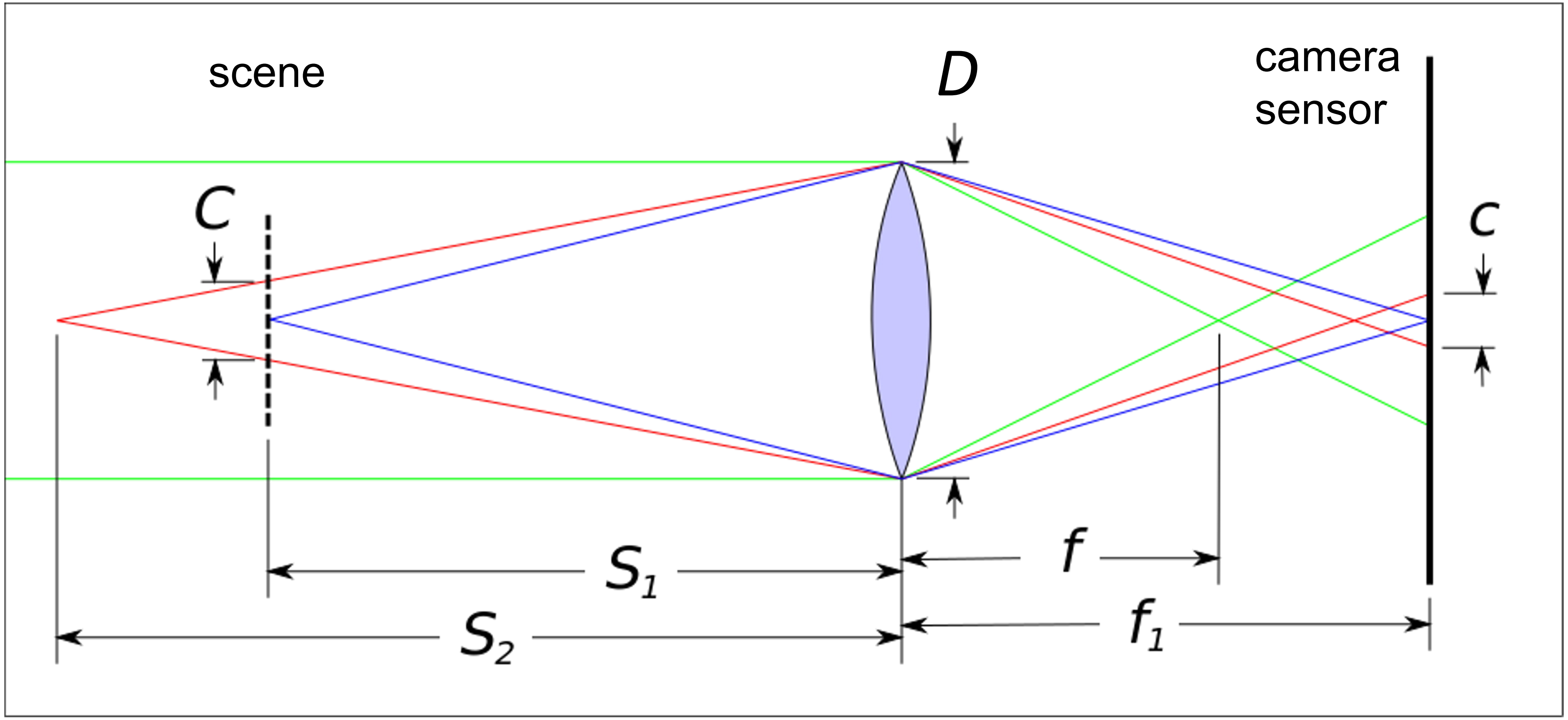
Problem Session 2

Topics

- Camera optics – concepts & intuition
  - F-number
  - Depth of Field (DoF)
  - Circle of confusion
- Image processing pipeline
  - Demosaicing: several methods
  - Gamma correction
- Denoising
  - Several methods

Camera optics – concepts & intuition

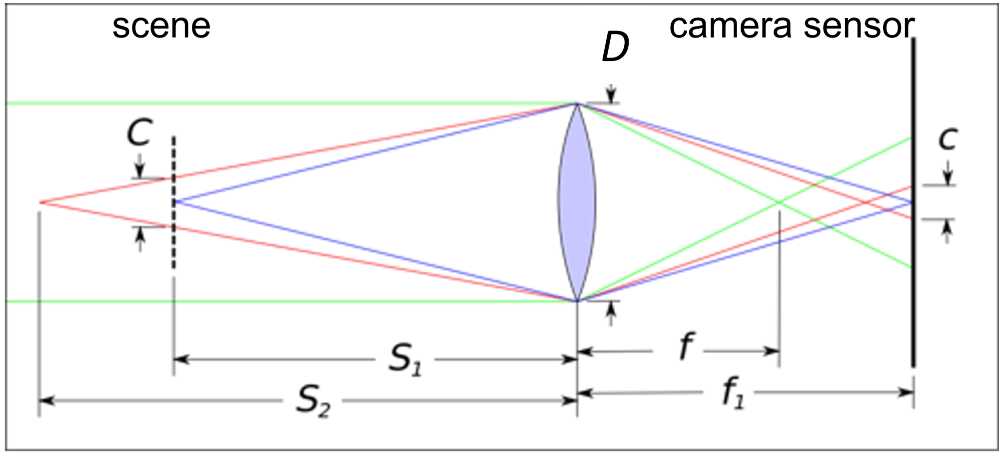


Task 1: Depth of field

f-number, N, is given by $N = \frac{f}{D}$, where f is the **focal length** and D is the diameter of the pupil (effective **aperture**). Written as f/N.

Magnification, $M = \frac{f_1}{S_1} = \frac{f}{S_1 - f}$, S_1 is the distance between lens and focal plane in the scene (not in the camera)

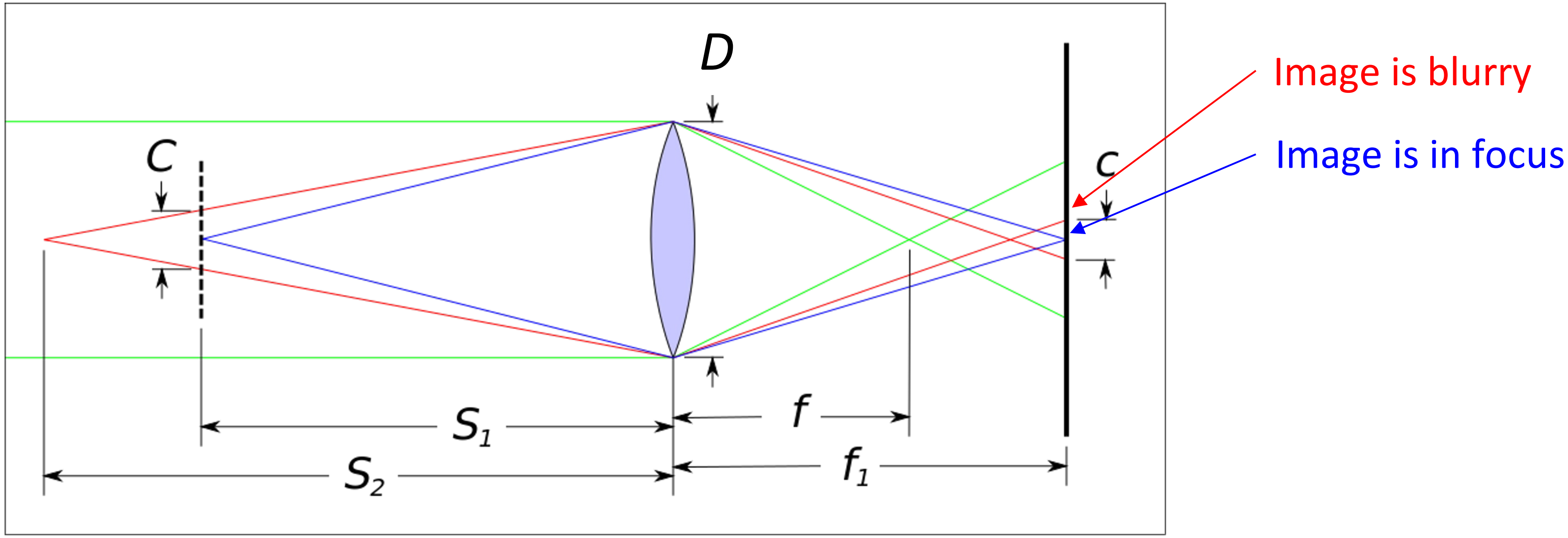
Lens equation: $\frac{1}{f} = \frac{1}{S_1} + \frac{1}{f_1}$



Task 1: Depth of field

Circle of confusion, $c = MD\frac{|S_2 - S_1|}{S_2}$

→ From this, calculate the near and far planes (S_N, S_F), which are at the edges of being in focus

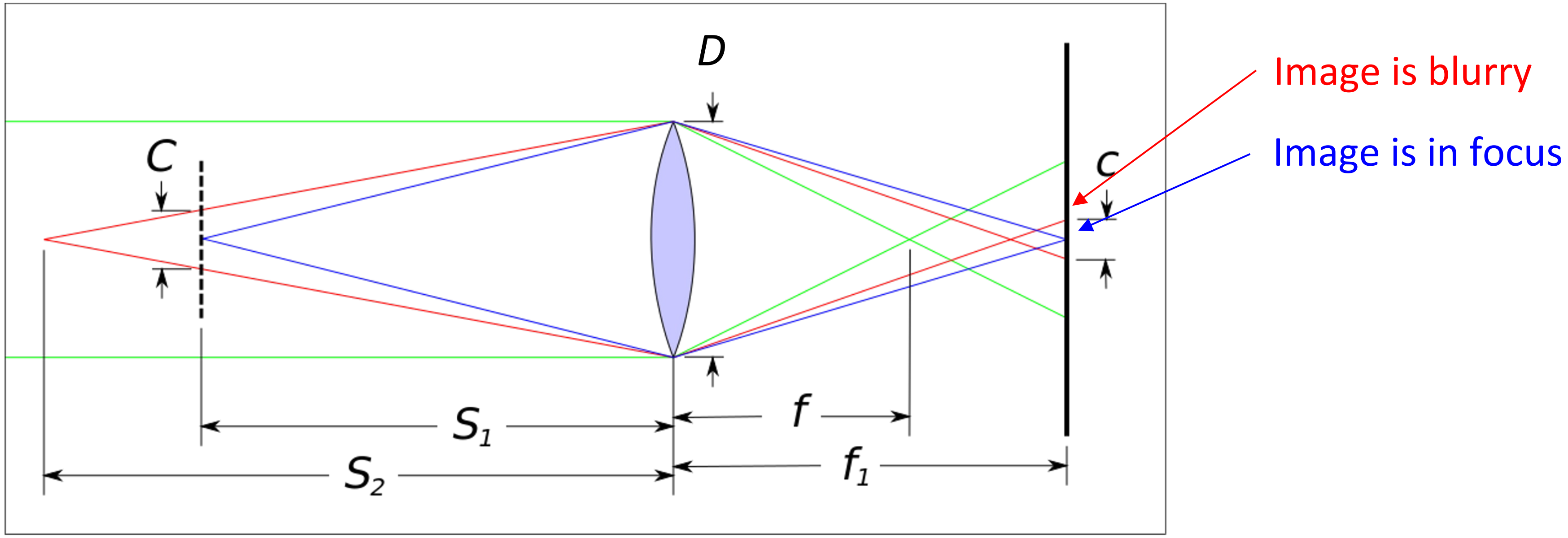


Task 1: Depth of field

Depth of field (DoF): the depth range around the focal plane of the camera that produces a circle of confusion with a diameter c.

$$\text{DoF} = S_F - S_N = \frac{2Nc(M+1)}{M^2 - \left(\frac{Nc}{f}\right)^2}$$

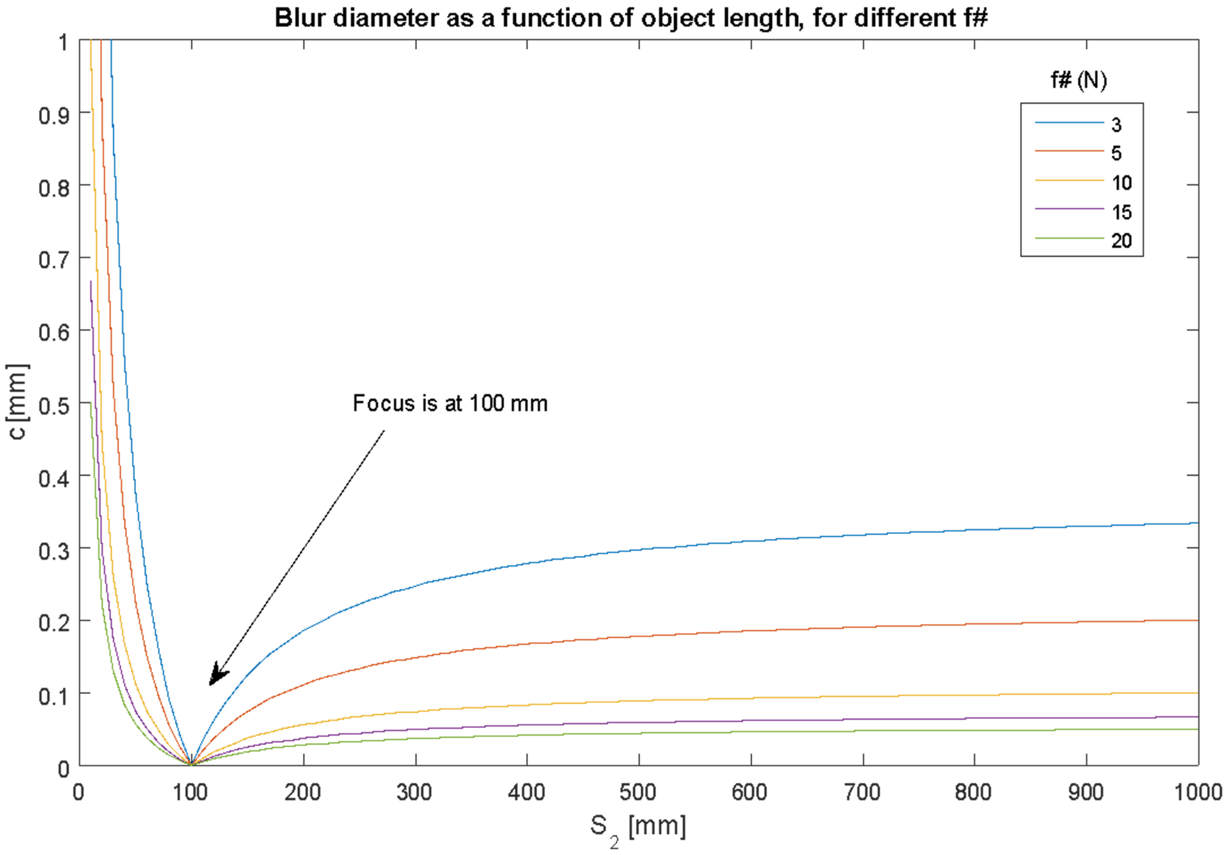
N is f-number of lens

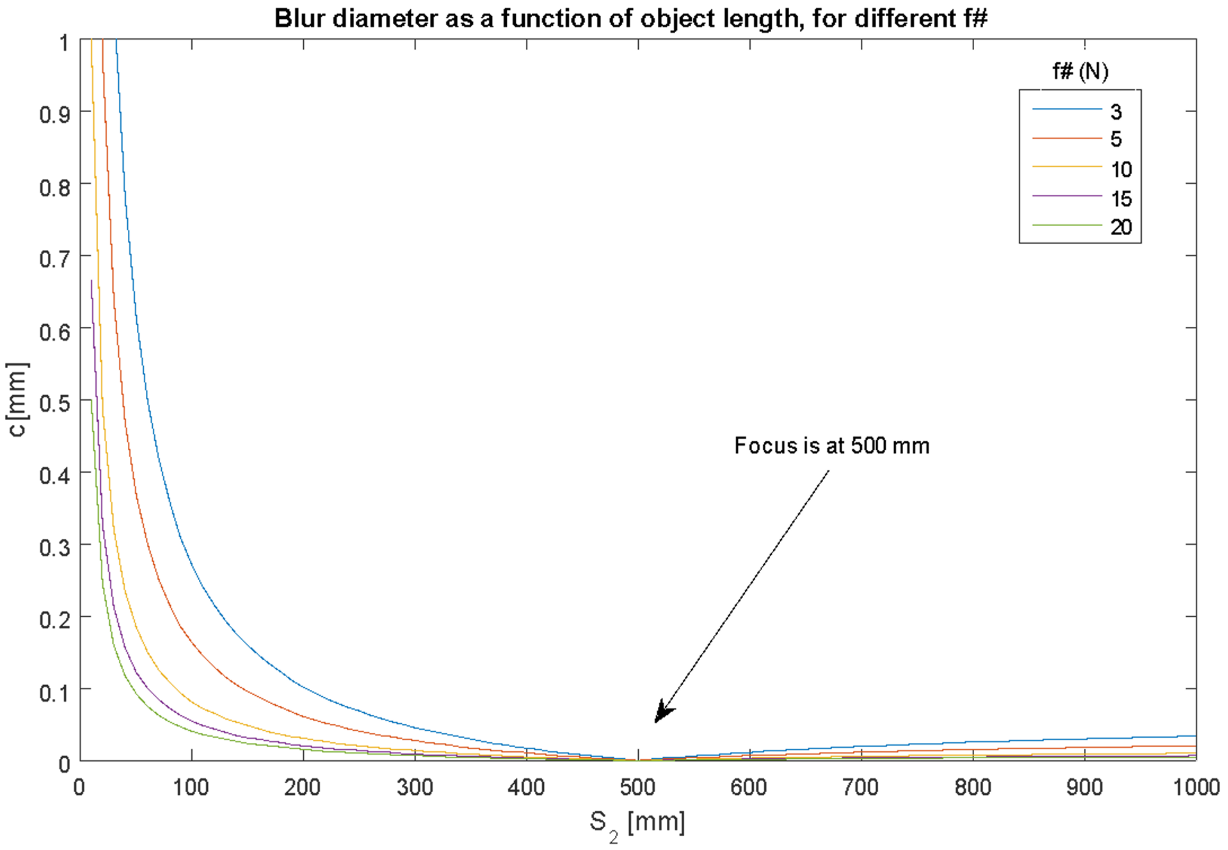


Task 1: Depth of field

So… why use small f#?

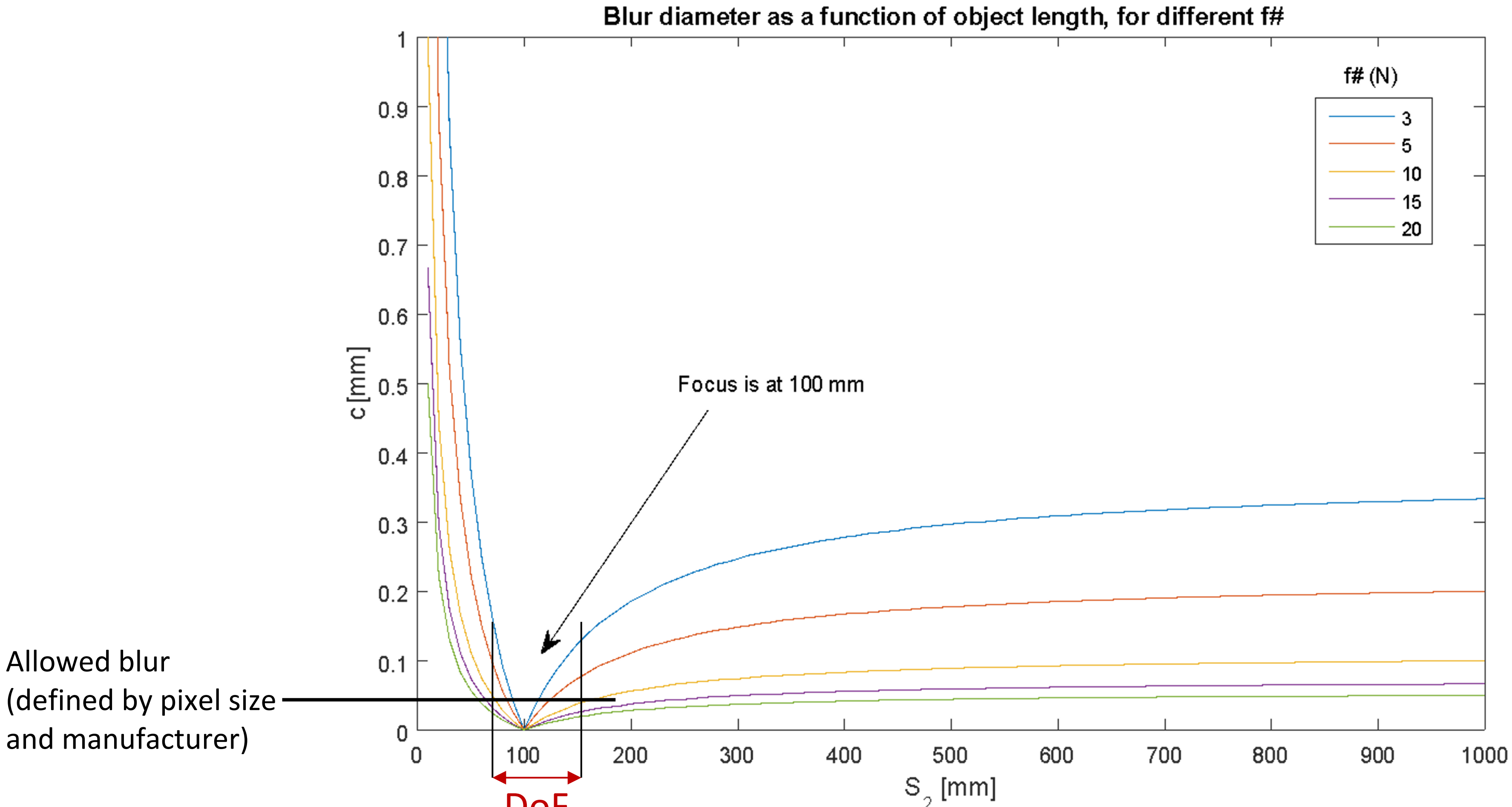
- Small f# (N) blur faster
- When focusing far, camera is more depth invariant (f is fixed)





Task 1: Depth of field

Using the graph, what is the DOF?

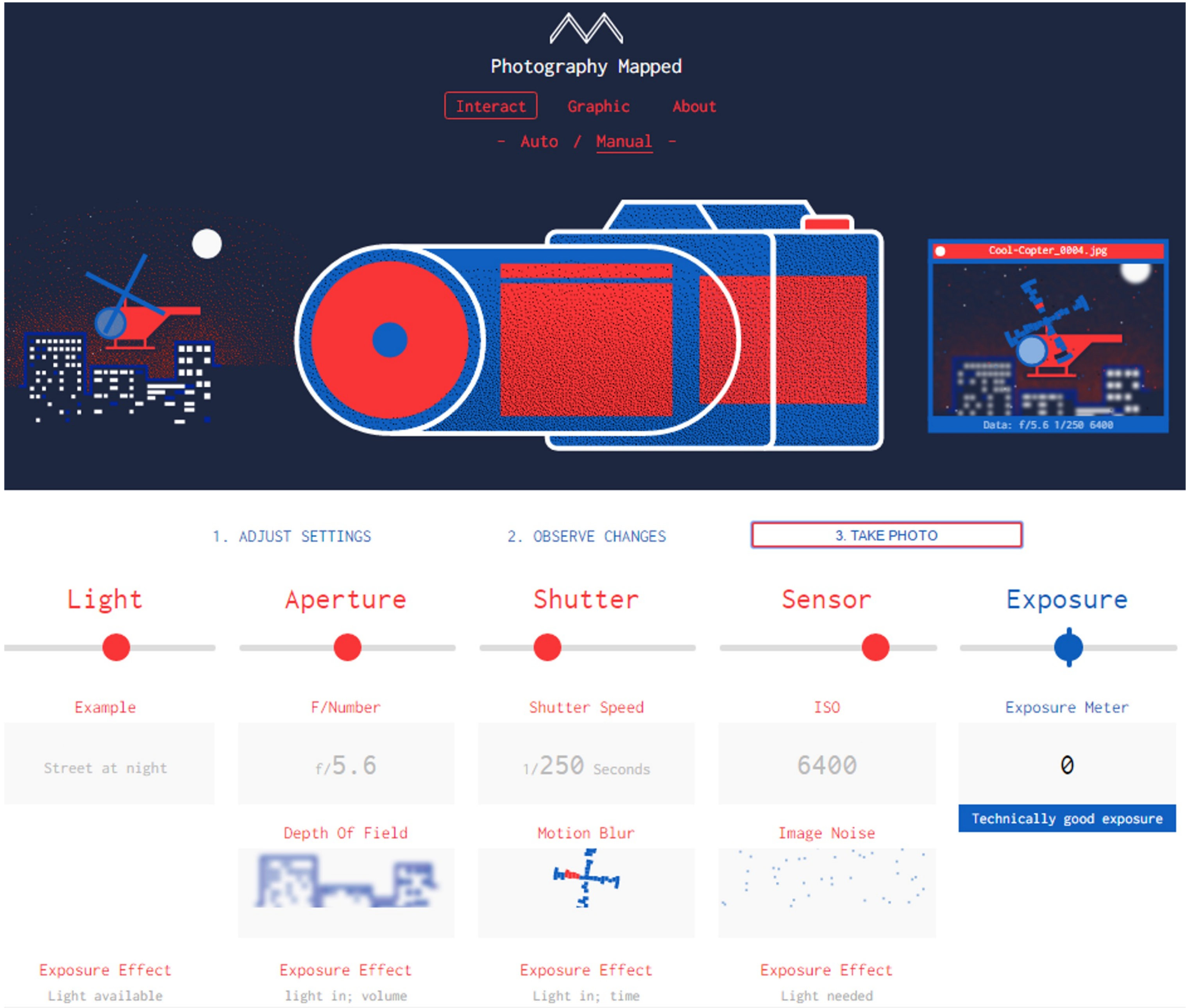


Nice visualization

Photography mapped:

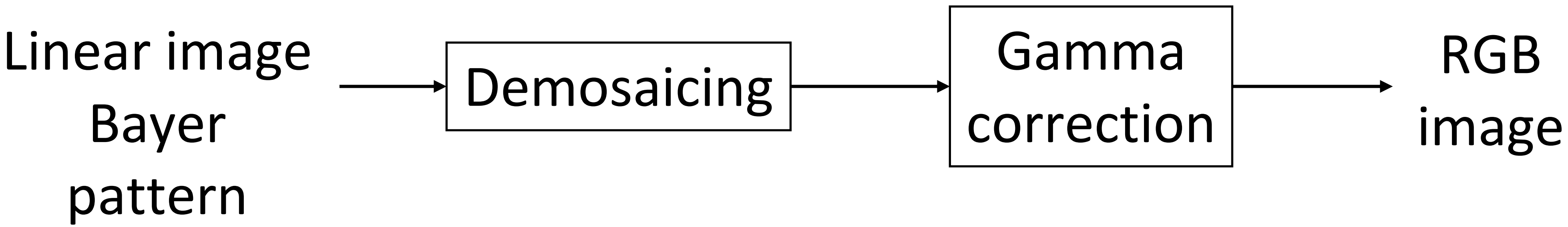
http://photography-mapped.com/interact.html

Play with the parameters and see what happens!



Task 2: Image processing pipeline (ISP)

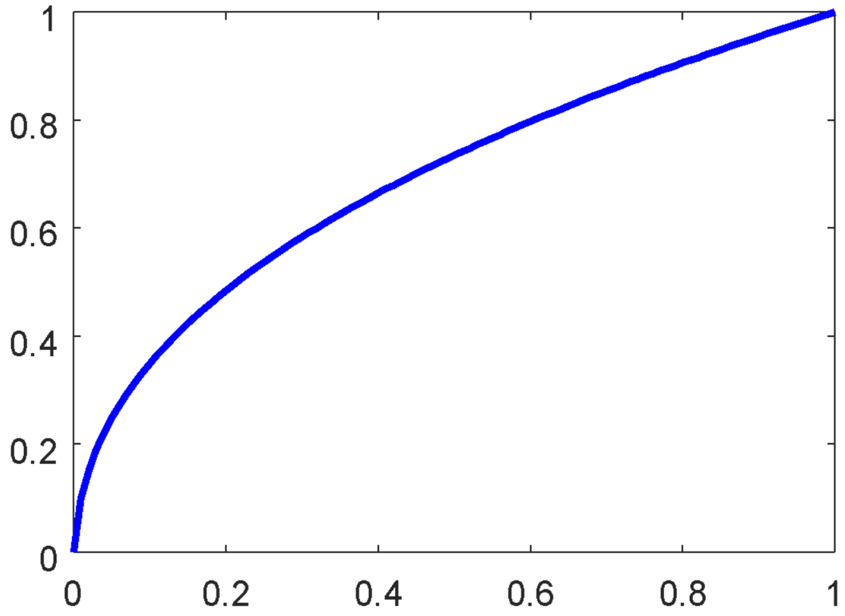
Simple ISP:



Task 2: Image processing pipeline (ISP)

Gamma correction:

- Scale pixel values to [0, 1] first
- Apply the gamma function $I \rightarrow I^{\left(\frac{1}{2.2}\right)}$

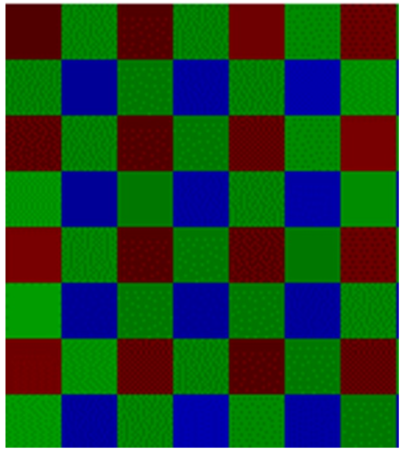


Task 2: Image processing pipeline (ISP)

Demosaicing:

Completing the missing values, for example, red in green pixel

First, find the order of colors in the Bayer pattern



Task 2: Image processing pipeline (ISP)

Implement several types of demosaicing:

1. Simple bilinear
2. Linear Demosaicing + low pass filtering the chrominance
3. High quality linear interpolation

Compare images both visually and quantitatively using the PSNR.

Task 2: Image processing pipeline (ISP)

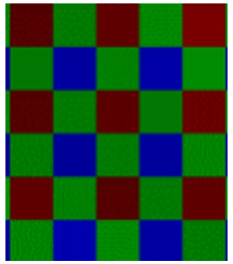
- Calculate the mean squared error (MSE) and the peak signal-to-noise ratio (PSNR):

$$MSE = \frac{1}{3mn}\sum_{c=1}^{3}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[I_{original}(i,j,c) - I_{restored}(i,j,c)\right]^2$$

$$PSNR = 10\ \log_{10}\left(\frac{\max\left(I_{original}\right)^2}{MSE}\right)$$

- Calculate PSNR after applying gamma correction

Task 2: Image processing pipeline (ISP)



Linear demosaicing:

- Create maps of the pixel coordinates with `np.meshgrid`.
- Interpolate the missing channels with `scipy.interpolate.interp2d`. The default is linear interpolation.
- Hints:
  - For the green channel, it's easier to average shifted versions of the original green channel. You can use `np.roll` to shift the image.
  - Combine the 3 channels into an RGB image using `np.stack([R, G, B], axis=2)`

Task 2: Image processing pipeline (ISP)

Simply interpolating the missing pixels may cause color artifacts

Task 2: Image processing pipeline (ISP)

Low pass filtering the chrominance should reduce these artifacts

- Using the result of the linear demosaiced image (before gamma correction)
- Use `skimage.color.rgb2ycbcr` to separate luminance and chrominance
- Luminance is given by Y and chrominance by Cb and Cr
- Smooth the chrominance, for example by median filtering (`scipy.ndimage.median_filter`, size 9)
- Convert back to RGB by using `skimage.color.ycbcr2rgb`

Task 2: Image processing pipeline (ISP)

Compare the results

Linear

Linear + smoothing the chrominance

Task 2: Image processing pipeline (ISP)

High quality linear interpolation:

- Refer to publication: R. Malvar, L. He, and R. Cutler, "High quality linear interpolation for demosaicing of Bayer-patterned color images", ICASPP, 2004.
- Still linear interpolation, BUT, interpolating one color channel uses information from other color channels.
- Exploiting the correlation among the RGB channels is the main idea for improving demosaicing performance
- Goal: preserve as much detail as possible

Task 2: Image processing pipeline (ISP)

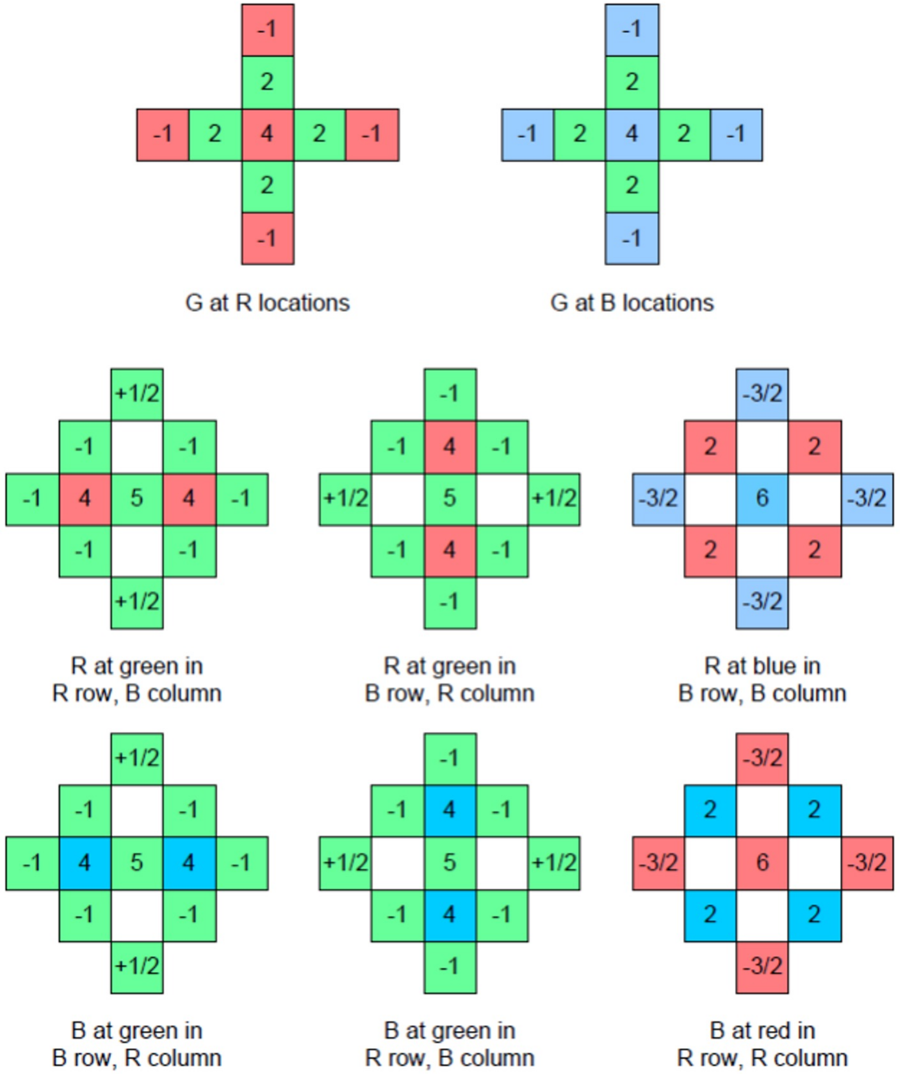
High quality linear interpolation:

- Assumption: edges have much stronger luminance than chrominance components
- → if there is a sharp change in one channel, it probably means there is a sharp luminance change
- Therefore, the change in one channel should be used in the interpolation of the other channels

Task 2: Image processing pipeline (ISP)

High quality linear interpolation filters:

- **Builds upon simple linear interpolation and adds gradients from other channels**
- Some of the filters are similar, only 4 unique
- Many multiplications are factors of 2. This is extremely more efficient in hardware compared to regular multiplication
- Hints:
    - Convolve the entire image with the filters, then choose the pixels you need in order to complete the RGB image correctly



Task 2: Image processing pipeline (ISP)

Compare the results

Are there less or more color artifacts?
What about detail?

Linear

High Quality Linear

Task 3: Denoising

1. Gaussian Filtering (use provided `fspecial_gaussian_2d` and `scipy.signal.convolve2d`)
2. Median Filtering (use `scipy.ndimage.median_filter`)
3. Bilateral Filtering
4. Non-local Means

Task 3: Denoising

Bilateral filtering

A non-linear, edge-preserving and noise-reducing smoothing filter.

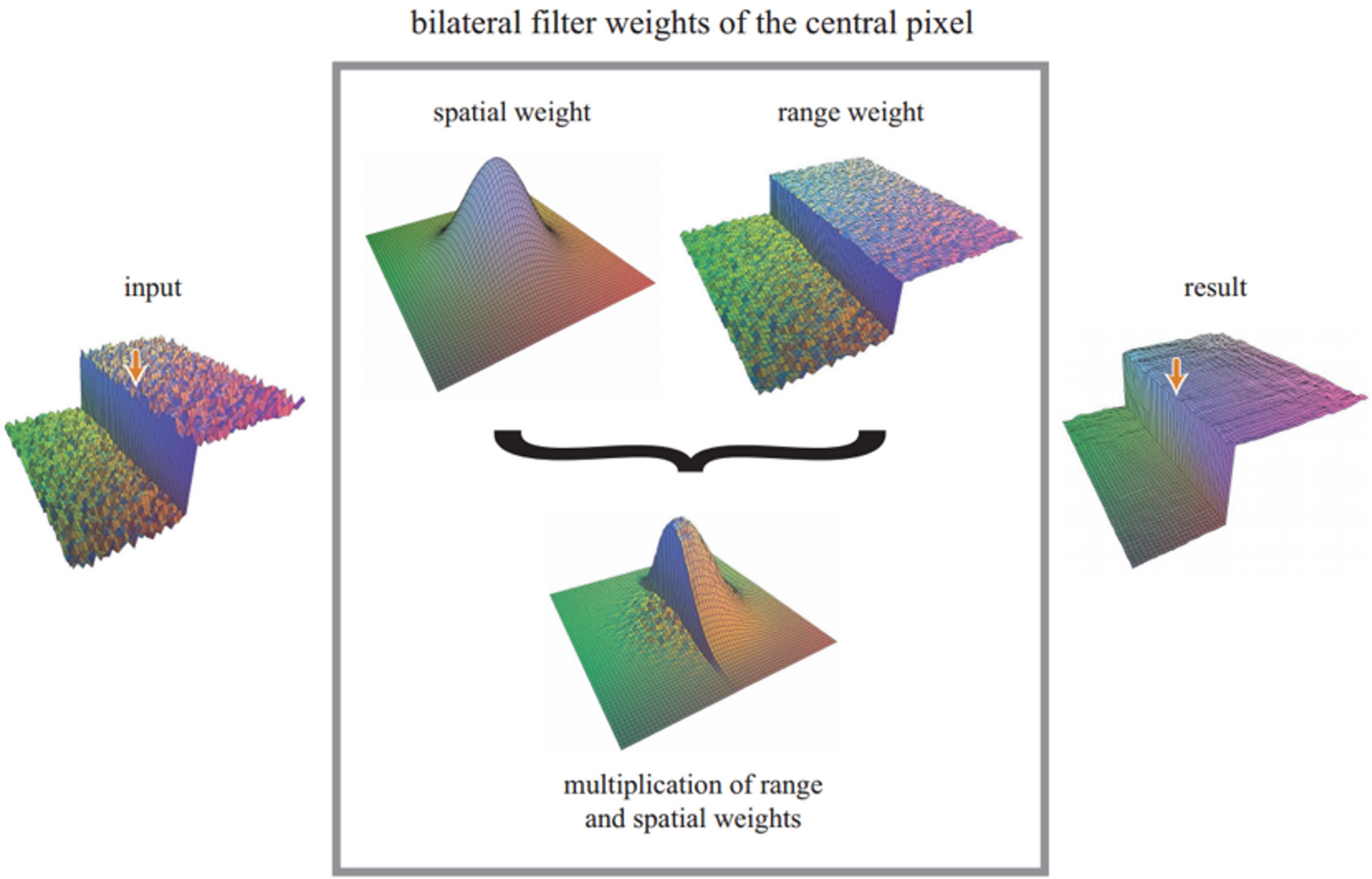
The intensity value at each pixel in an image is replaced by a weighted average of intensity values from nearby pixels. **The weights depend not only on Euclidean distance of pixels, but also on the radiometric differences.**

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) \overbrace{f_r(\|I(x_i) - I(x)\|)}^{\text{Intensity weight}} \overbrace{g_s(\|x_i - x\|)}^{\text{Spatial weight}},$$

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

Task 3: Denoising

Bilateral filtering of a pixel on the high side of the edge



Task 3: Denoising

Bilateral filtering hints:

- Use a Gaussian function for both weights. For the spatial weight use fspecial and for the intensity weight calculate $e^{-\left(I(x_i)-I(x)\right)^2/2\sigma_{int}^2}$
- Accumulate the weights to obtain W_p

$$I^{\text{filtered}}(x) = \frac{1}{W_p}\sum_{x_i\in\Omega} I(x_i)\underbrace{f_r(\|I(x_i)-I(x)\|)}_{\text{Intensity weight}}\underbrace{g_s(\|x_i-x\|)}_{\text{Spatial weight}},$$

$$W_p = \sum_{x_i\in\Omega} f_r(\|I(x_i)-I(x)\|)g_s(\|x_i-x\|)$$

Task 3: Denoising

"A non-local algorithm for image denoising", A. Buades, B. Coll, JM. Morel (2005)

Non-local means

Given a discrete noisy image $v = \{v(i) \mid i \in I\}$, the estimated value $NL[v](i)$, for a pixel i, is computed as a weighted average of all the pixels in the image,

$$NL[v](i) = \sum_{j \in I} w(i,j) v(j),$$

where the family of weights $\{w(i,j)\}_j$ depend on the similarity between the pixels i and j, and satisfy the usual conditions $0 \leq w(i,j) \leq 1$ and $\sum_j w(i,j) = 1$.

These weights are defined as,

$$w(i,j) = \frac{1}{Z(i)} e^{-\frac{\|v(\mathcal{N}_i) - v(\mathcal{N}_j)\|^2_{2,a}}{h^2}},$$

Weighted norm: Gaussian with standard deviation a.

h controls the decay of the weights as a function of the Euclidean distances.

where $Z(i)$ is the normalizing constant

$$Z(i) = \sum_j e^{-\frac{\|v(\mathcal{N}_i) - v(\mathcal{N}_j)\|^2_{2,a}}{h^2}}$$

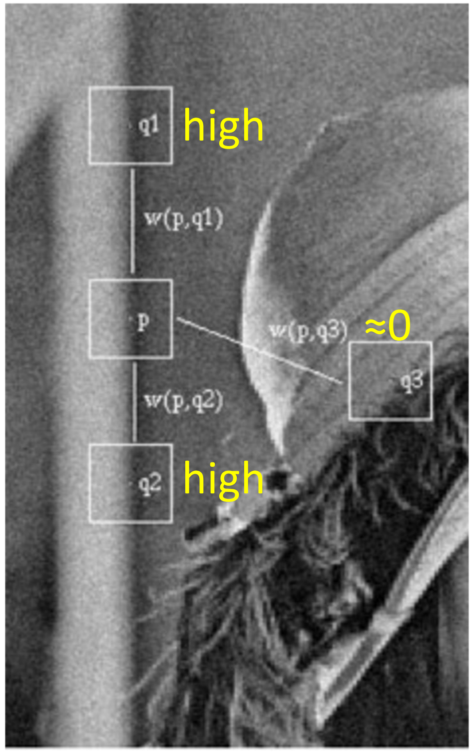
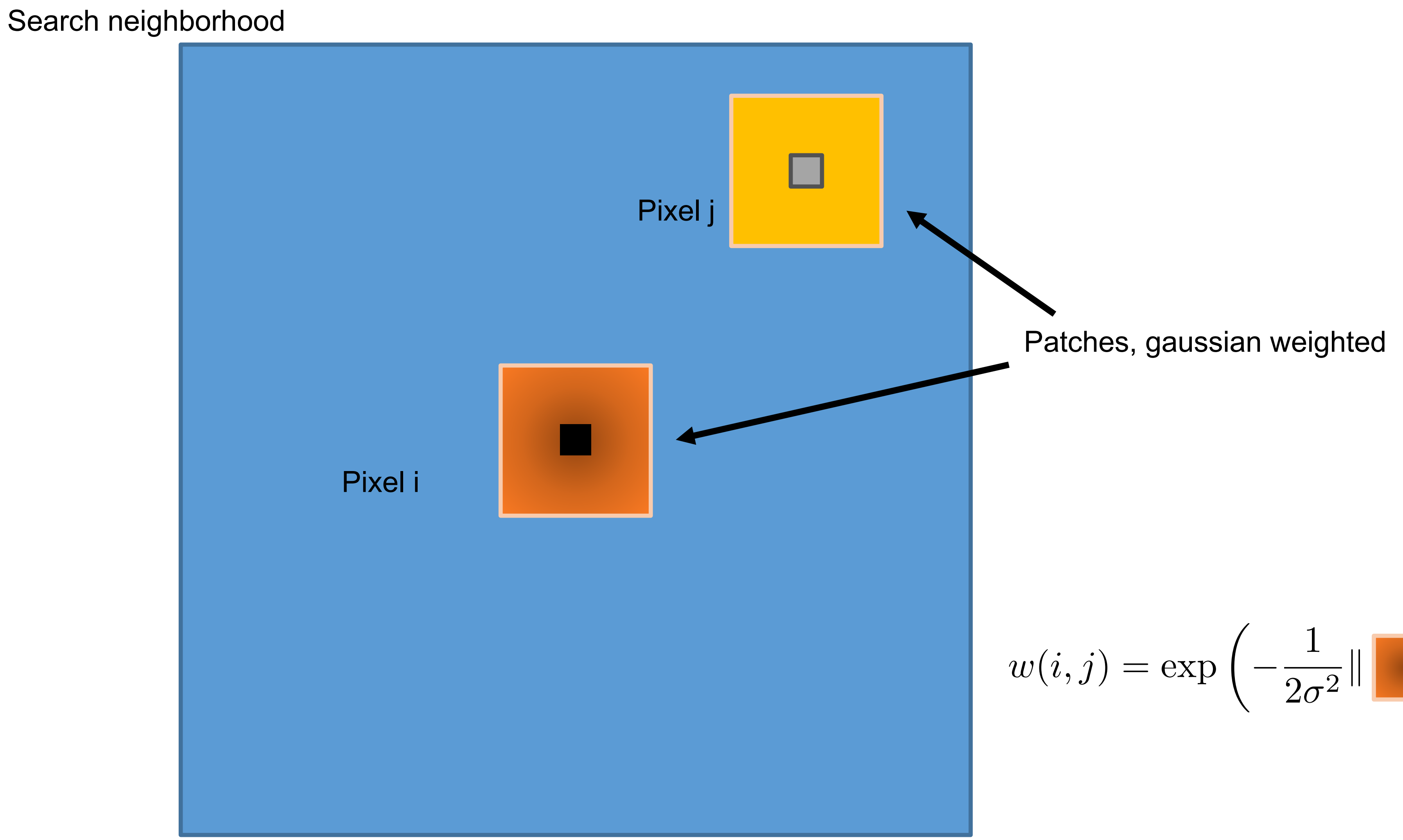


Figure 1. Scheme of NL-means strategy. Similar pixel neighborhoods give a large weight, w(p,q1) and w(p,q2), while much different neighborhoods give a small weight w(p,q3).

Nonlocal Means – weight calculation



$$w(i,j) = \exp\left(-\frac{1}{2\sigma^2}\| \quad - \quad \|^2\right)$$

Task 3: Denoising

Non-local means hints:

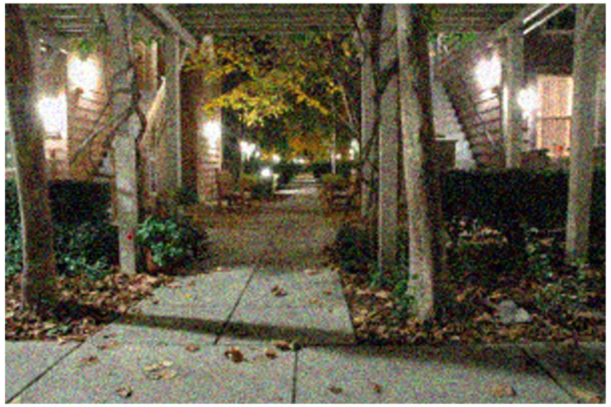
- Pad image to reduce boundary artifacts.
- Don't search the whole image for similar neighborhoods – it will take too long. Restrict the search to 15x15 pixels (or less).
- The windows overlap.
- Do not include the window centered on the current pixel when searching for other, similar windows!
- The formula for the weighted norm is $$w(i,j) = \frac{1}{Z(i)} \exp\left(-\frac{\sum_{mn}\left(k_{mn}\left(v(N_i)_{mn} - v(N_j)_{mn}\right)^2\right)}{h^2}\right)$$
- k_{mn} can be coefficients of a 2D Gaussian kernel.
- Accumulate the weights to obtain the normalization factor. Normalize the weights to obtain a sum of 1.

NB: Weight the center pixel with the maximal weight seen in the neighborhood.

Task 3

Expected results

Noisy image



Gaussian filter, σ=1, window size=3

Gaussian filter, σ=2, window size=5

Gaussian filter, σ=3, window size=7

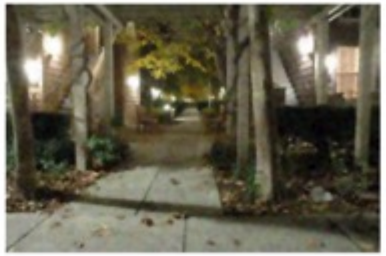
Median filter, window size=3

Median filter, window size=5

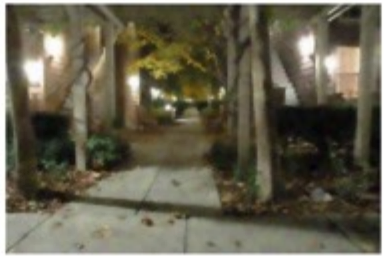
Median filter, window size=7

Bilateral filter, σ=1, window size=3, σ_{int}=0.25

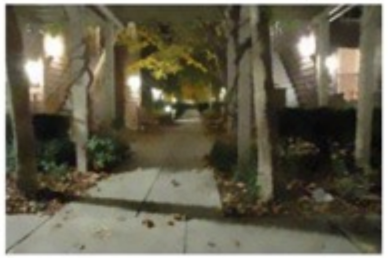
Bilateral filter, σ=2, window size=5, σ_{int}=0.25



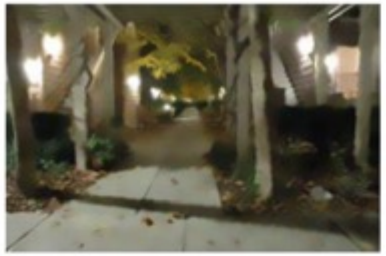
Bilateral filter, σ=3, window size=7, σ_{int}=0.25



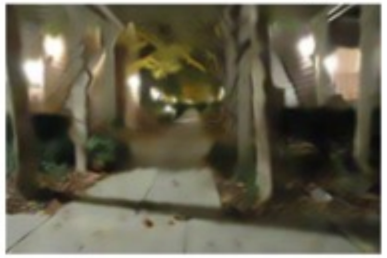
NLM, σ=1, window size=3, σ_{NLM}=0.1



NLM, σ=2, window size=5, σ_{NLM}=0.1



NLM, σ=3, window size=7, σ_{NLM}=0.1



Which looks best?

Have a good week!

And good luck with the homework!