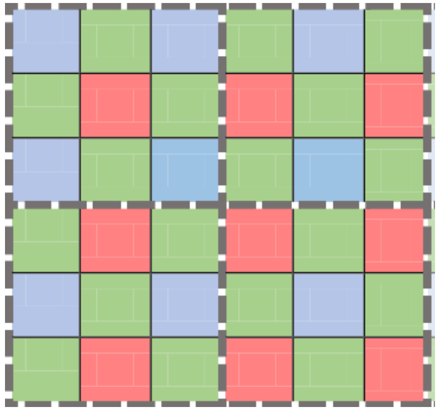
- When the size of the block is **odd**, this systematic bias vanishes, but the ratio of color sensors are different
  - This ratio goes to **one**, when the size of the block increases

down-scaling strategies

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

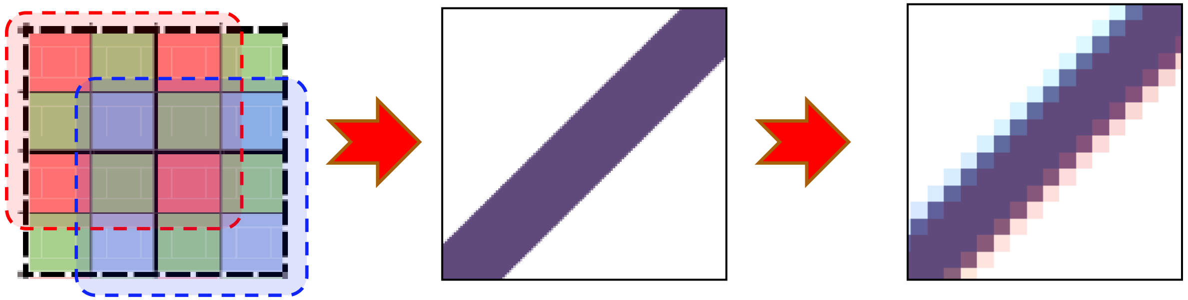


- When the size of the block is **odd**, this systematic bias vanishes, but the ratio of color sensors are different
    - This ratio goes to **one**, when the size of the block increases

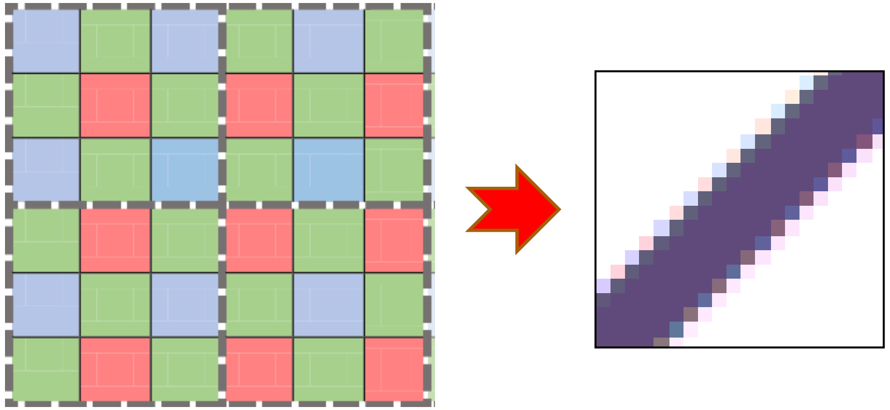


down-scaling strategies

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

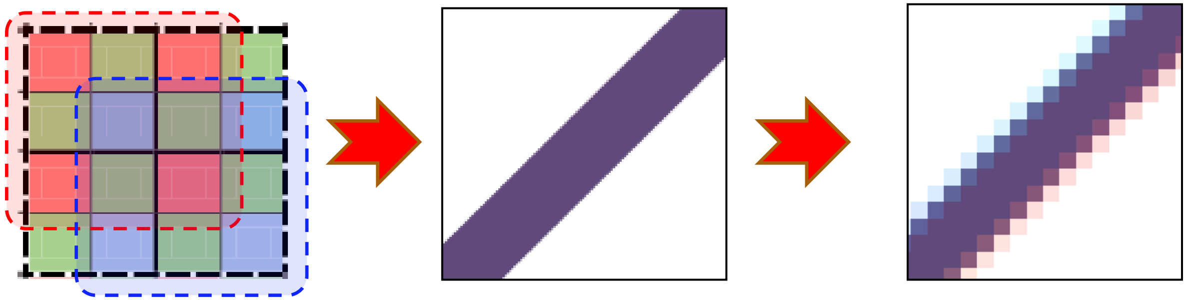


- When the size of the block is **odd**, this systematic bias vanishes, but the ratio of color sensors are different
  - This ratio goes to **one**, when the size of the block increases

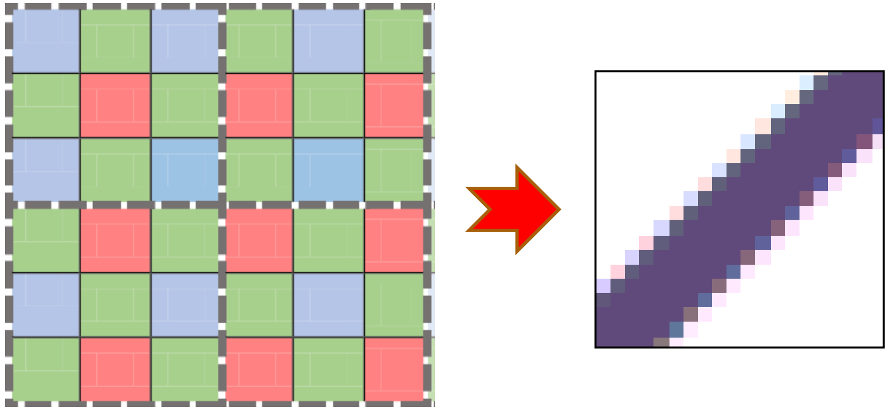


down-scaling strategies

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



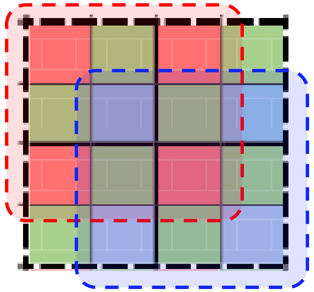
- When the size of the block is **odd**, this systematic bias vanishes, but the ratio of color sensors are different
  - This ratio goes to **one**, when the size of the block increases



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

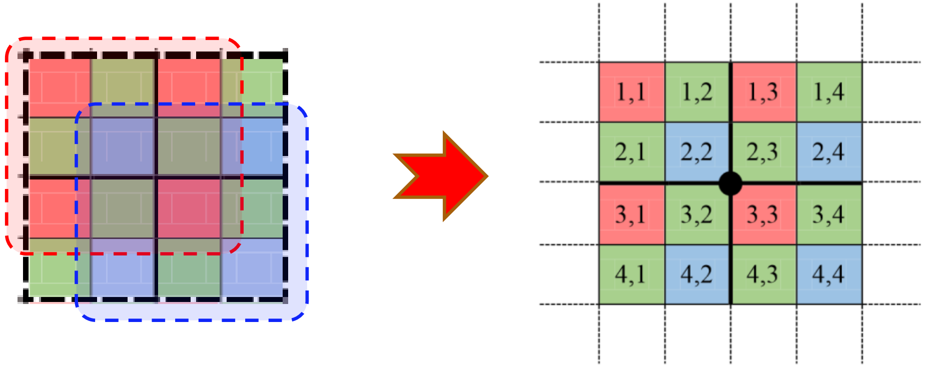
Max-Ent down-scaling

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



Max-Ent down-scaling

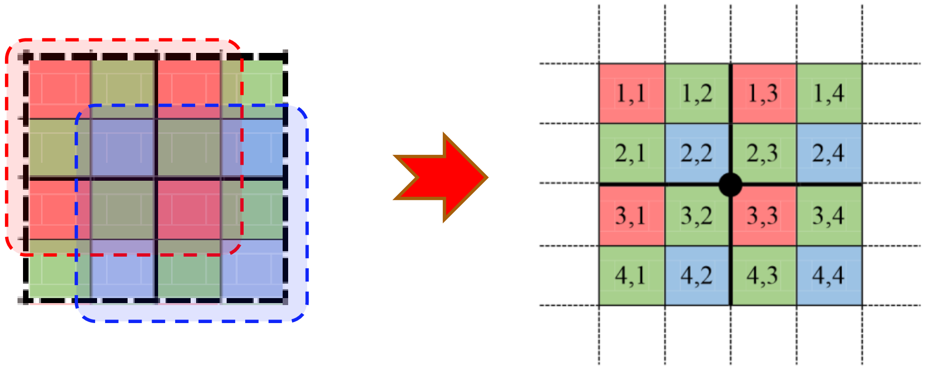
- Goal:
  - Want to bring the **center of the mass**, for each color, to the center of each block



Max-Ent down-scaling

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

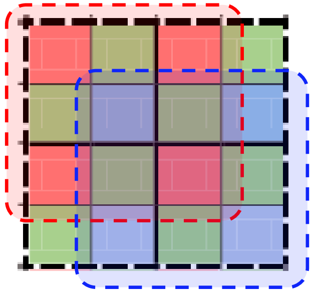
$$
\begin{cases}
\max_p \left\{ \mathcal{H}(p) = \sum_{x,y} p(x,y) \log p(x,y) \right\} \\
\sum_{x,y} p(x,y) = 1 \\
p(x,y) \geq 0, \forall x, y \\
\sum_{x,y} x p(x,y) = W + 0.5, \forall y \\
\sum_{x,y} y p(x,y) = W + 0.5, \forall x
\end{cases}
$$

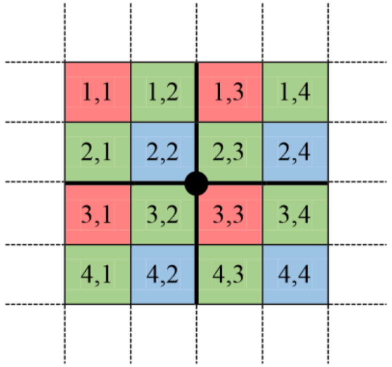


Max-Ent down-scaling

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

$$\begin{cases} \max_p \left\{ \mathcal{H}(p) = \sum_{x,y} p(x,y) \log p(x,y) \right\} \\ \sum_{x,y} p(x,y) = 1 \\ p(x,y) \geq 0, \forall x,y \\ \sum_{x,y} x p(x,y) = W + 0.5, \forall y \\ \sum_{x,y} y p(x,y) = W + 0.5, \forall x \end{cases}$$



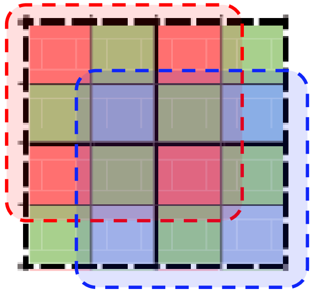


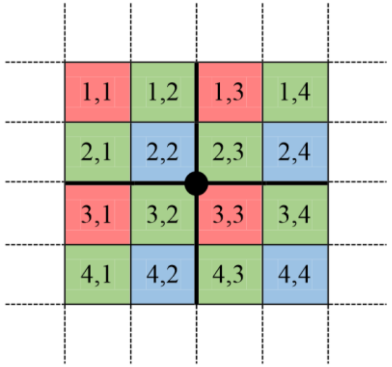
$$p_r(x,y) = \frac{1}{16}\left[\begin{array}{cc|cc} 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 3 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right], \quad p_b(x,y) = \frac{1}{16}\left[\begin{array}{cc|cc} 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 3 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 1 \end{array}\right] \quad p_g(x,y) = \frac{1}{8}\left[\begin{array}{cc|cc} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{array}\right]$$

Max-Ent down-scaling

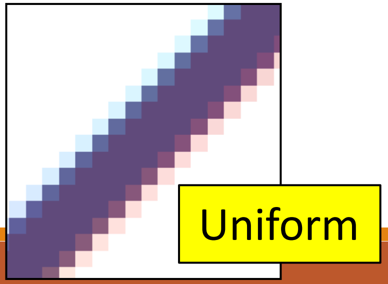
- Goal:
  - Want to bring the **center of the mass**, for each color, to the center of each block

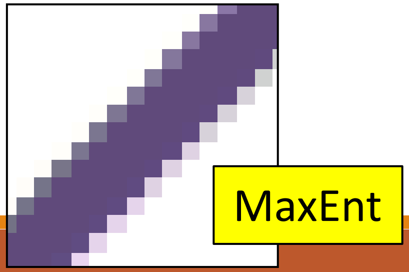
$$\begin{cases} \max_p \left\{ \mathcal{H}(p) = \sum_{x,y} p(x,y) \log p(x,y) \right\} \\ \sum_{x,y} p(x,y) = 1 \\ p(x,y) \geq 0, \forall x, y \\ \sum_{x,y} x p(x,y) = W + 0.5, \forall y \\ \sum_{x,y} y p(x,y) = W + 0.5, \forall x \end{cases}$$





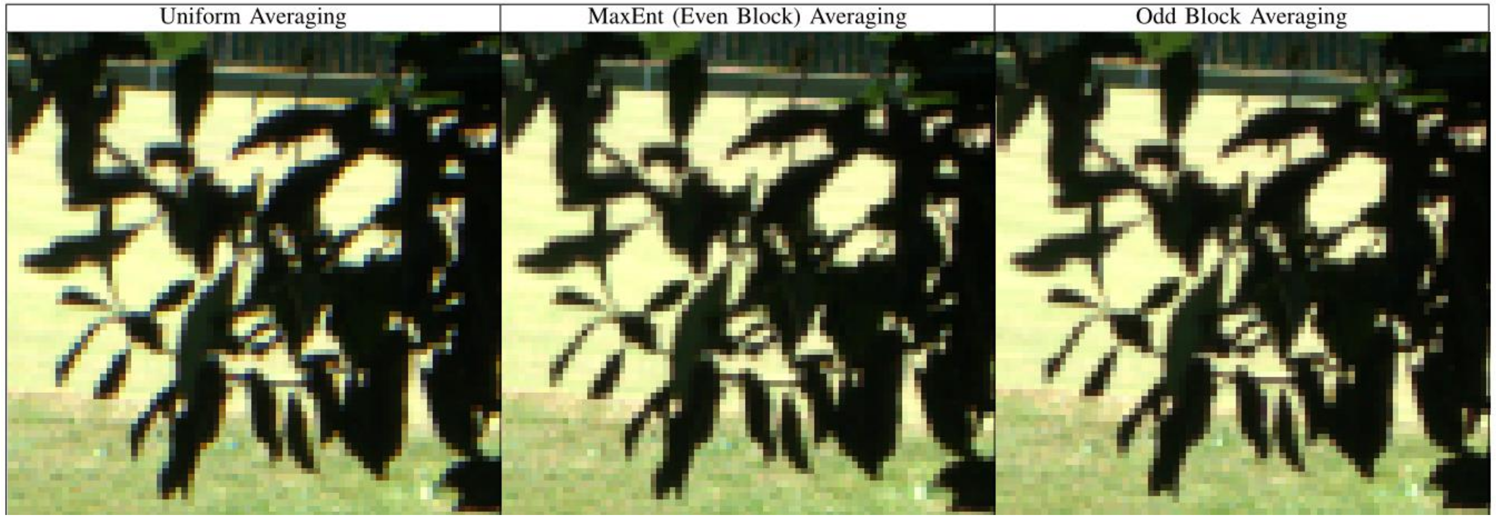
$$p_r(x,y) = \frac{1}{16}\left[\begin{array}{cc|cc} 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 3 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right], \quad p_b(x,y) = \frac{1}{16}\left[\begin{array}{cc|cc} 0 & 0 & 0 & 0 \\ 0 & 9 & 0 & 3 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 1 \end{array}\right] \quad p_g(x,y) = \frac{1}{8}\left[\begin{array}{cc|cc} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{array}\right]$$



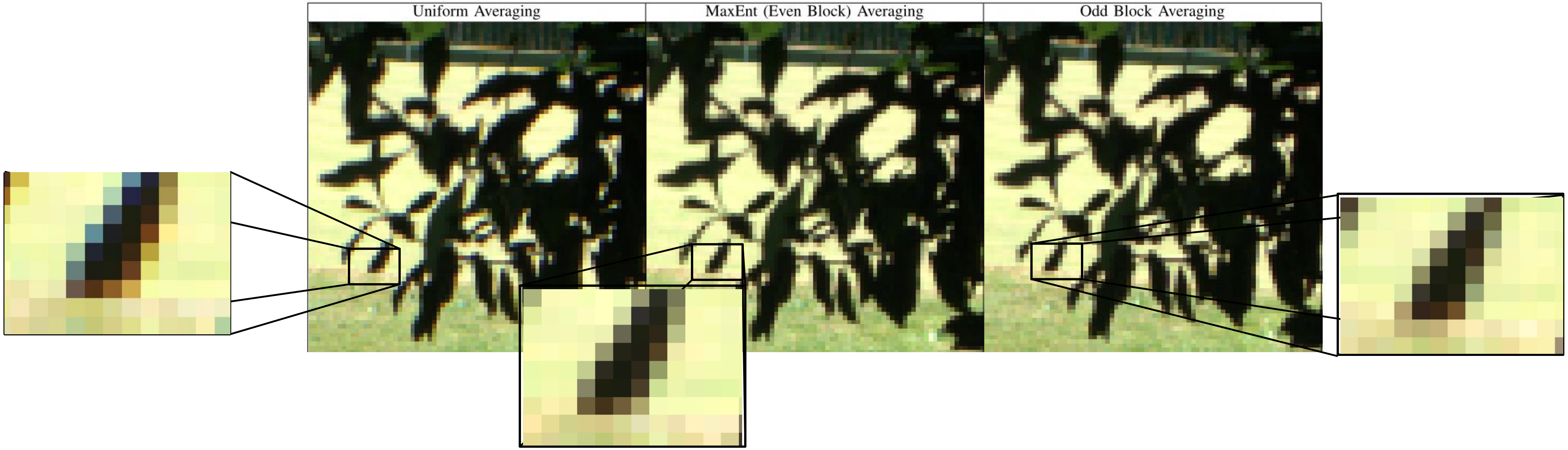


Down-scaling, a real example

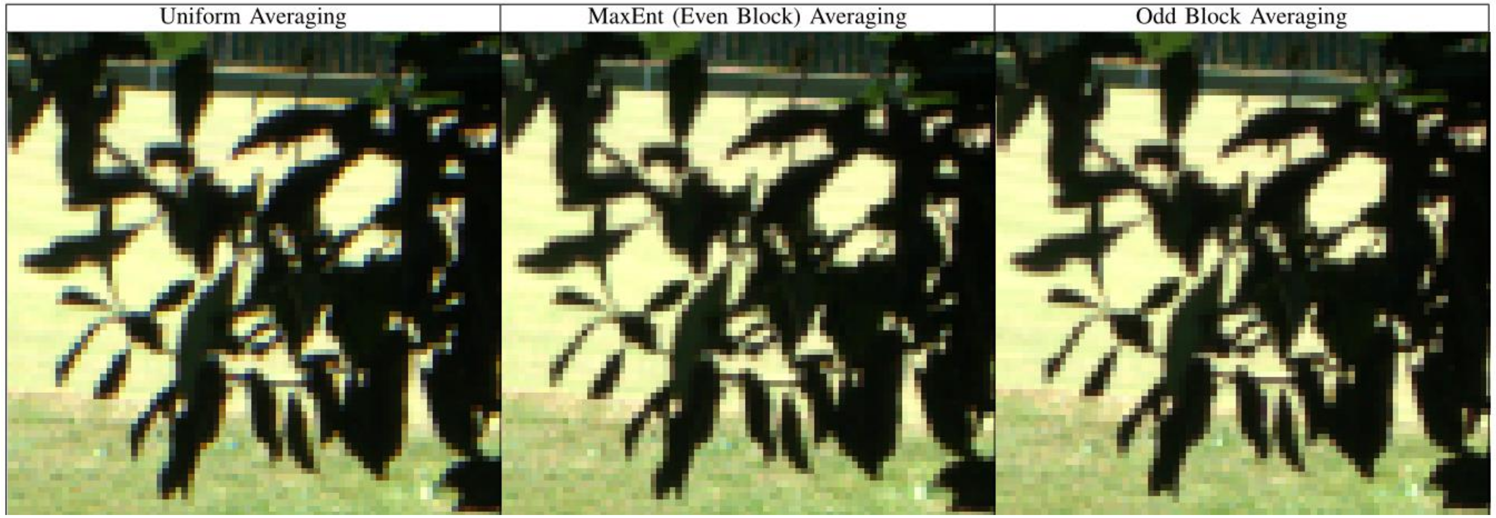
Down-scaling, a real example



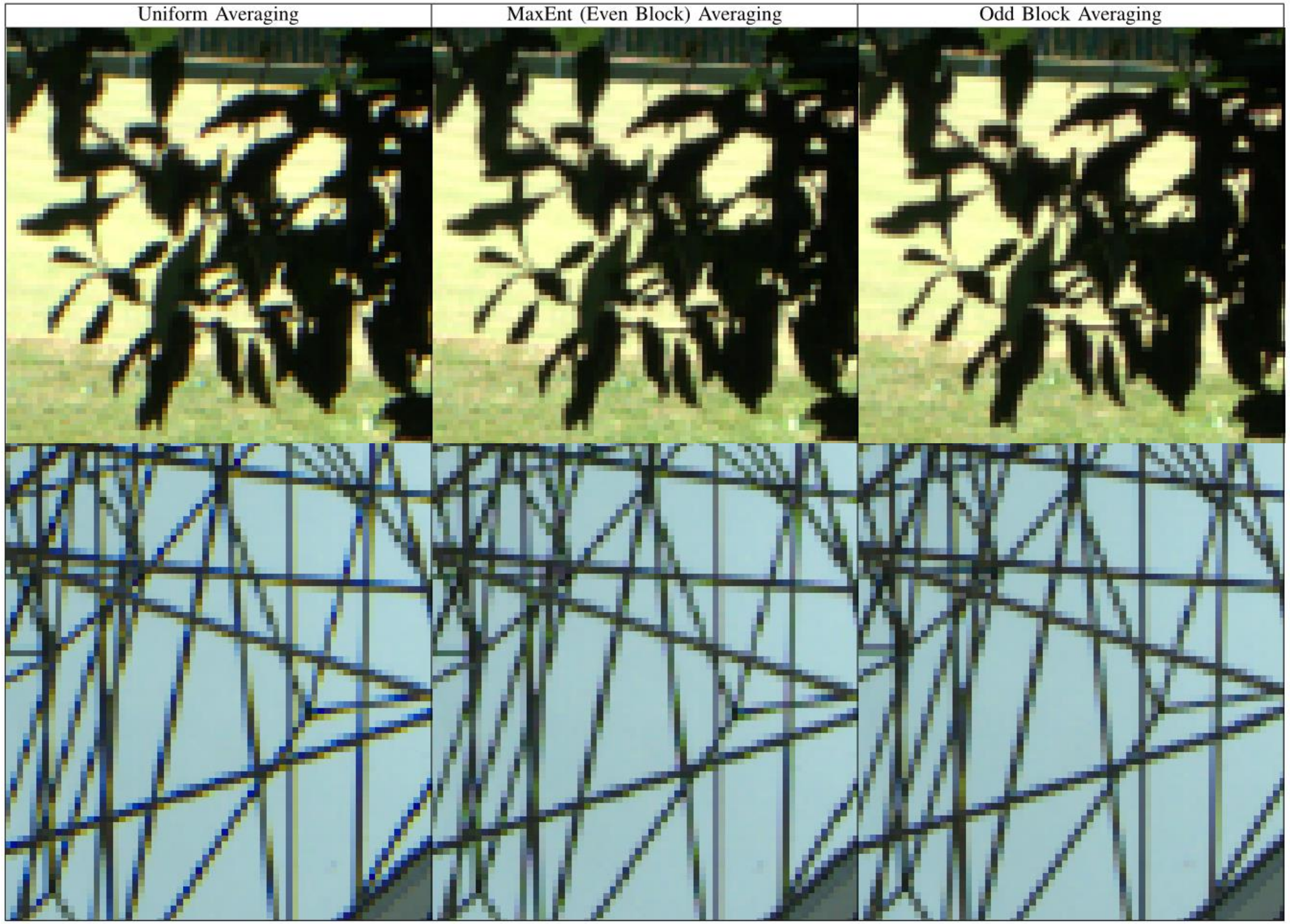
Down-scaling, a real example



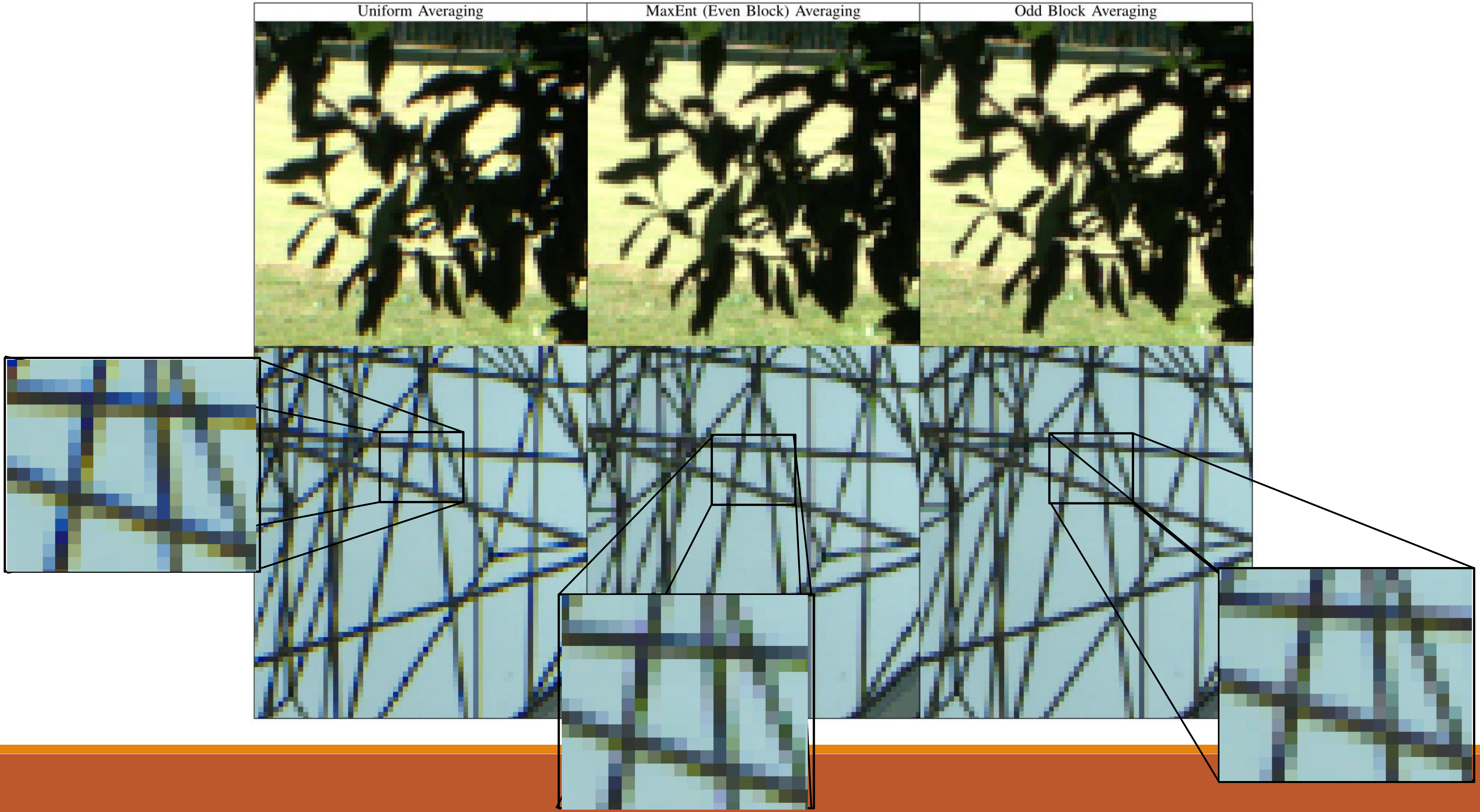
Down-scaling, a real example



Down-scaling, a real example



Down-scaling, a real example



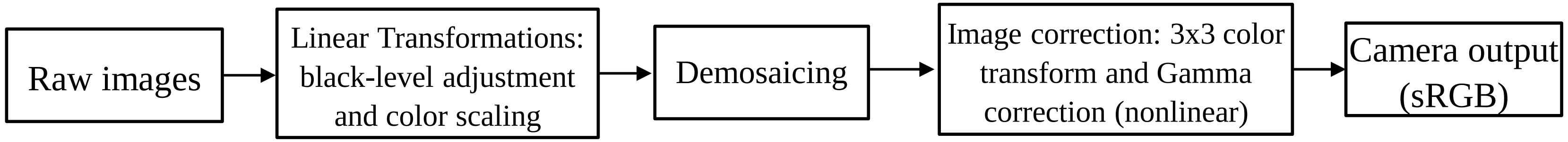
Creating ground-truth images

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

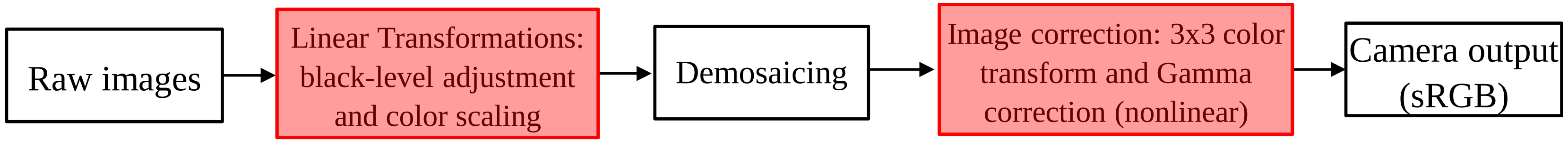
Creating ground-truth images

- 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

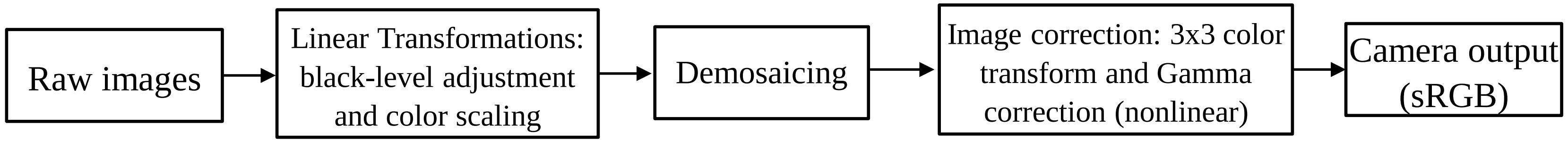
- Collected 500 raw images captured by Panasonic-Lumix LX-3
- We need to be able to go along a good camera pipeline
- We use **dcraw** to transfer images to different stages in camera pipeline



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

Creating ground-truth images

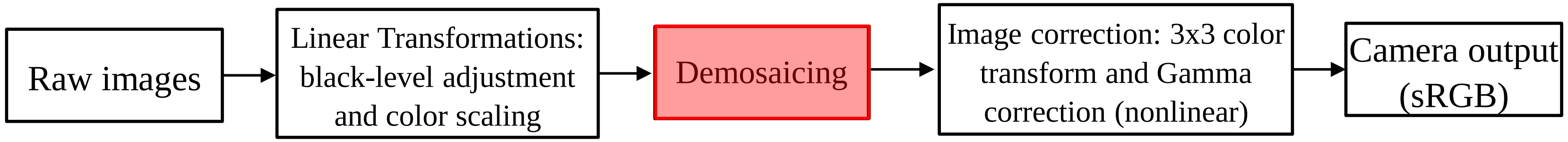
- Collected 500 raw images captured by Panasonic-Lumix LX-3
- We need to be able to go along a good camera pipeline
- We use **dcraw** to transfer images to different stages in camera pipeline



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

Creating ground-truth images

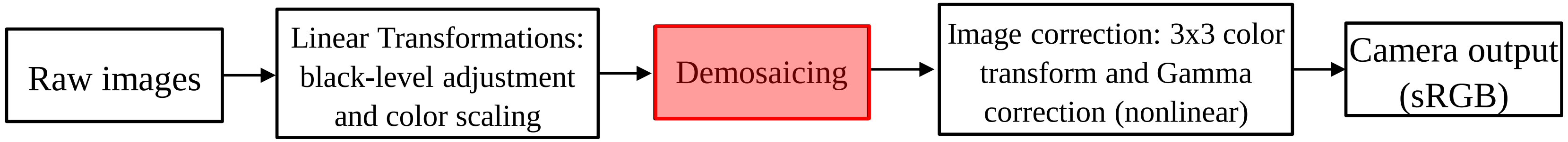
- Collected 500 raw images captured by Panasonic-Lumix LX-3
- We need to be able to go along a good camera pipeline
- We use **dcraw** to transfer images to different stages in camera pipeline



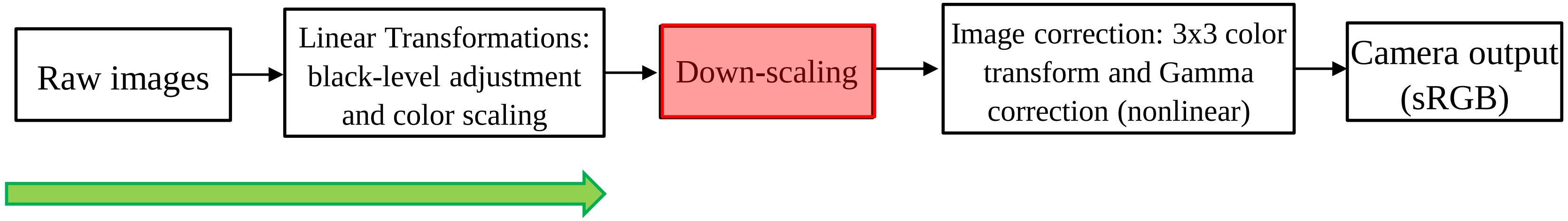
- We write out the **specifications of each step**, for each image
- Then simulate our own pipeline in MATLAB, with **demosaicing** replaced with our own **down-scaling**

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

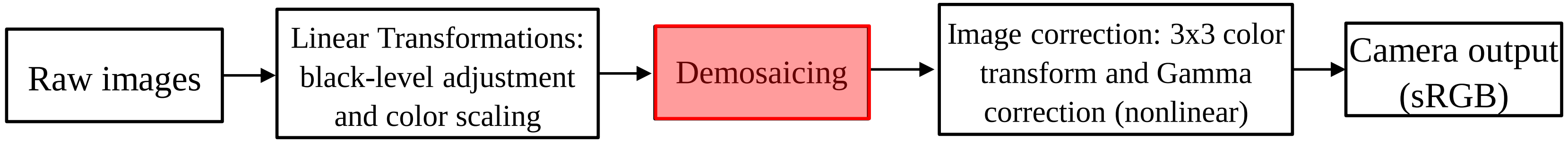


- We write out the **specifications of each step**, for each image
- Then simulate our own pipeline in MATLAB, with **demosaicing** replaced with our own **down-scaling**

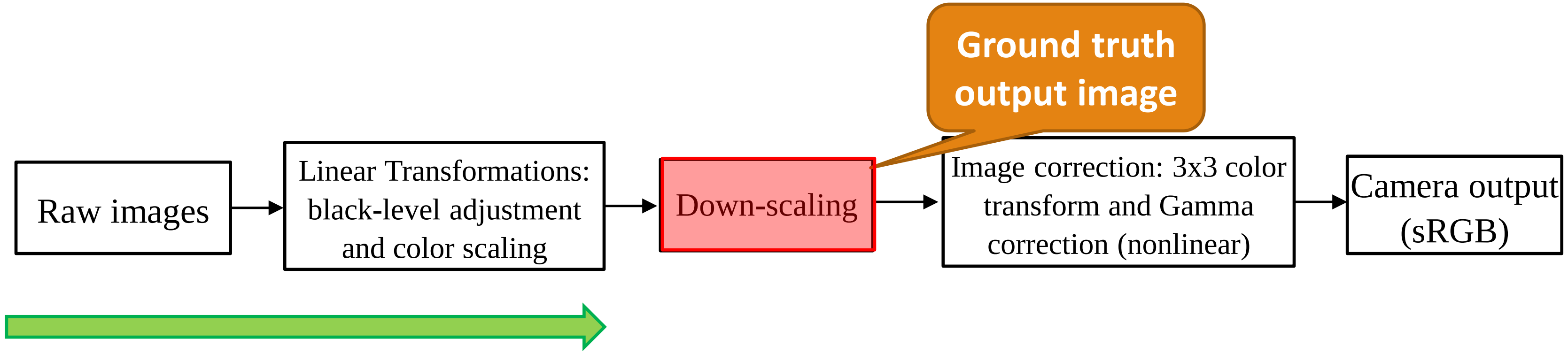


Creating ground-truth images

- 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

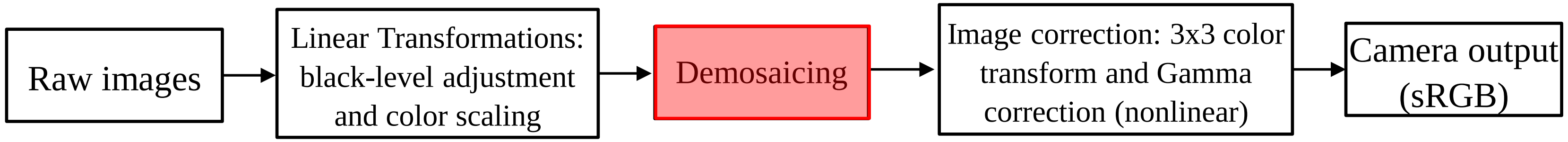


- We write out the **specifications of each step**, for each image
- Then simulate our own pipeline in MATLAB, with **demosaicing** replaced with our own **down-scaling**

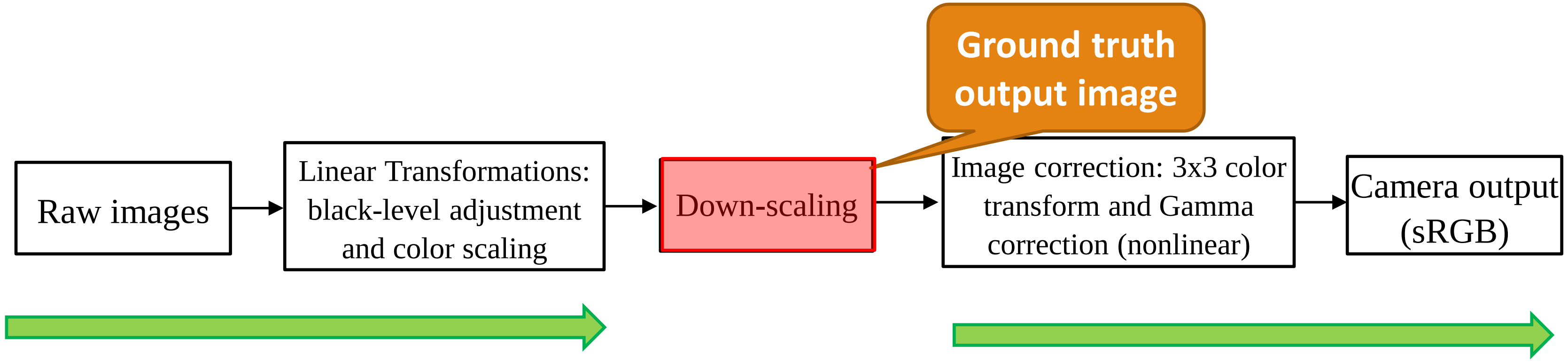


Creating ground-truth images

- 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

$$z(x) = y(x) + \eta_{\mathrm{p}}(y(x)) + \eta_{\mathrm{g}}(x)$$

$$\chi\left(y(x) + \eta_{\mathrm{p}}(y(x))\right) \sim \mathcal{P}(\chi y(x)), \quad \eta_{\mathrm{g}}(x) \sim \mathcal{N}(0, b),$$

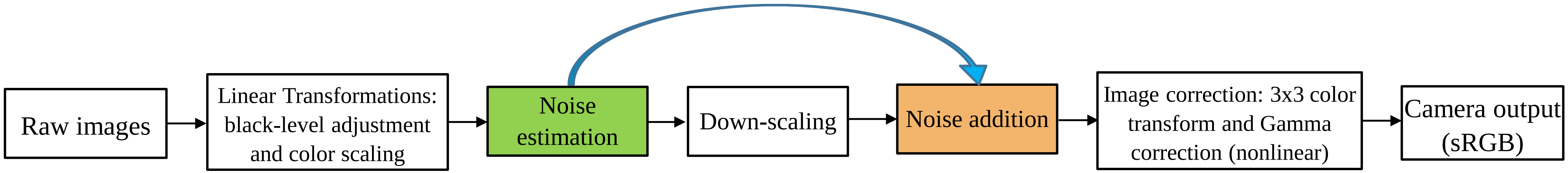
[1] Foi, Alessandro, et al. "Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data." *Image Processing, IEEE Transactions on* 17.10 (2008): 1737-1754.

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

$$z(x) = y(x) + \eta_{\mathrm{p}}(y(x)) + \eta_{\mathrm{g}}(x)$$
$$\chi\left(y(x) + \eta_{\mathrm{p}}(y(x))\right) \sim \mathcal{P}(\chi y(x)), \quad \eta_{\mathrm{g}}(x) \sim \mathcal{N}(0, b),$$

- We then add back the noise into images



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

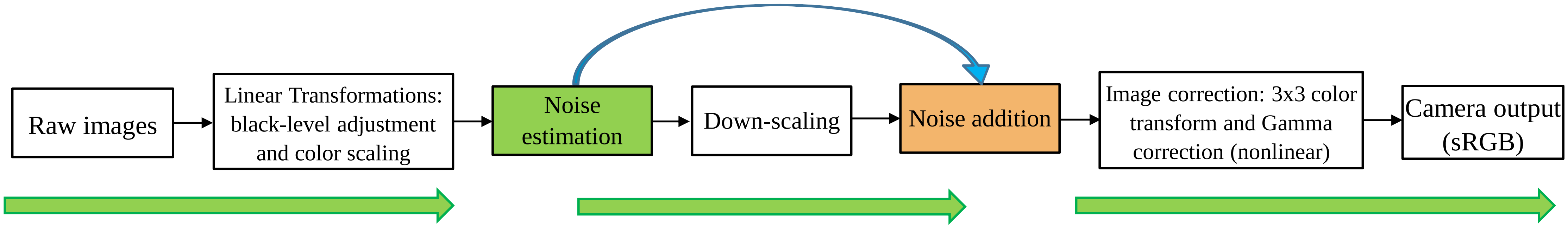
The noise behaviour in down-scaled images

- 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

$$z(x) = y(x) + \eta_{\mathrm{p}}(y(x)) + \eta_{\mathrm{g}}(x)$$

$$\chi\left(y(x) + \eta_{\mathrm{p}}(y(x))\right) \sim \mathcal{P}(\chi y(x)), \quad \eta_{\mathrm{g}}(x) \sim \mathcal{N}(0, b),$$

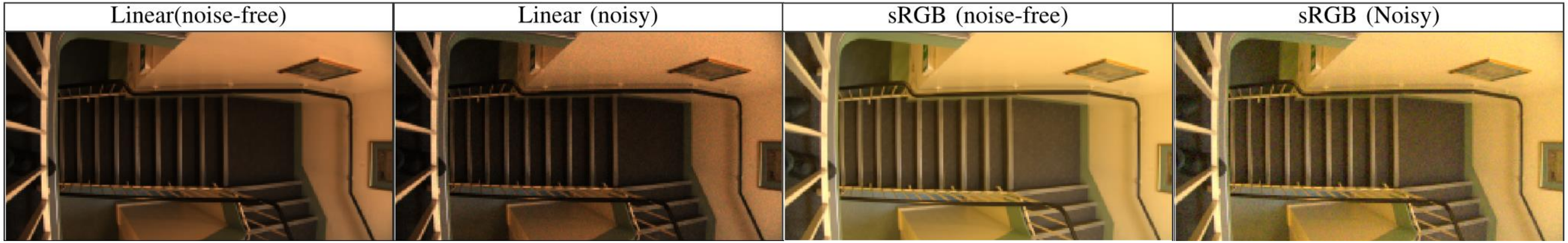
- We then add back the noise into images



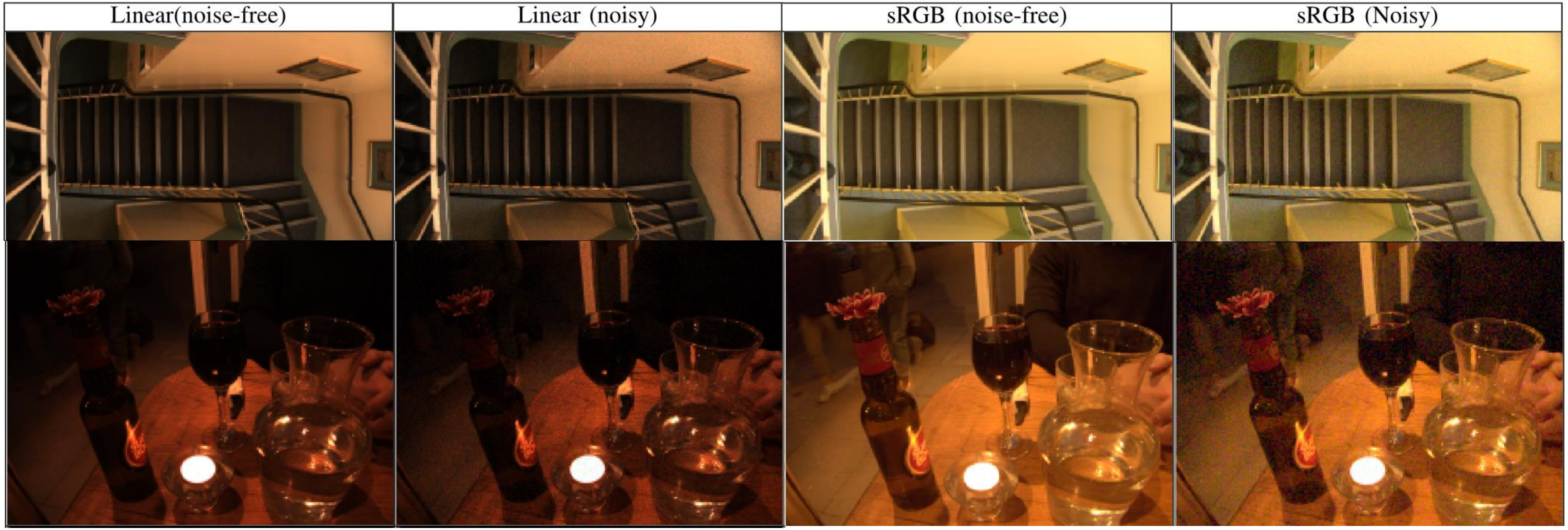
[1] Foi, Alessandro, et al. "Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data." *Image Processing, IEEE Transactions on* 17.10 (2008): 1737-1754.

Samples of noise-addition to image

Samples of noise-addition to image



Samples of noise-addition to image



Samples of noise-addition to image

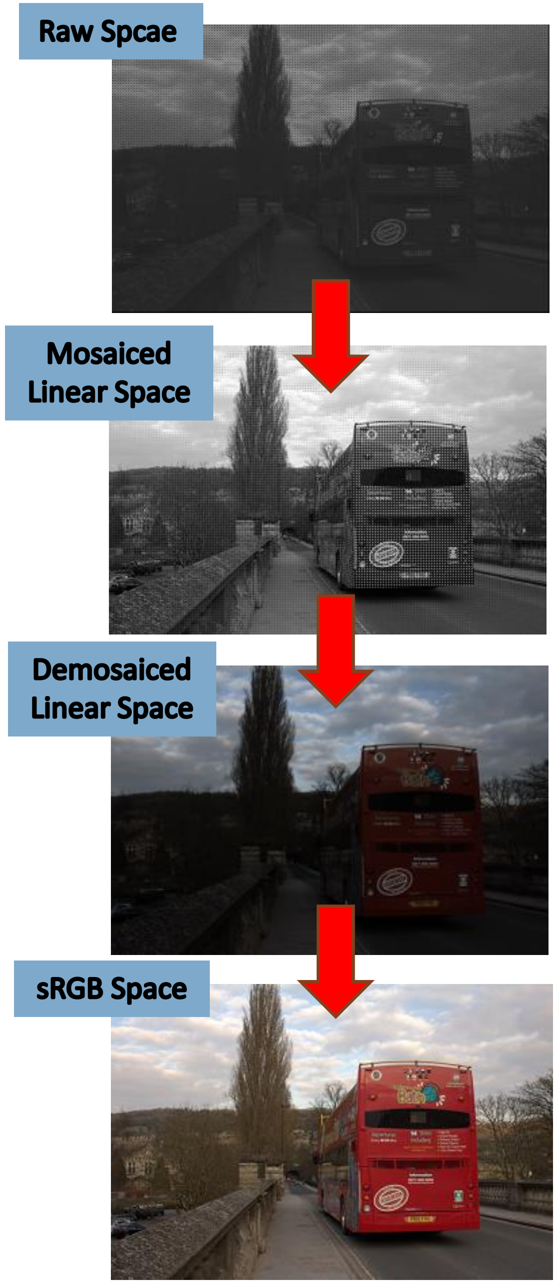
Linear(noise-free)	Linear (noisy)	sRGB (noise-free)	sRGB (Noisy)

Possible tasks

- We have developed these images

Raw images

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

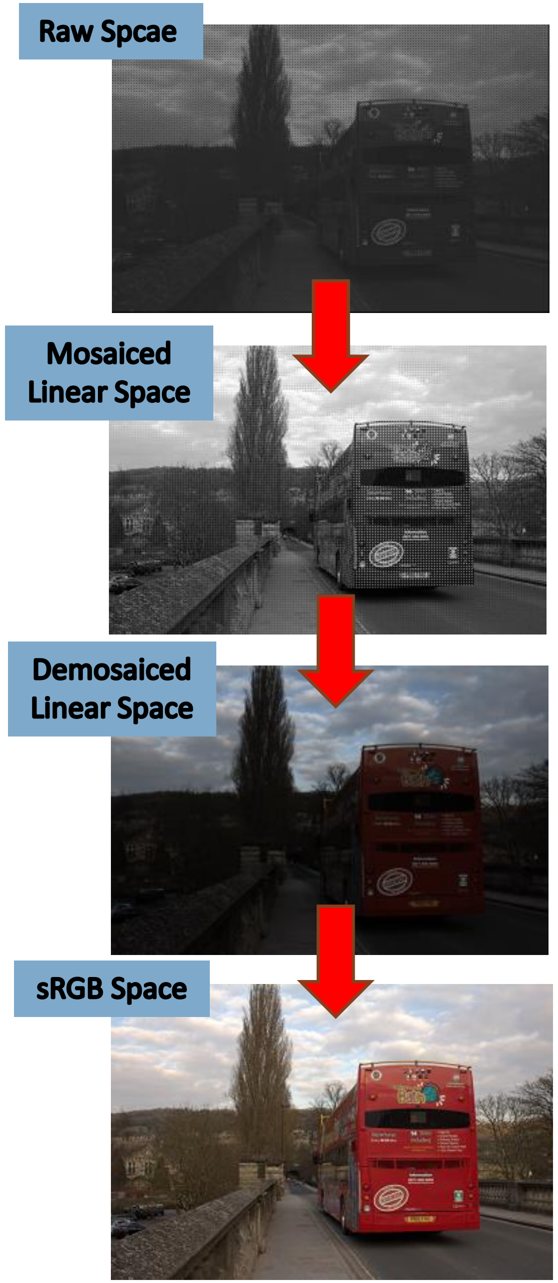


Possible tasks

- We have developed these images

Raw images → Noise-free-linear

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



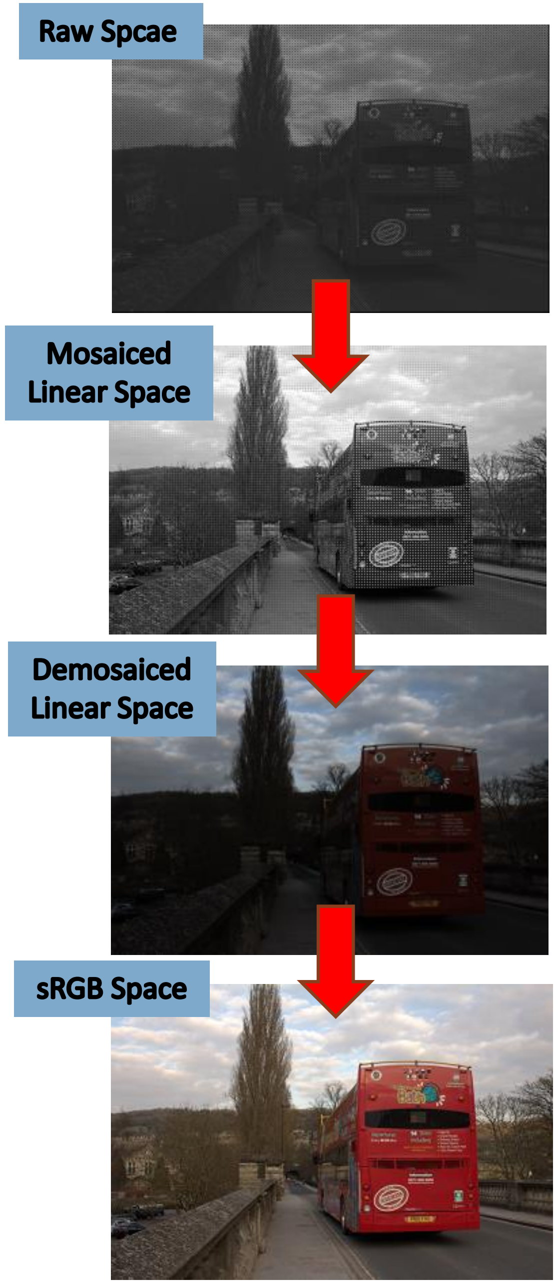
Possible tasks

- We have developed these images

Raw images → Noise-free-linear → Noise-free-sRGB

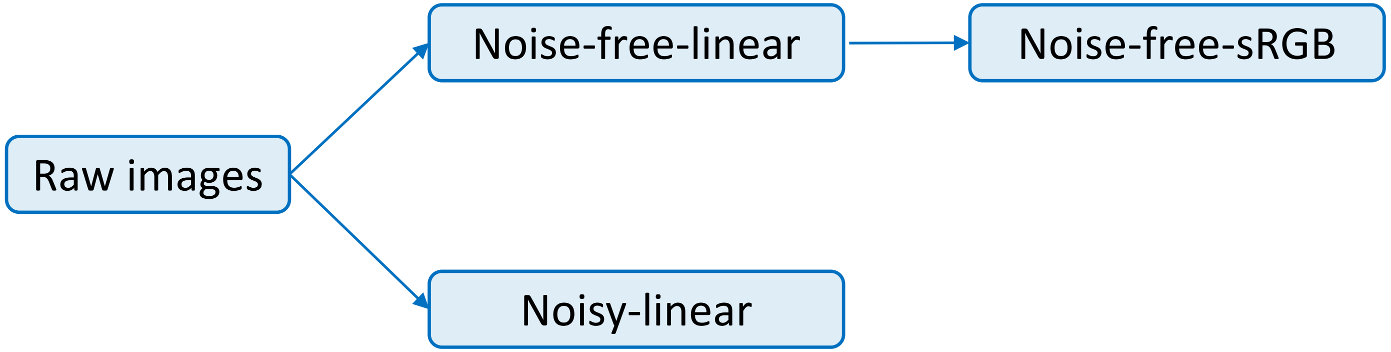
Raw images →

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

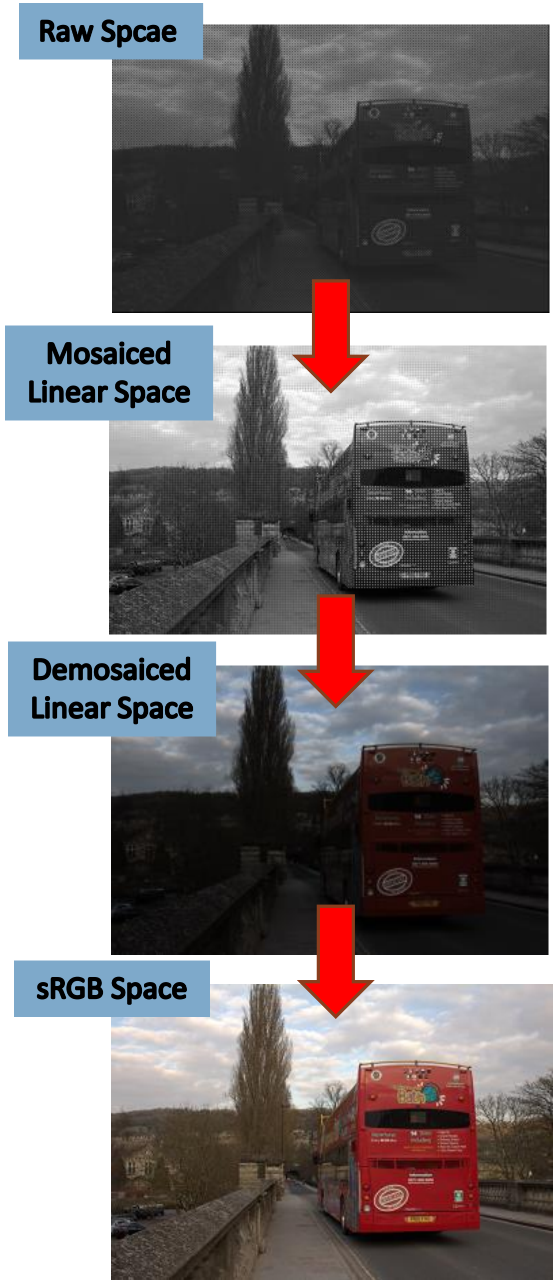


Possible tasks

- We have developed these images

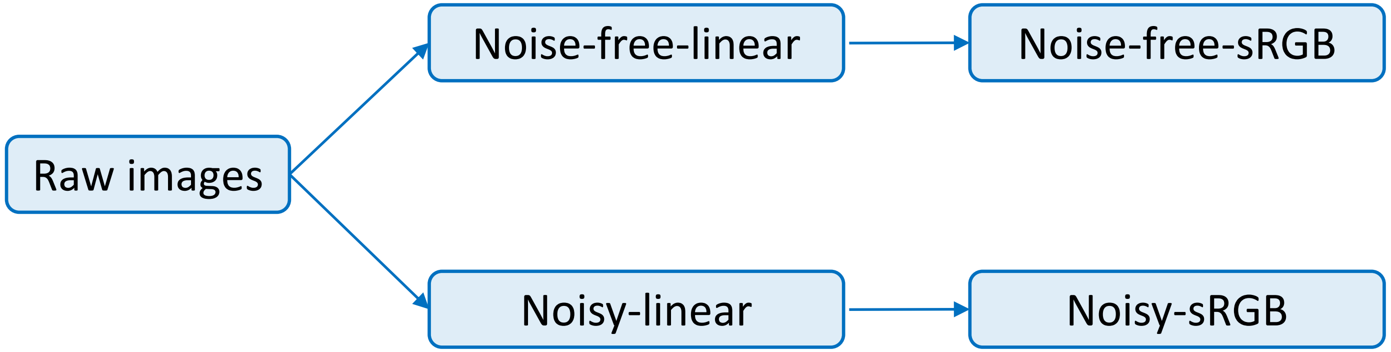


- Among the above tasks we are mostly interested in doing the followings:
  - Demosaicing in linear light-space
  - Deniosing +demosaicing in linear light-spcae

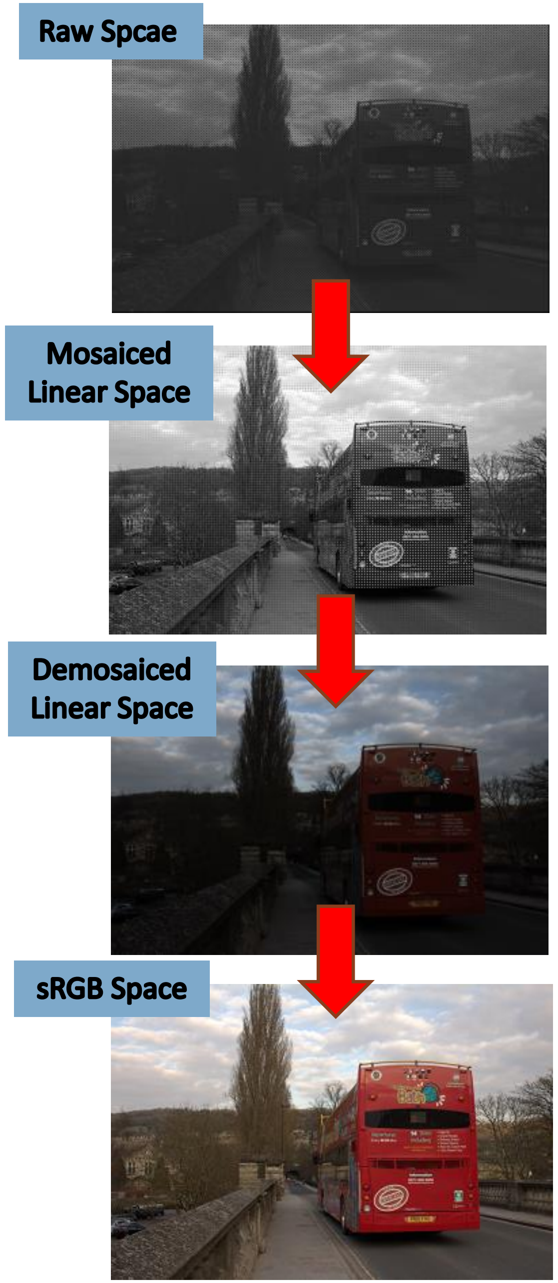


Possible tasks

- We have developed these images

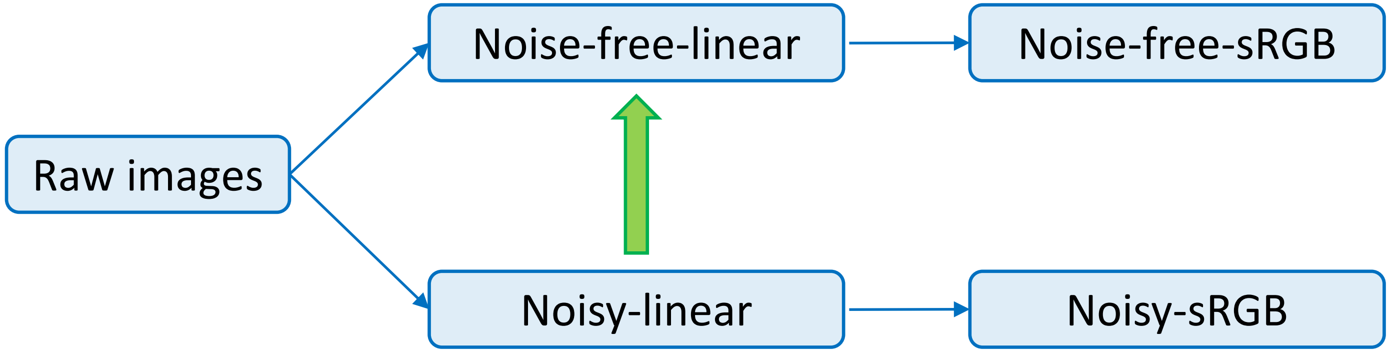


- Among the above tasks we are mostly interested in doing the followings:
  - Demosaicing in linear light-space
  - Deniosing +demosaicing in linear light-spcae

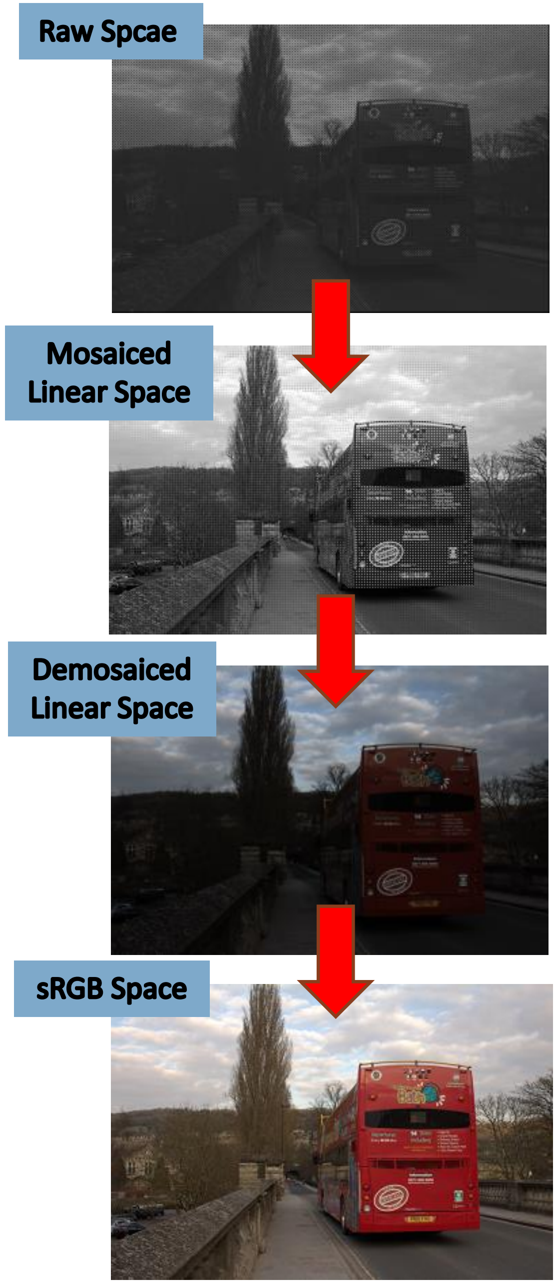


Possible tasks

- We have developed these images

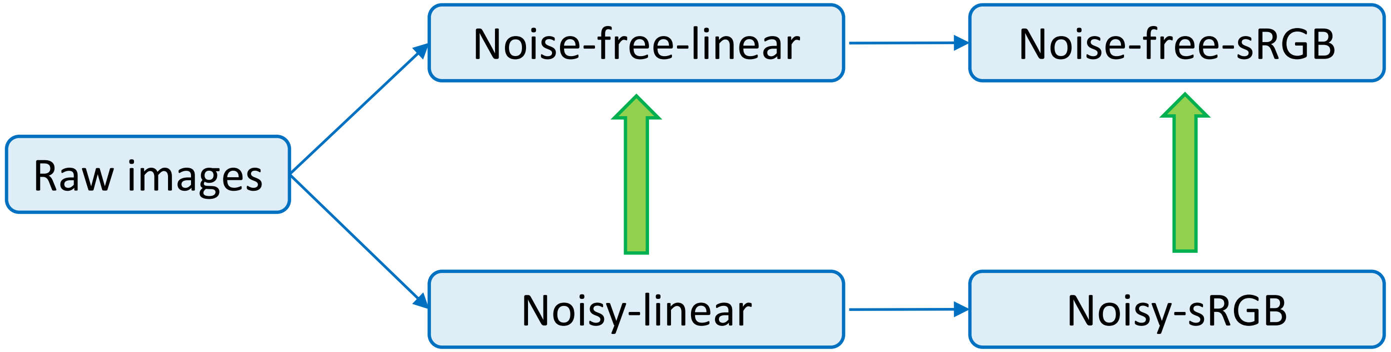


- Among the above tasks we are mostly interested in doing the followings:
  - Demosaicing in linear light-space
  - Deniosing +demosaicing in linear light-spcae

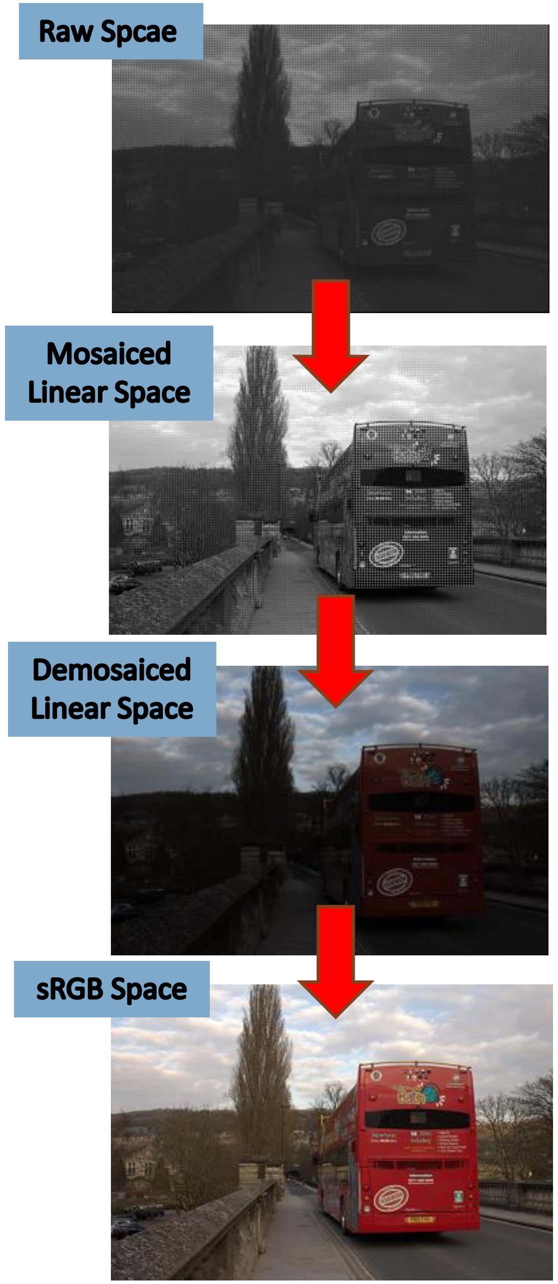


Possible tasks

- We have developed these images

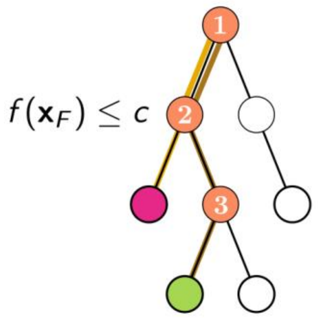


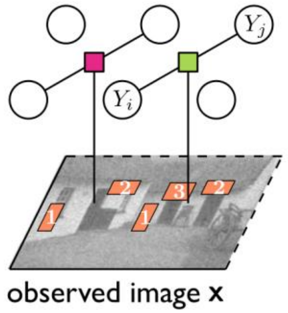
- Among the above tasks we are mostly interested in doing the followings:
  - Demosaicing in linear light-space
  - Deniosing +demosaicing in linear light-spcae



Regression tree field

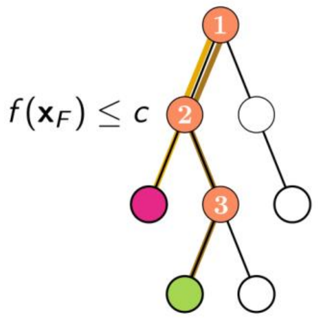
- Combines **regression tress (Non-parametric)** and **Gaussian Random Fields**

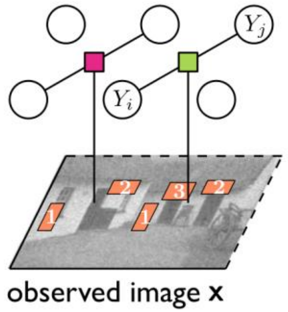




Regression tree field

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





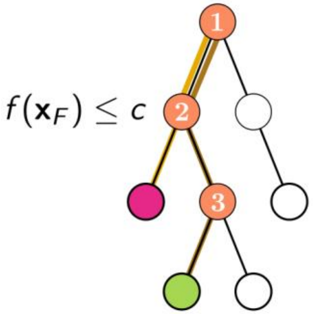
- Energy defined for each factor:

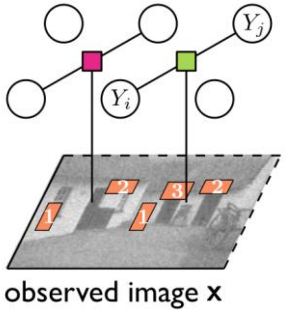
$$E(\mathbf{y}_F \mid \mathbf{x}_F) = \frac{1}{2}\mathbf{y}_F^{\mathsf{T}}\mathbf{Q}(\mathbf{x}_F)\mathbf{y}_F - \mathbf{y}^{\mathsf{T}}\mathbf{L}(\mathbf{x}_F)\mathbf{b}(\mathbf{x}_F)$$

  - Measure of the goodness of particular labelling given inputs, and parameters of the factor

Regression tree field

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



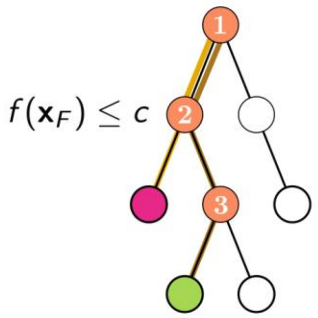


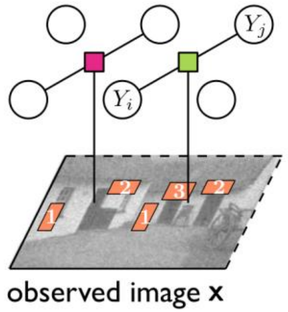
$$E(\mathbf{y}_F \mid \mathbf{x}_F) = \frac{1}{2}\mathbf{y}_F^T \mathbf{Q}(\mathbf{x}_F)\mathbf{y}_F - \mathbf{y}^T \mathbf{L}(\mathbf{x}_F)\mathbf{b}(\mathbf{x}_F)$$

- 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.

Regression tree field

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



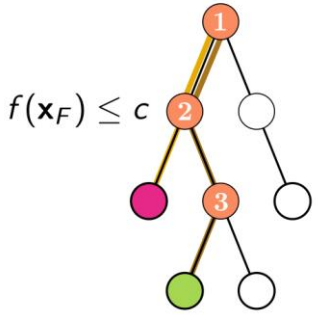


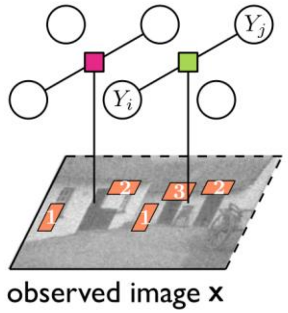
- Energy defined for each factor: $E(\mathbf{y}_F \mid \mathbf{x}_F) = \frac{1}{2}\mathbf{y}_F^{\mathsf{T}}\mathbf{Q}(\mathbf{x}_F)\mathbf{y}_F - \mathbf{y}^{\mathsf{T}}\mathbf{L}(\mathbf{x}_F)\mathbf{b}(\mathbf{x}_F)$
  - Measure of the goodness of particular labelling given inputs, and parameters of the factor
  - Coefficients of are determined by the **regression tree**, or each leave stores set of parameters.
- **Inference**:

$$p(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) \propto \exp[-E(\mathbf{y} \mid \mathbf{x}; \mathbf{w})] \quad \Rightarrow \quad \hat{\mathbf{y}}(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y}}\, p(\mathbf{y} \mid \mathbf{x}) = \boldsymbol{\mu} = [\mathbf{Q}(\mathbf{x}; \mathbf{w})]^{-1}\mathbf{l}(\mathbf{x}; \mathbf{w})$$

Regression tree field

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



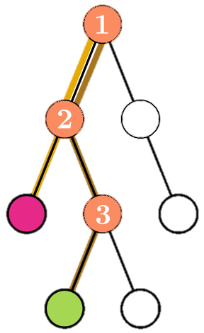


- Energy defined for each factor: $$E(\mathbf{y}_F \mid \mathbf{x}_F) = \frac{1}{2}\mathbf{y}_F^{\mathsf{T}}\mathbf{Q}(\mathbf{x}_F)\mathbf{y}_F - \mathbf{y}^{\mathsf{T}}\mathbf{L}(\mathbf{x}_F)\mathbf{b}(\mathbf{x}_F)$$
  - Measure of the goodness of particular labelling given inputs, and parameters of the factor
  - Coefficients of are determined by the **regression tree**, or each 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:

$$p(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) \propto \exp[-E(\mathbf{y} \mid \mathbf{x}; \mathbf{w})] \quad \Longrightarrow \quad \hat{\mathbf{y}}(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y}}\, p(\mathbf{y} \mid \mathbf{x}) = \boldsymbol{\mu} = [\mathbf{Q}(\mathbf{x}; \mathbf{w})]^{-1}\mathbf{l}(\mathbf{x}; \mathbf{w})$$

$$\frac{1}{N}\sum_{i}^{N} \ell(\hat{\mathbf{y}}(\mathbf{x}^{(i)}; \mathbf{w}), \mathbf{y}^{(i)}) \approx \mathbb{E}_{p(\mathbf{x},\mathbf{y})}\left[\ell(\hat{\mathbf{y}}(\mathbf{x}; \mathbf{w}), \mathbf{y})\right]$$

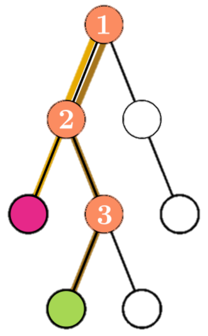
Regression tree fields, for demosaicing



[1] RFS filters: http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

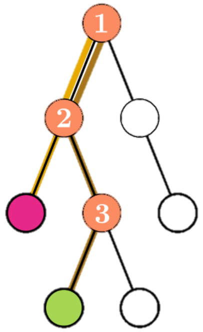
Regression tree fields, for demosaicing

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



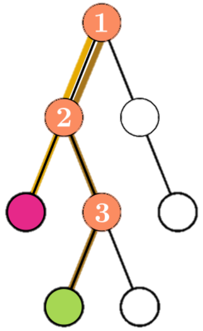
Regression tree fields, for demosaicing

- **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

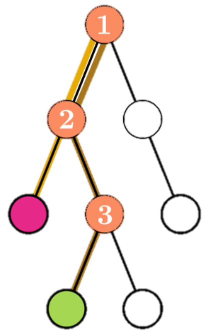
- **Tree Feature checks**,
  - A preliminary bilinear interpolation
  - RFS filters [1] which act like derivatives in different directions, with various scales
- Quadratic energy basis vectors,
  - Set of neightbouring pixels,



[1] RFS filters: http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

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 neightbouring pixels,
  - RFS filter responses



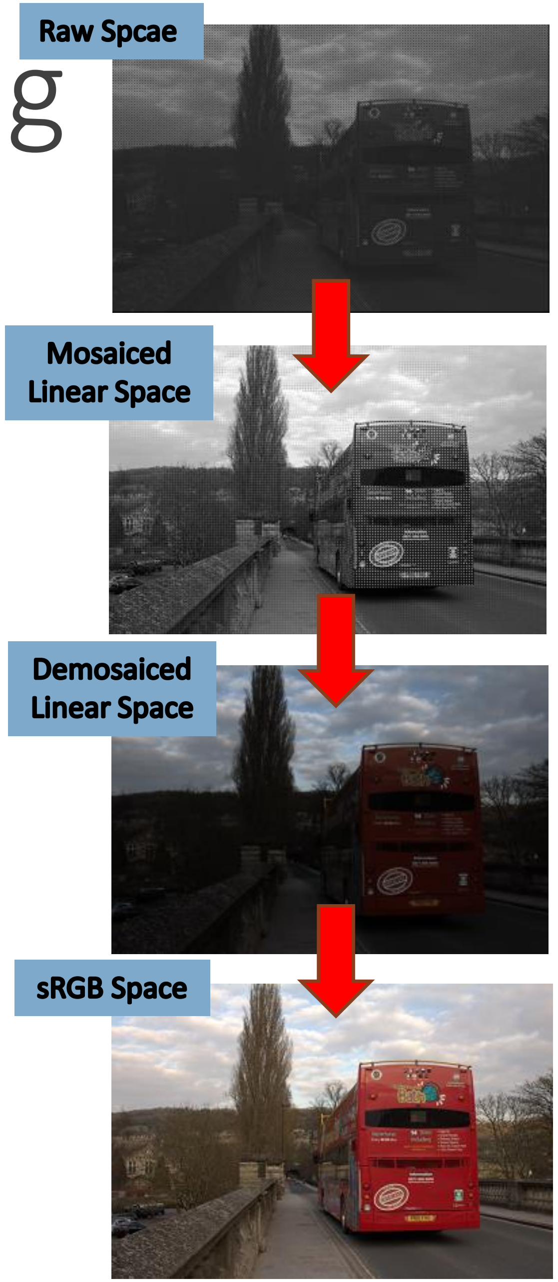
$$E(\mathbf{y}_F \mid \mathbf{x}_F) = \frac{1}{2}\mathbf{y}_F^{\mathsf{T}}\mathbf{Q}(\mathbf{x}_F)\mathbf{y}_F - \mathbf{y}^{\mathsf{T}}\mathbf{L}(\mathbf{x}_F)\mathbf{b}(\mathbf{x}_F)$$

Regression tree fields, for demosaicing

[1] RFS filters: http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

Regression tree fields, for demosaicing

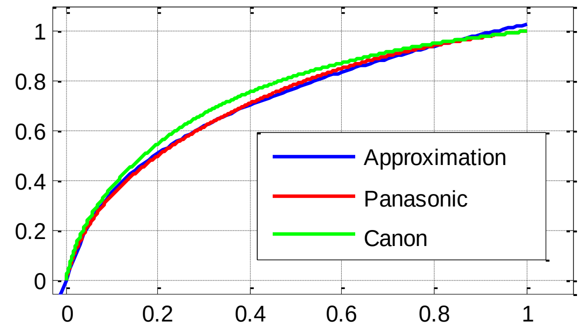
- The generalized loss function,

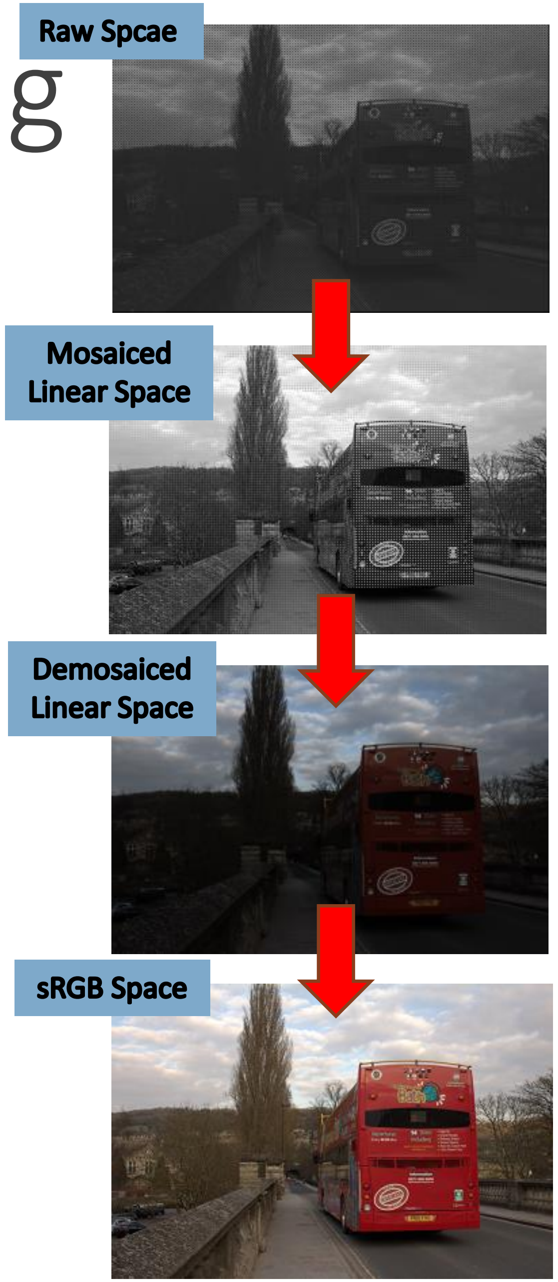


[1] RFS filters: http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

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

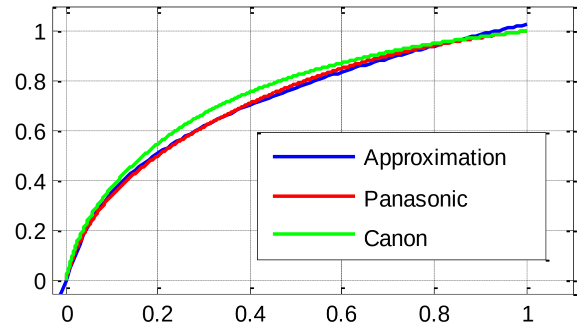


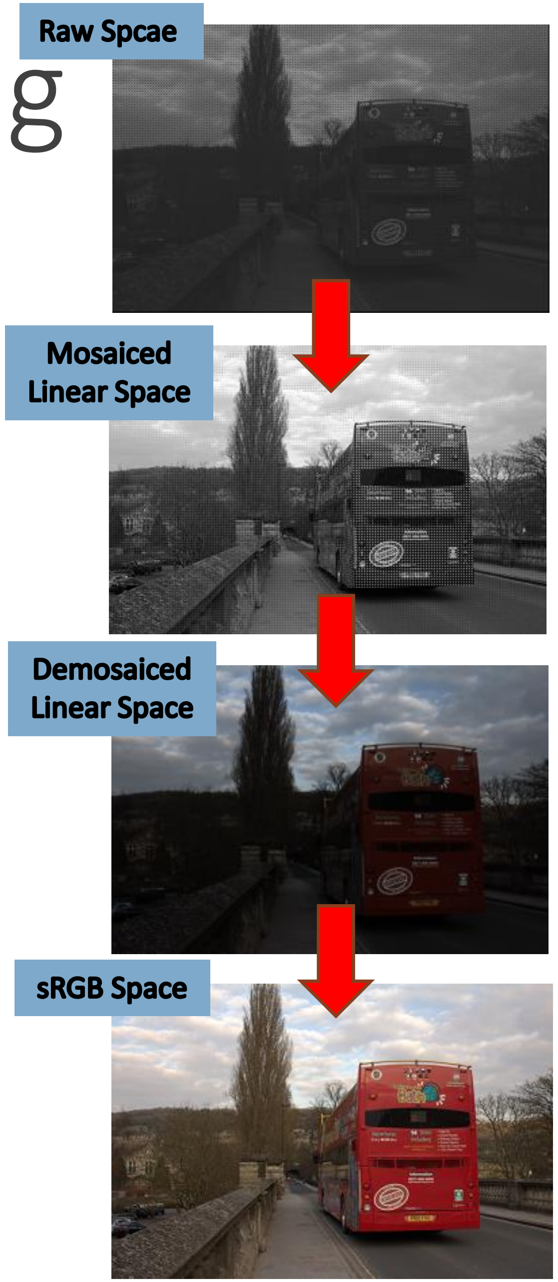


[1] RFS filters: http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

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,

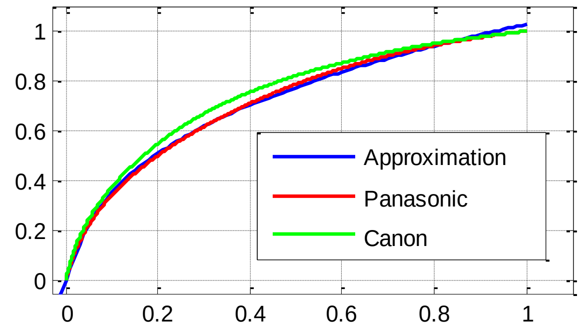


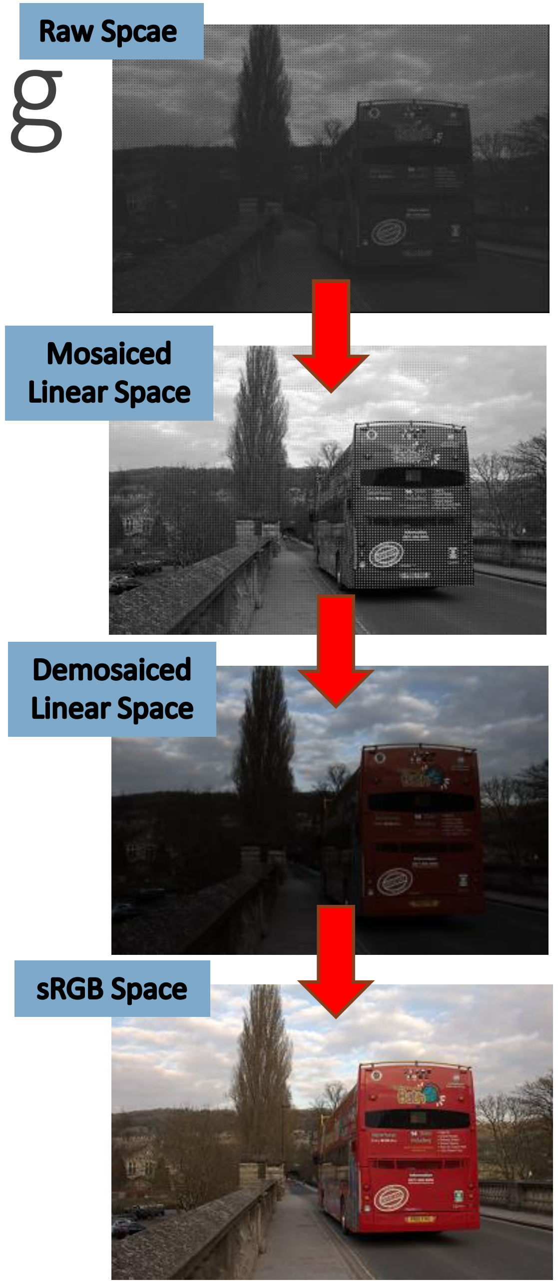


Regression tree fields, for demosaicing

- 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 3x3 color transform,

$$c_{\mathrm{MSE}}(\mathbf{I}, \tilde{\mathbf{I}}) = \sum_{i=\{R,G,B\}} \sum_{x,y} \left(I^i_{x,y} - \tilde{I}^i_{x,y} \right)^2.$$

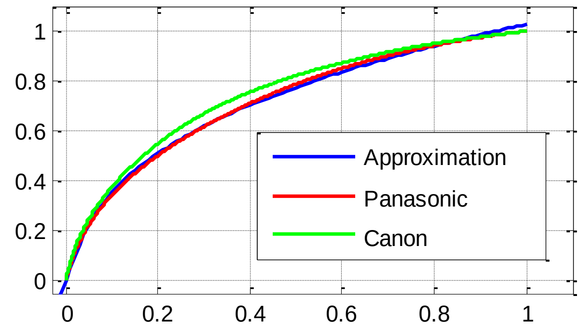




[1] RFS filters: http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

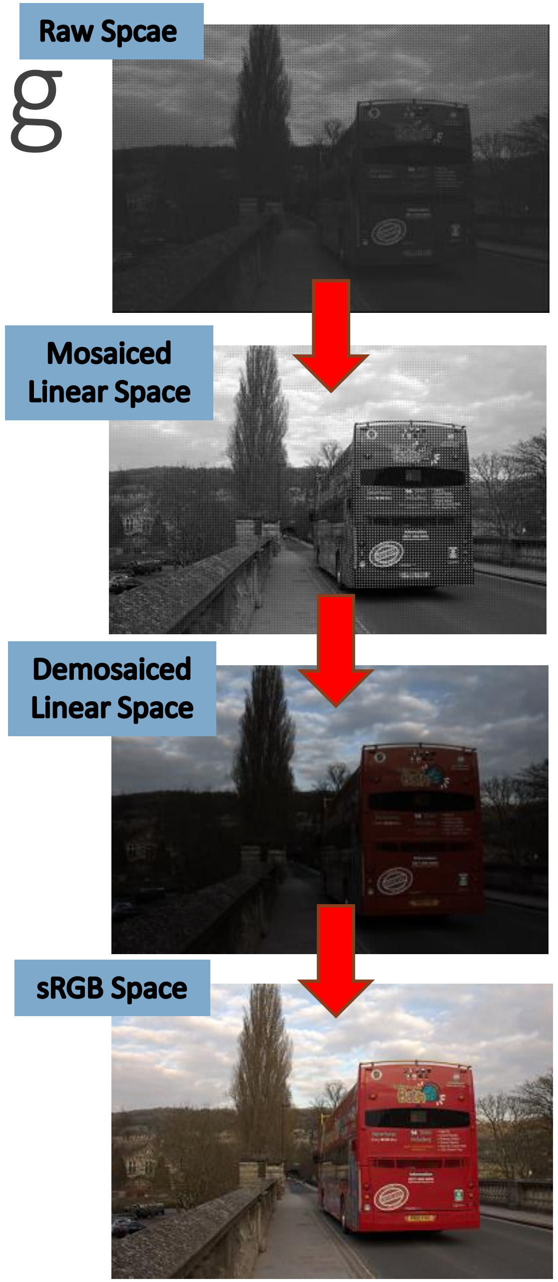
Regression tree fields, for demosaicing

- 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,



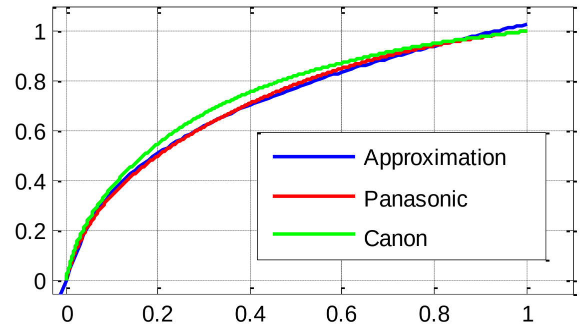
$$c_{\text{MSE}}(\mathbf{I},\tilde{\mathbf{I}}) = \sum_{i=\{R,G,B\}} \sum_{x,y} \left(I^i_{x,y} - \tilde{I}^i_{x,y}\right)^2.$$

$$f : \mathbf{L} \to \mathbf{I} \implies c_{\text{G-MSE}}(\mathbf{L},\tilde{\mathbf{L}}) \triangleq (c_{\text{MSE}} \circ f)(\mathbf{L},\tilde{\mathbf{L}}) = \sum_{i=\{R,G,B\}} \sum_{x,y} \left(f(L^i_{x,y}) - f(\tilde{L}^i_{x,y})\right)^2$$



Regression tree fields, for demosaicing

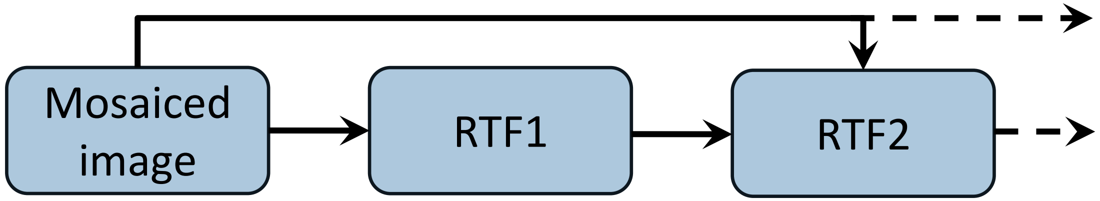
- The generalized loss function,
  - An approximate camera pipeline, from linear-space, to sRGB space, which is differentiable
    - An analytic approximation for gamma transform
    - An approximate 3x3 color transform,

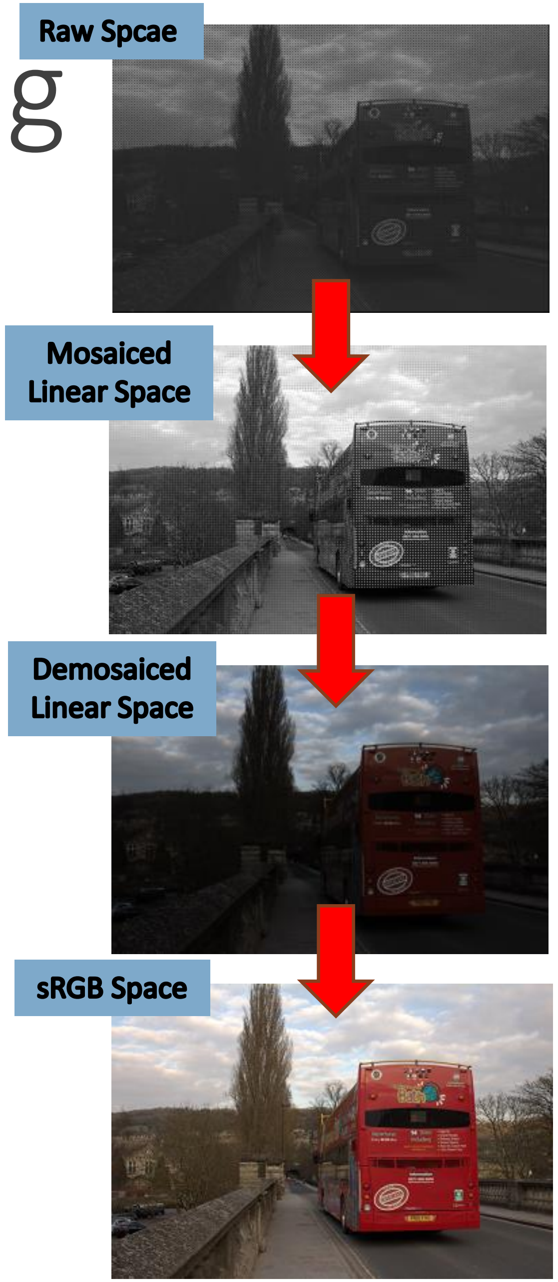


$$c_{\text{MSE}}(\mathbf{I}, \tilde{\mathbf{I}}) = \sum_{i=\{R,G,B\}} \sum_{x,y} \left(I^i_{x,y} - \tilde{I}^i_{x,y}\right)^2.$$

$$f : \mathbf{L} \to \mathbf{I} \implies c_{\text{G-MSE}}(\mathbf{L}, \tilde{\mathbf{L}}) \triangleq (c_{\text{MSE}} \circ f)(\mathbf{L}, \tilde{\mathbf{L}}) = \sum_{i=\{R,G,B\}} \sum_{x,y} \left(f(L^i_{x,y}) - f(\tilde{L}^i_{x,y})\right)^2$$

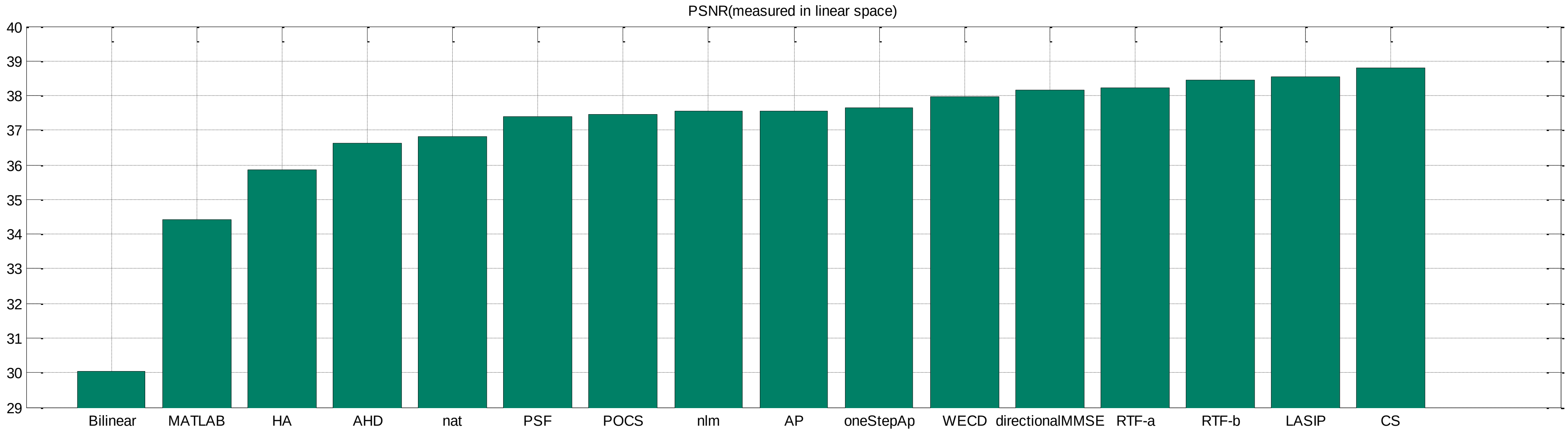
- Stacking RTFs,





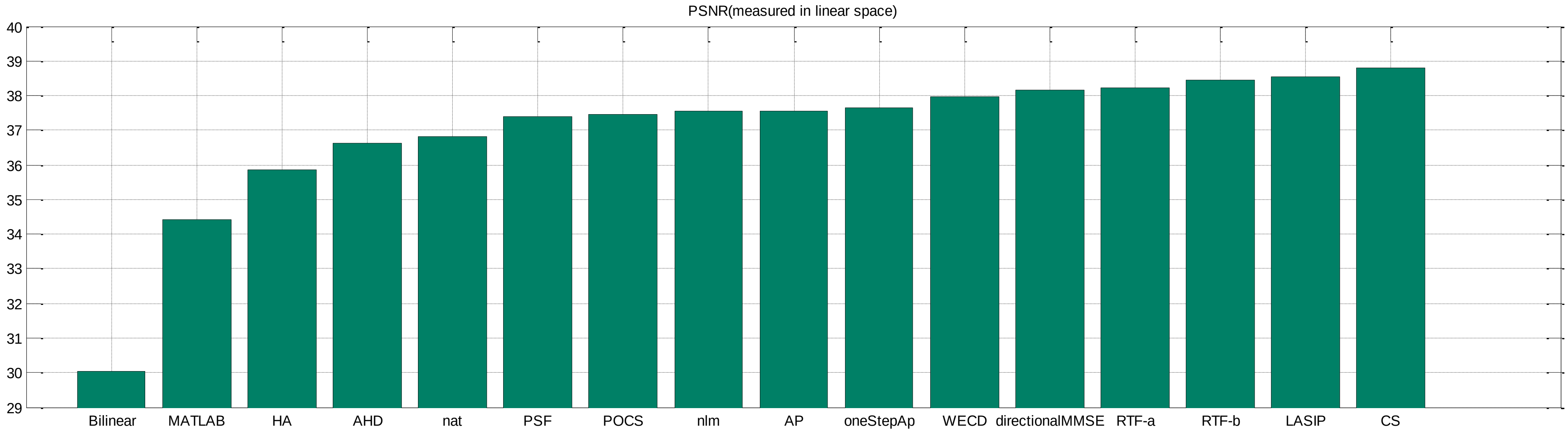
[1] RFS filters: http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

Experiments(1)

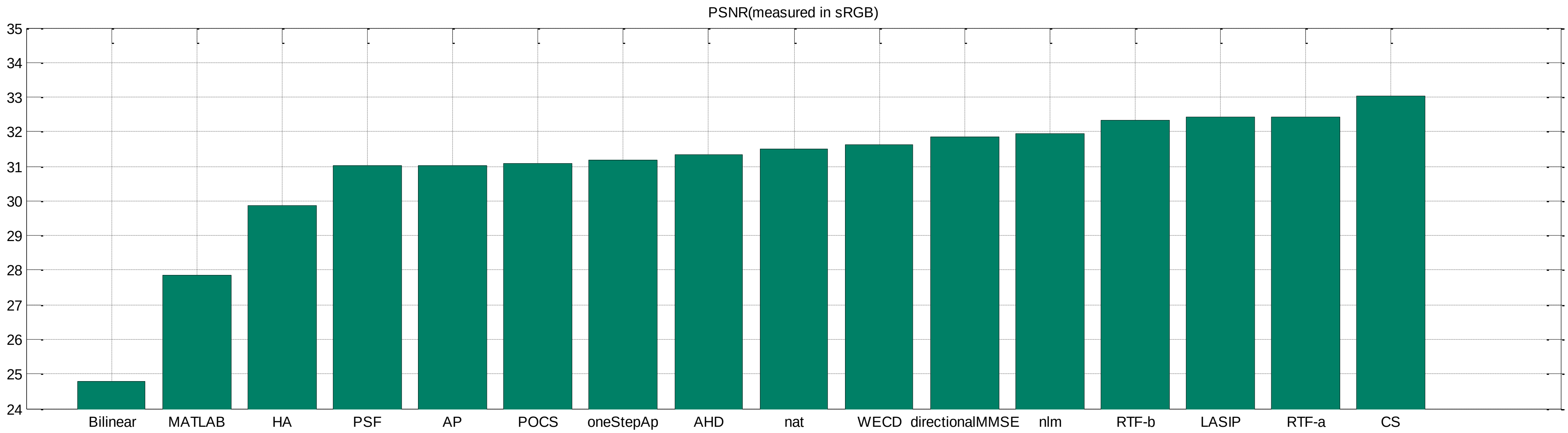


Experiments(1)

- **Demosaicing, in linear-space, without noise:**

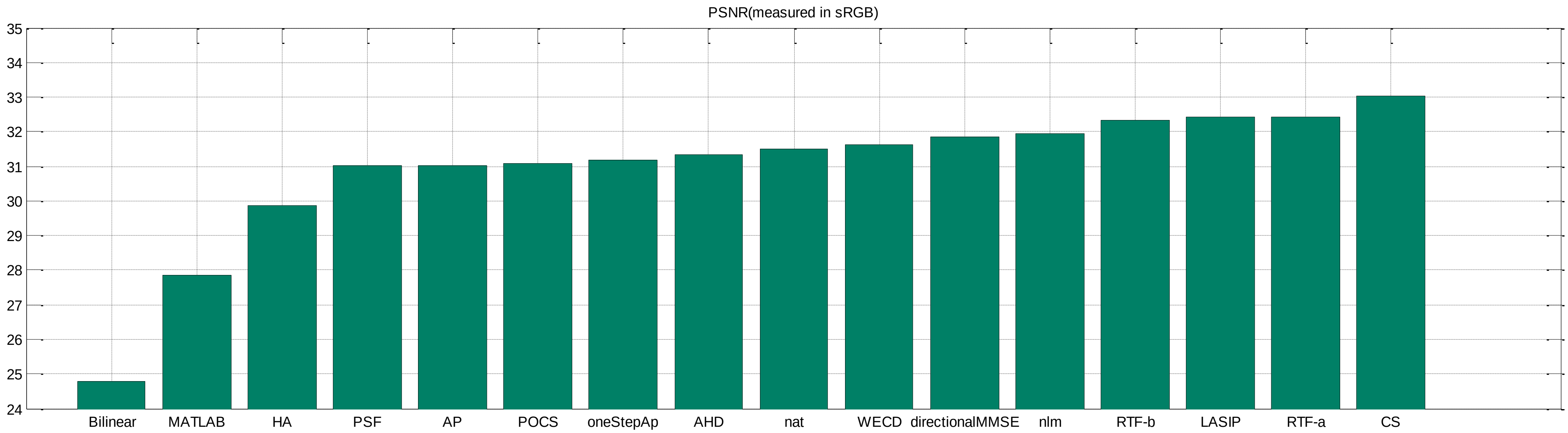


Experiments(2)



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

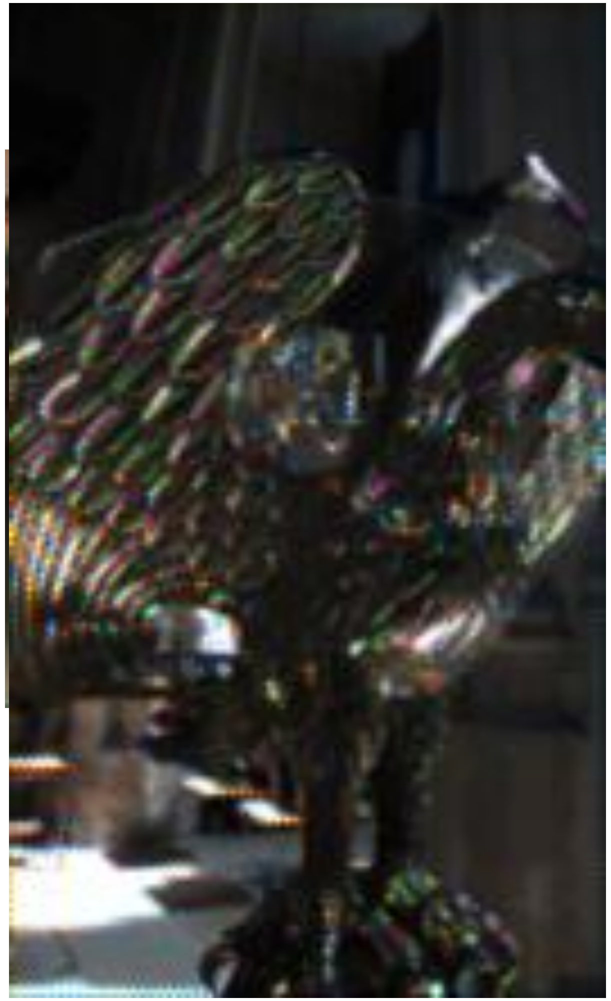
NAT: Non-Local Adaptive Thresholding:
NLM: Non-Local Means

Some samples

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

Some samples


Ground truth


Bilinear


CS


RTF

Image Margins




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

Image Margins

- Not a considerable margin in RTF outputs

NAT

NLM

OneStepAP

RTF

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-demosiacing experiments

And more to go ….

- RTF with the generalized loss function
- Full denoising-demosiacing experiments
- Other CFA patterns: Fuji-Xtrans pattern

And more to go ….

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