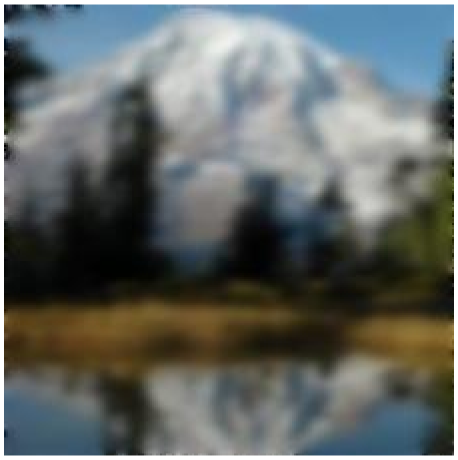
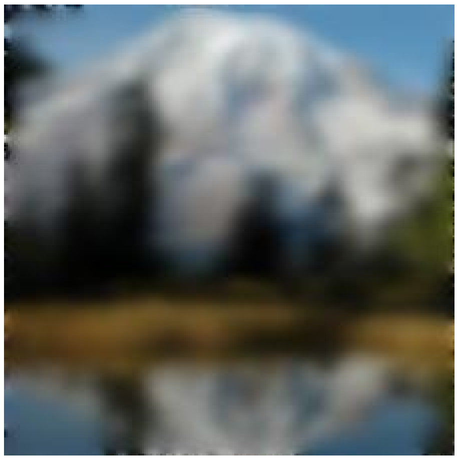
HACETTEPE UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING
FUNDAMENTALS OF IMAGE PROCESSING LABORATORY
PROGRAMMING ASSIGNMENT 2
FALL 2021-2022
TA. YUNUS BİLGE
DR. AYDIN KAYA

İSMAİL ATEŞ - 21626953

Mean Filter

- Mean filtering is a basic, intuitive, and straightforward way of smoothing images, i.e. reducing the amount of intensity variance between each pixel. It is frequently used to reduce image noise.
- Mean filtering works by replacing each pixel in an image with the mean ('average') value of its neighbors, including itself. As a result, pixel values that aren't typical of their surroundings are removed. It is based on a kernel, which describes the shape and size of the sampled neighborhood for calculating the mean.
- A single pixel with a very unusual value can have a considerable impact on the mean value of all the pixels in its neighborhood.
- When a filter neighborhood crosses an edge, the filter interpolates new values for pixels on the edge, blurring the edge. If sharp edges are required in the output, this might be an issue.
- The larger the kernel, the more blurred the image becomes. As you can see in the code implementation, the number of unblurred pixels increases as the size of the kernel increases, since the edge pixels are not interfered with.
- **Examples**
  Example 1:
  original
  3x3 5x5
  7x7 9x9

Example 2

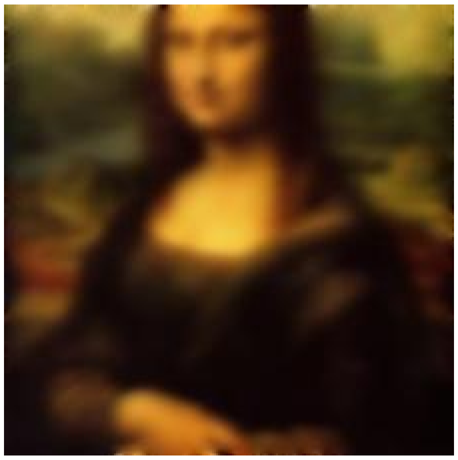
original
3x3 5x5
7x7 9x9

Example 3

original

3x3 5x5

7x7 9x9

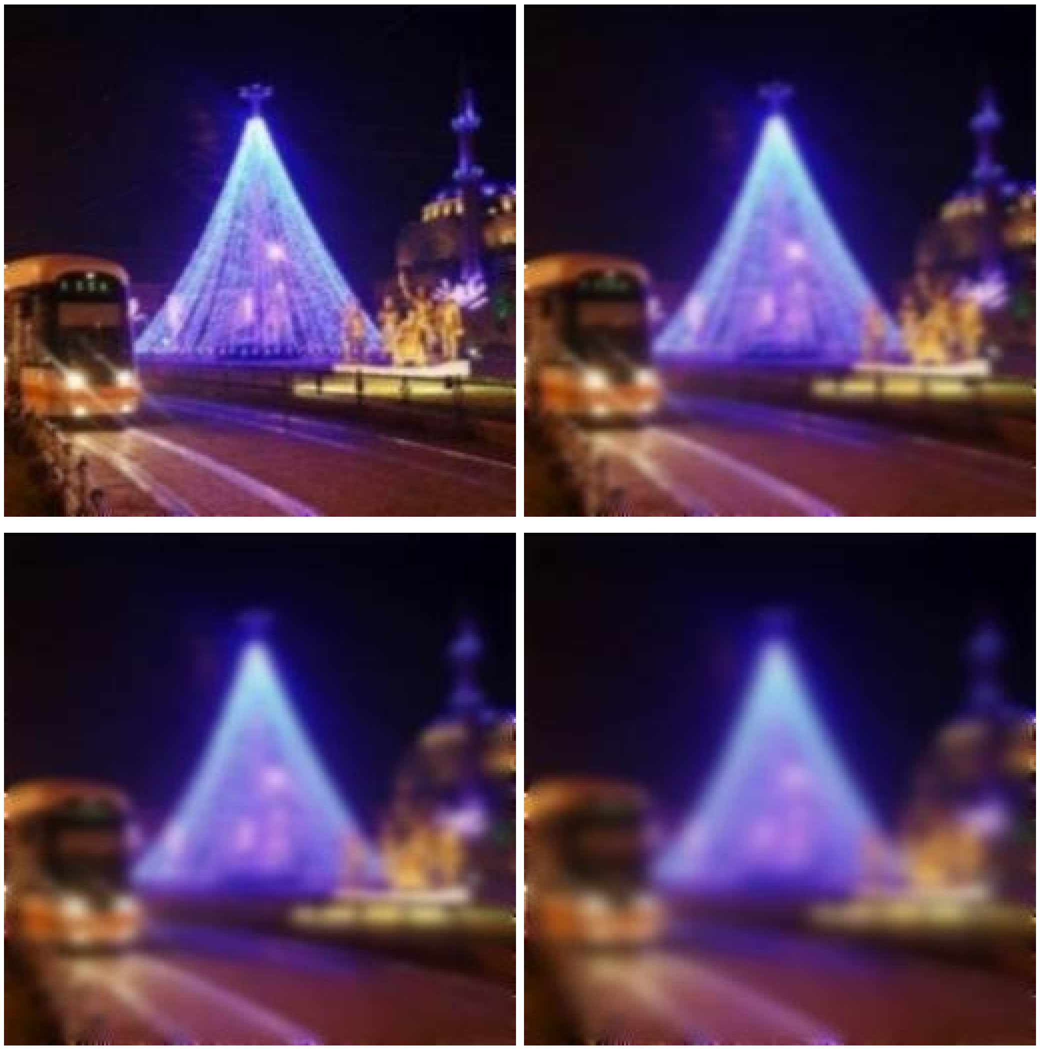


Example 4

original

3x3 5x5

7x7 9x9



Example 5

original

3x3 5x5

7x7 9x9

Gaussian Filter

- The Gaussian filter removes detail and noise from images by 'blurring' them. It's similar to the mean filter in this regard, but it employs a different kernel to reflect the form of a Gaussian hump.
- The Gaussian Filter in 1-D has the form:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

where $\boldsymbol{\sigma}$ is the standard deviation of the distribution.

- In 2-D, Gaussian Filter has the form:

$$G(x,y) = \frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

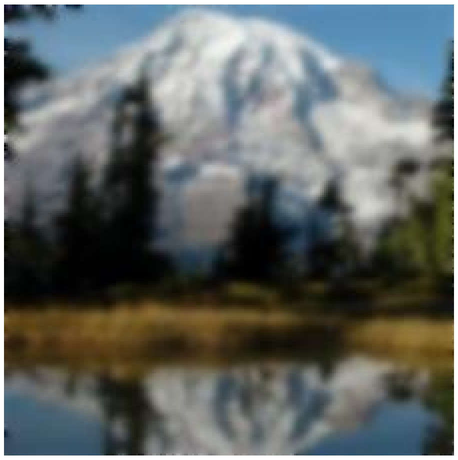
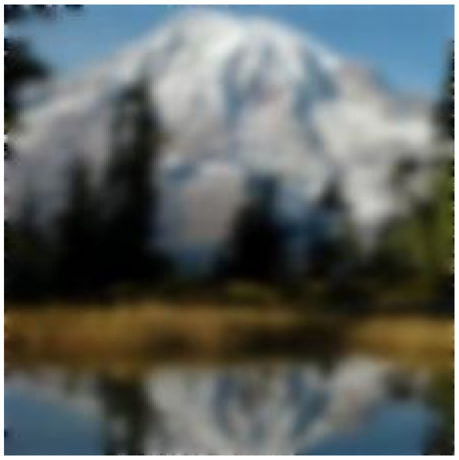
- Gaussian filtering is based on the notion of using this 2-D distribution as a 'point-spread' function, which is accomplished using convolution. We need to construct a discrete approximation to the Gaussian function before we can execute the convolution since the image is stored as a collection of discrete pixels. The Gaussian distribution is non-zero everywhere in principle, requiring an arbitrarily long convolution kernel, but it is essentially zero beyond around three standard deviations from the mean in practice, allowing us to terminate the kernel at this point.
- The larger the kernel and the sigma value, the more blurred the image becomes. As you can see in the code implementation, the number of unblurred pixels increases as the size of the kernel increases, since the edge pixels are not interfered with.
- **Examples**
  Example 1 (kernel 3x3)
  orginal
  sigma 0.5 sigma 1.5
  sigma 1 sigma 2

Example1 (Kernel 5x5)

orginal

sigma 0.5 sigma 1.5

sigma 1 sigma 2

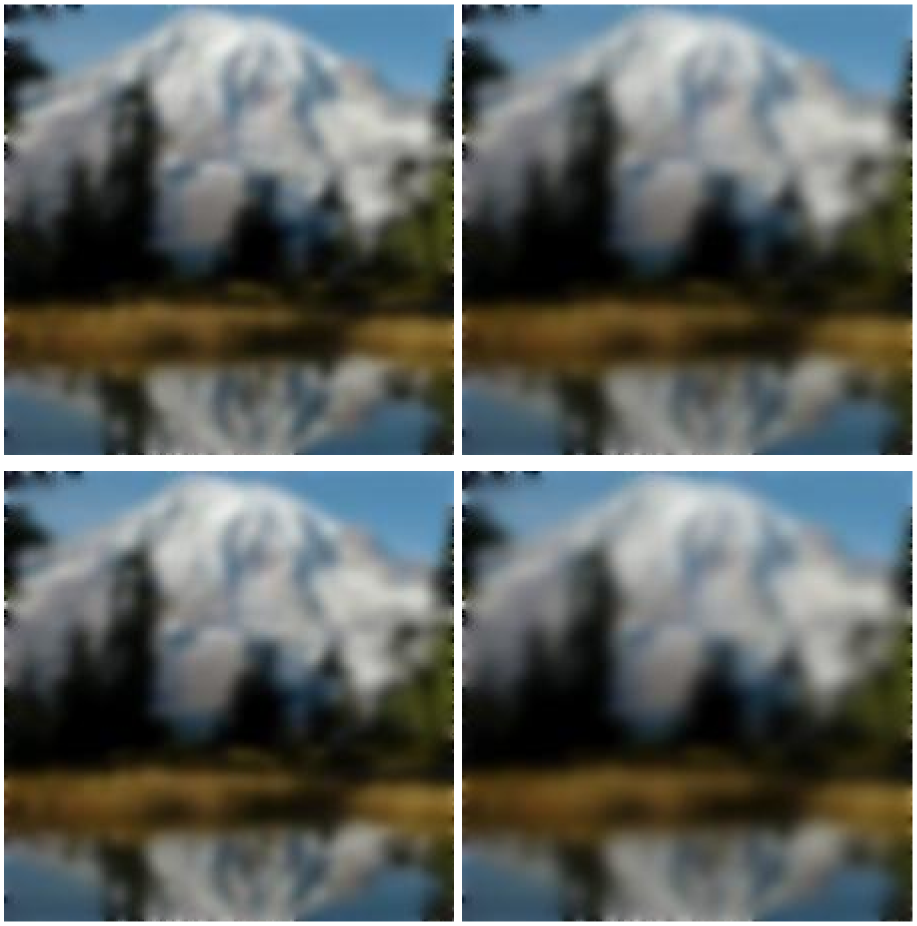



Example1 (Kernel 7x7)

orginal

sigma 0.5 sigma 1.5

sigma 1 sigma 2

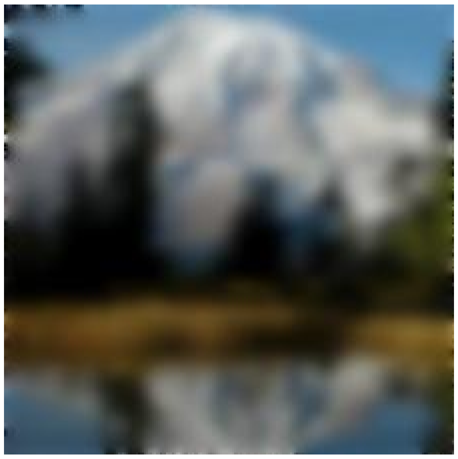
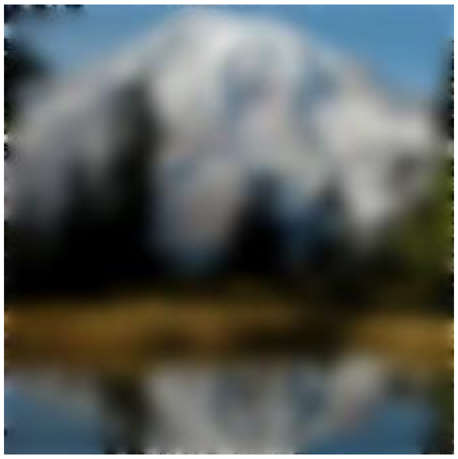


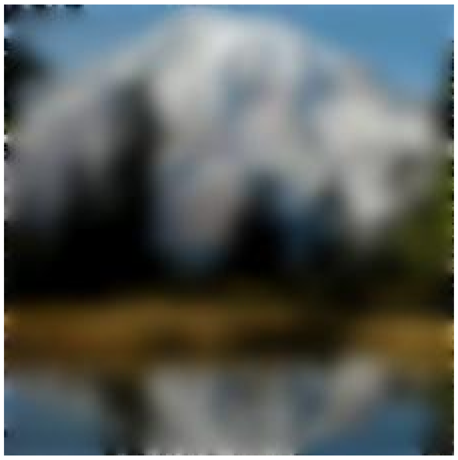
Example1 (Kernel 9x9)

orginal

sigma 0.5 sigma 1.5

sigma 1 sigma 2



Example2 (Kernel 3x3)

orginal

sigma 0.5 sigma 1.5

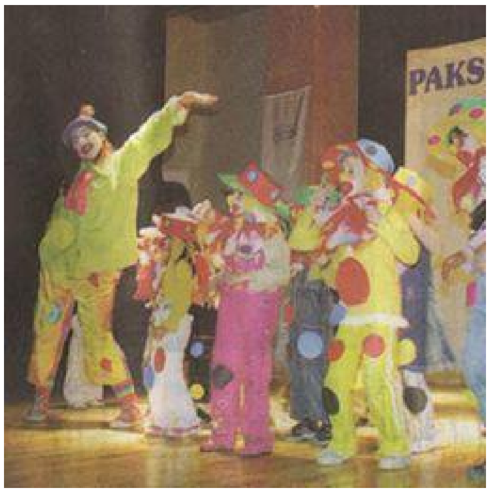
sigma 1 sigma 2

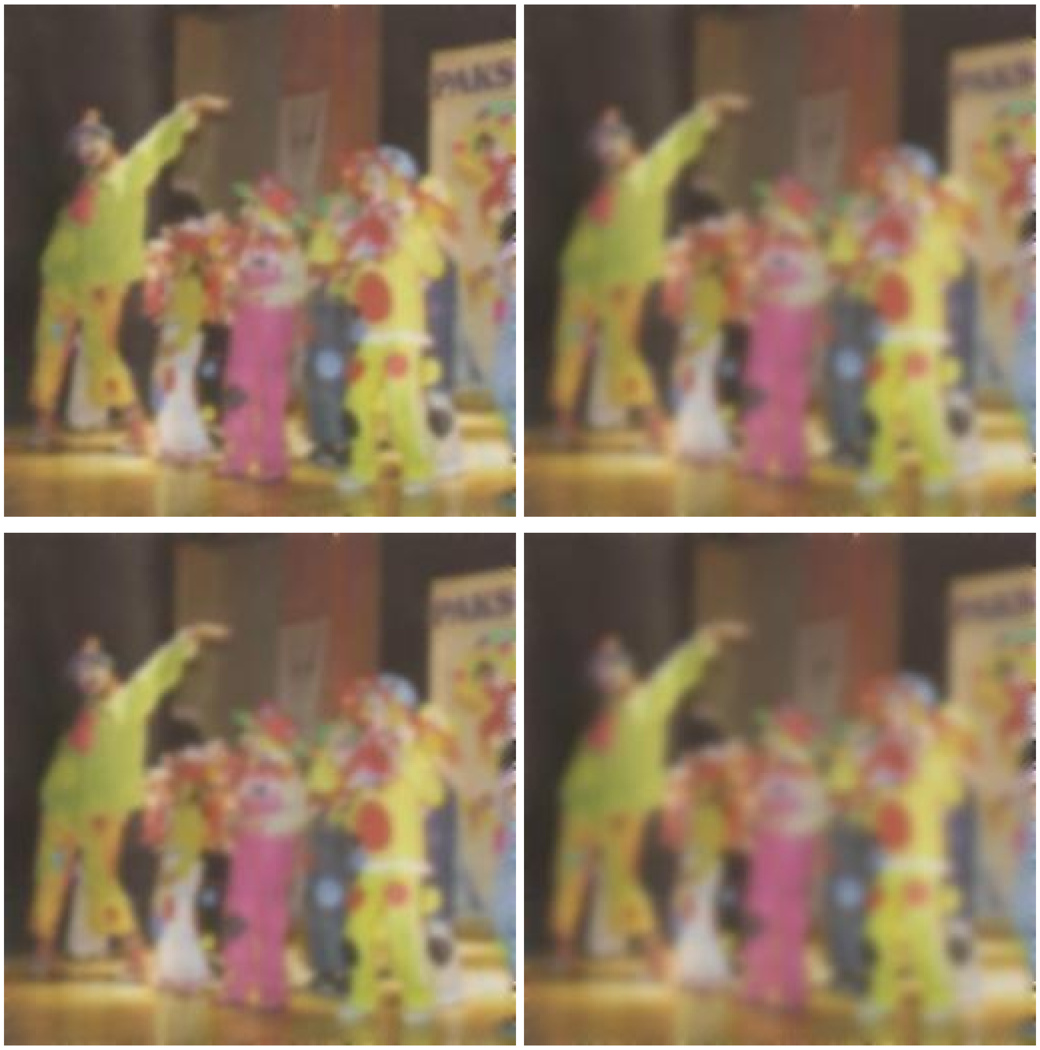
Example2 (Kernel 5x5)

orginal

sigma 0.5 sigma 1.5

sigma 1 sigma 2



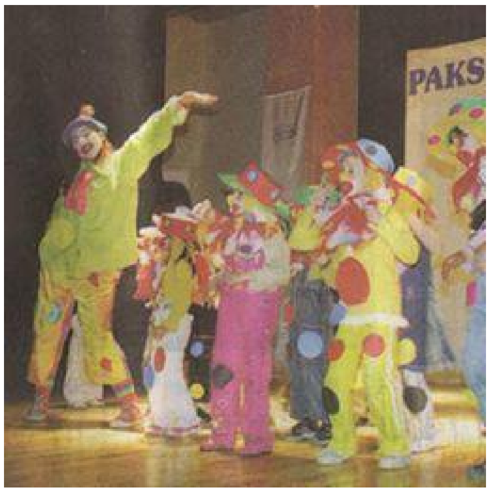


Example2 (Kernel 7x7)

orginal

sigma 0.5 sigma 1.5

sigma 1 sigma 2

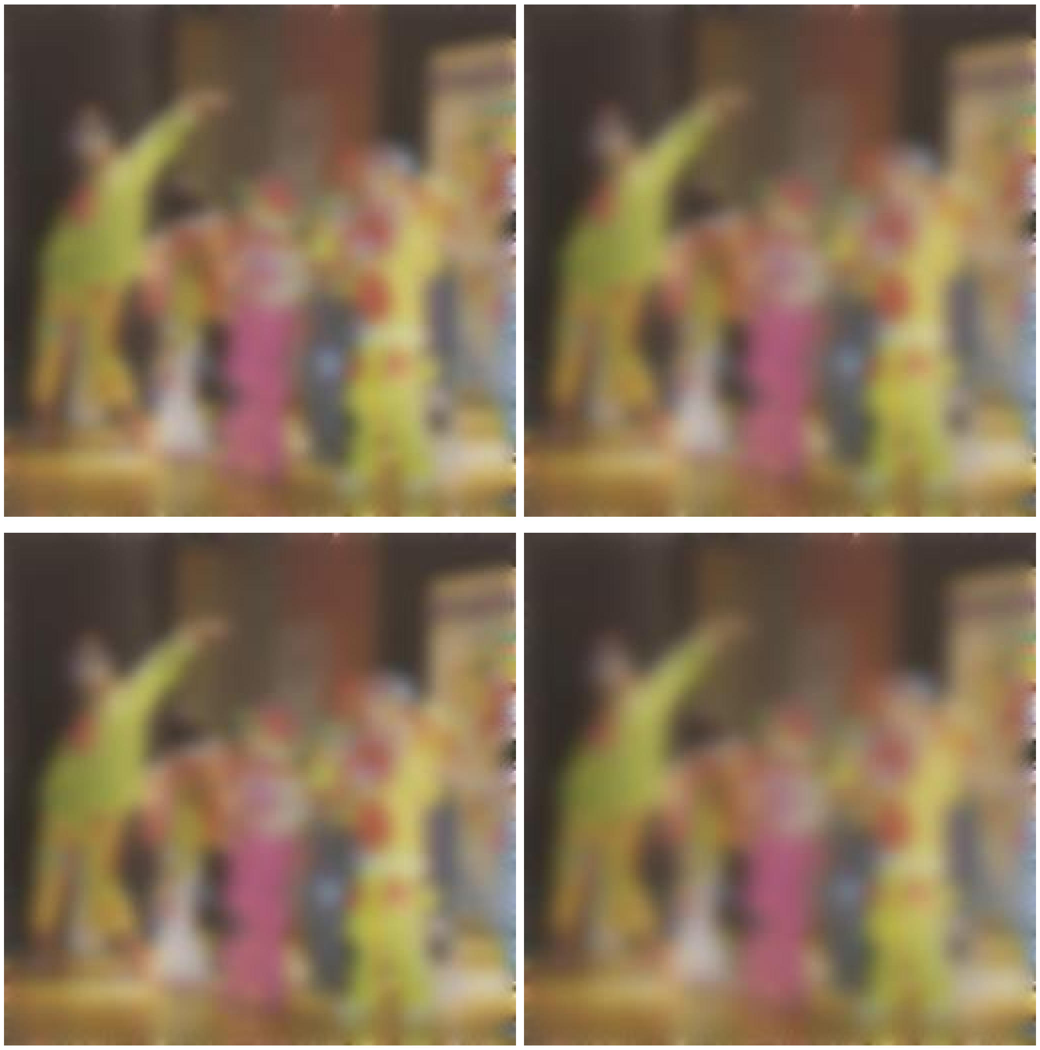


Example2 (Kernel 9x9)

orginal

sigma 0.5 sigma 1.5

sigma 1 sigma 2



Example 3 (Kernel 3x3)

orginal

sigma 0.5 sigma 1.5

sigma 1 sigma 2

Example 3 (Kernel 5x5)

orginal

sigma 0.5 sigma 1.5

sigma 1 sigma 2

Example 3 (Kernel 7x7)

orginal

sigma 0.5 sigma 1.5

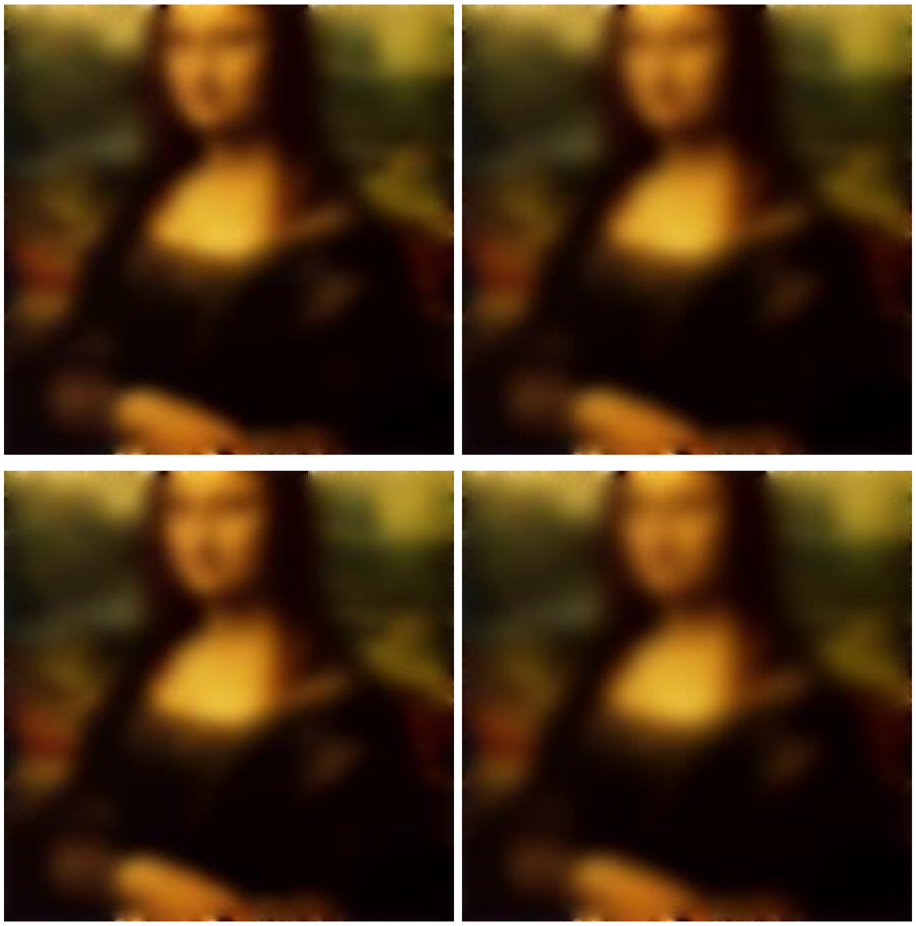
sigma 1 sigma 2

Example 3 (Kernel 9x9)

orginal

sigma 0.5 sigma 1.5

sigma 1 sigma 2



Example 4 (Kernel 3x3)

orginal

sigma 0.5 sigma 1.5

sigma 1 sigma 2

Example 4 (Kernel 5x5)
orginal
sigma 0.5 sigma 1.5
sigma 1 sigma 2

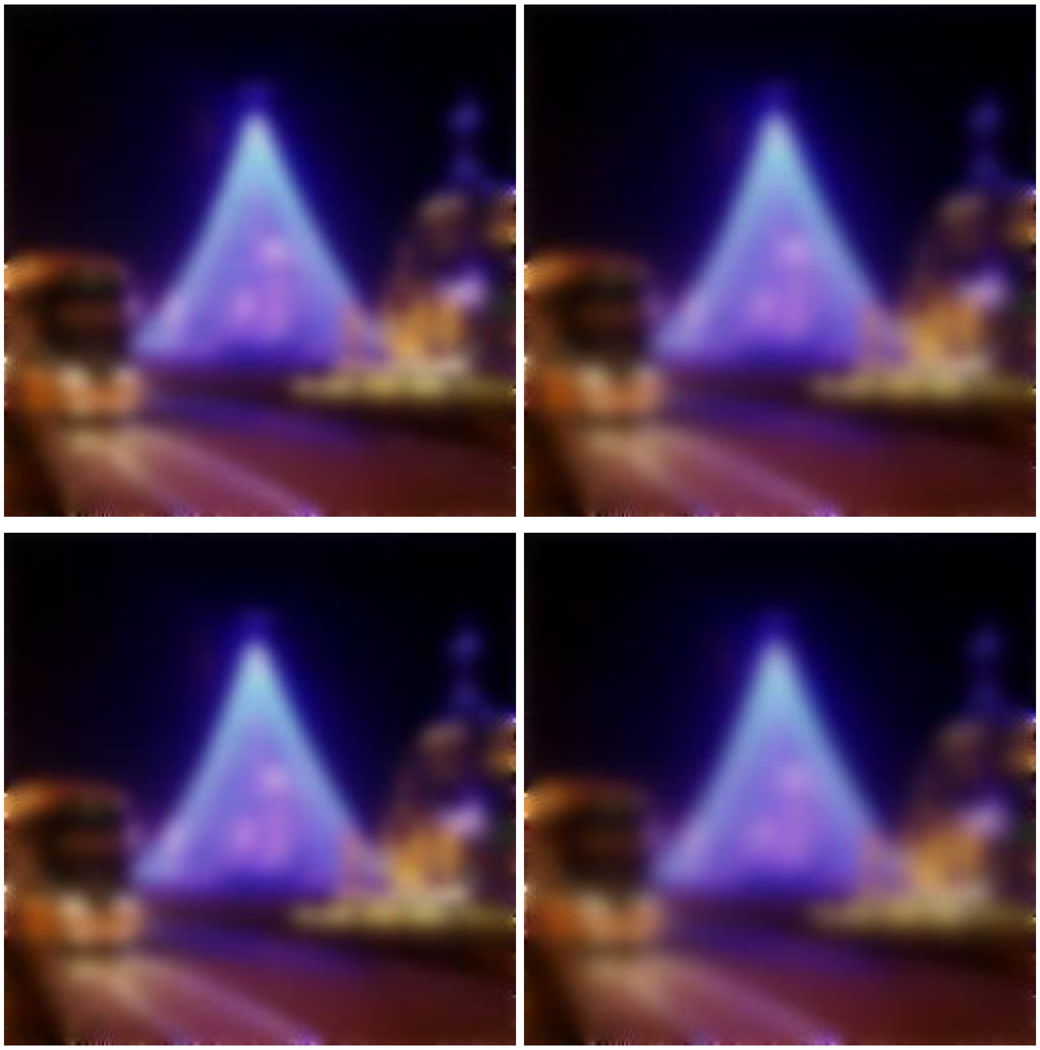
Example 4 (7x7)

orginal

sigma 0.5 sigma 1.5

sigma 1 sigma 2

Example 4 (Kernel 9x9)
orginal
sigma 0.5 sigma 1.5
sigma 1 sigma 2



Example 5 (Kernel 3x3)

orginal

sigma 0.5 sigma 1.5

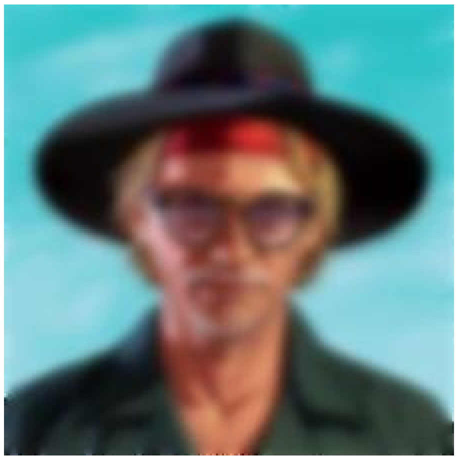
sigma 1 sigma 2

Example 5 (Kernel 5x5)

orginal

sigma 0.5 sigma 1.5

sigma 1 sigma 2

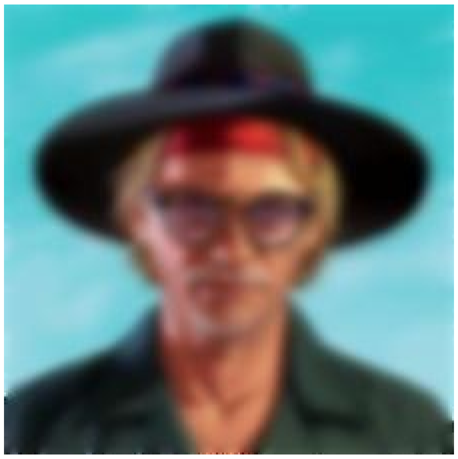
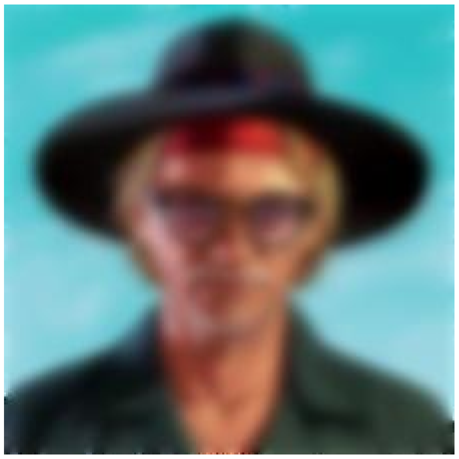
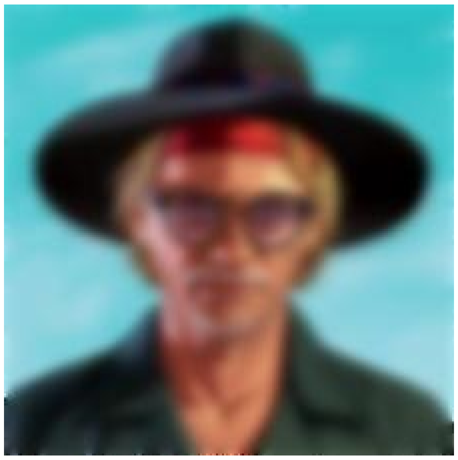


Example 5 (Kernel 7x7)

orginal

sigma 0.5 sigma 1.5

sigma 1 sigma 2

Example 5 (Kernel 9x9)

orginal

sigma 0.5 sigma 1.5

sigma 1 sigma 2

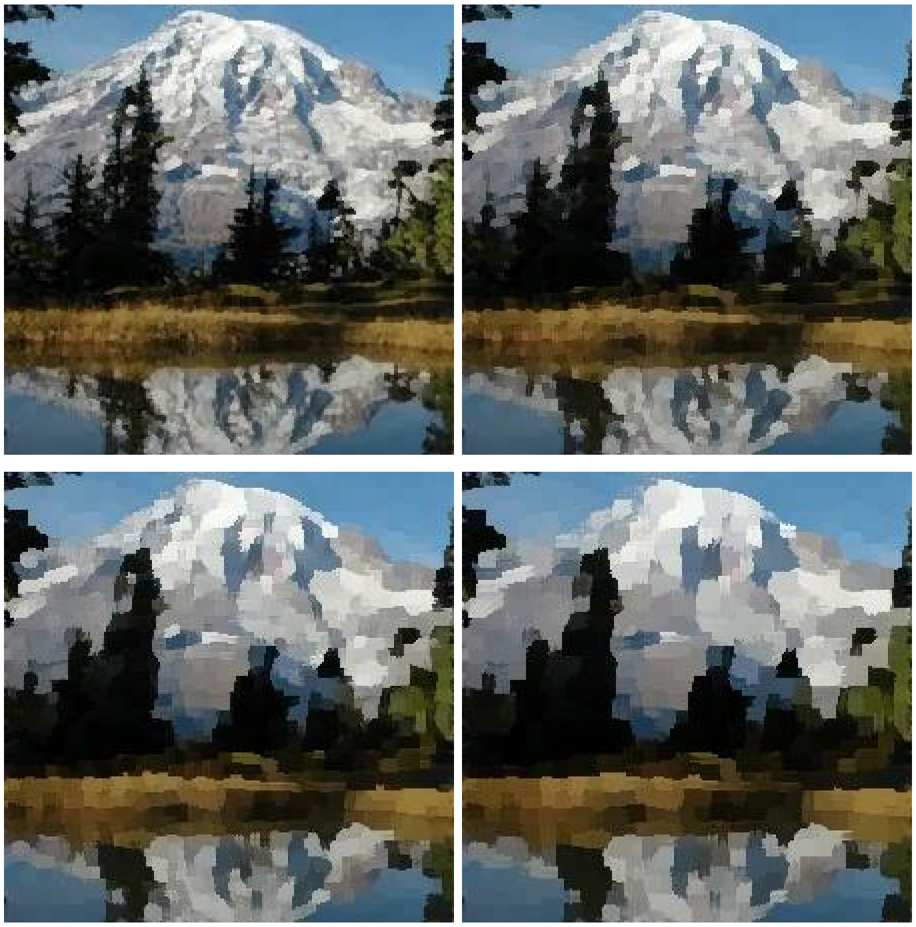
Kuwahara Filter

- The Kuwahara filter is an adaptive noise reduction smoothing filter. The most common image smoothing filters are linear low-pass filters, which efficiently decrease noise while blurring the edges. The Kuwahara filter, on the other hand, may smooth the image while keeping the edges.
- The filter formula:

$$Q_i(x,y) = \begin{cases} [x, x+a] \times [y, y+a] & \text{if } i = 1 \\ [x-a, x] \times [y, y+a] & \text{if } i = 2 \\ [x-a, x] \times [y-a, y] & \text{if } i = 3 \\ [x, x+a] \times [y-a, y] & \text{if } i = 4 \end{cases}$$

- This signifies that the mean value of the most homogeneous area will be used as the central pixel. The position of a pixel in relation to an edge has a significant impact on which region has the highest standard deviation. If a pixel is placed on the dark side of an edge, it will most likely take the dark region's mean value.

- Applying the filter to each RGB channel independently and then recombining the three filtered color channels to generate the filtered RGB image is not recommended for color images. The key issue is that the standard deviations for each of the channels will be different in the quadrants. For color pictures, a slightly modified Kuwahara filter must be employed to solve this problem. The image is first transformed to the HSV color model, which is a different color space. The improved filter now only works on the "brightness" channel, which corresponds to the Value coordinate in the HSV model. The variance of each quadrant's "brightness" is computed to decide which quadrant should be used for the final filtered color.
- As you can see in the sample images, the images become more cartoonish as the kernel size increases.
- **Examples**
  Example 1
  original
  3x3 5x5
  7x7 9x9

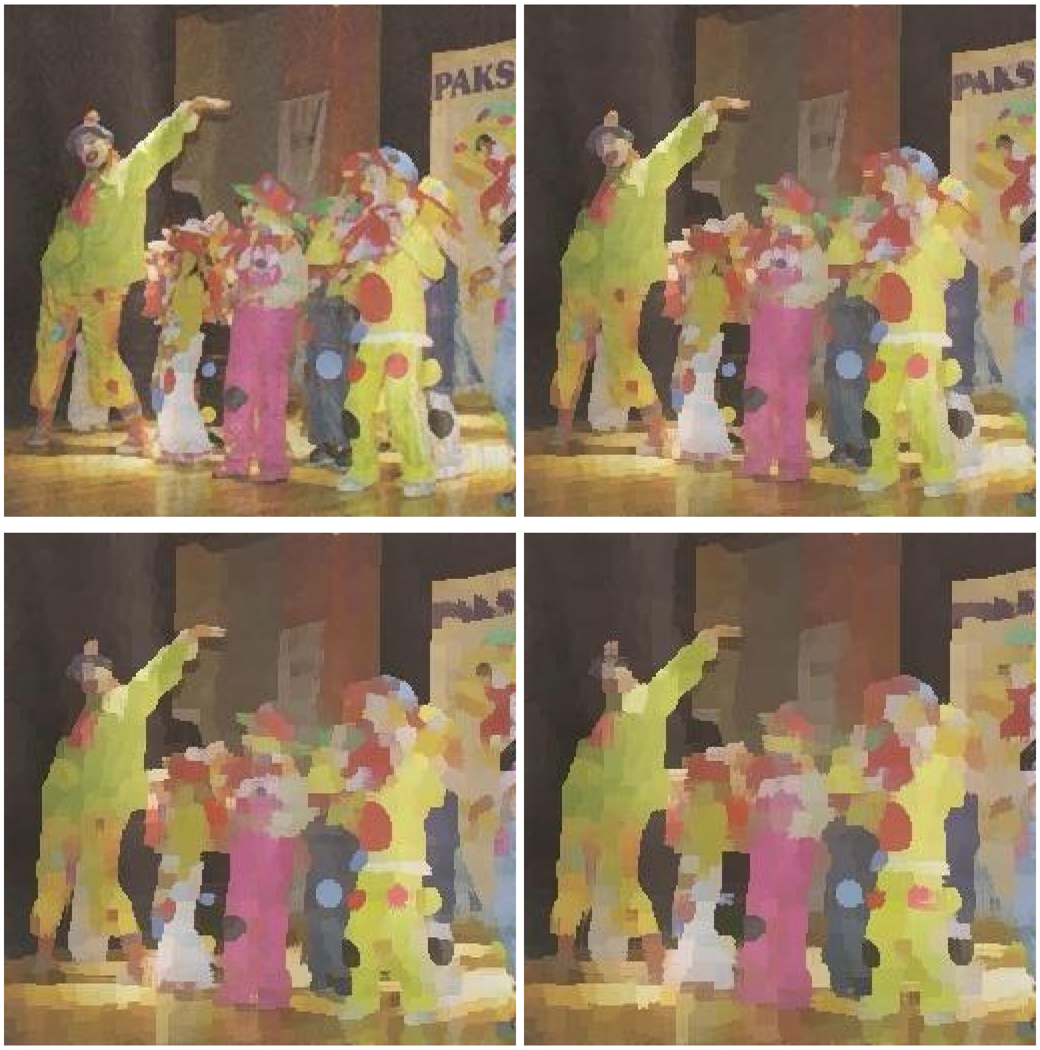


Example 2

original

3x3 5x5

7x7 9x9

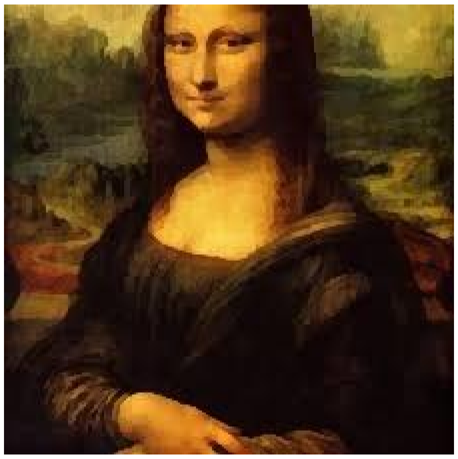
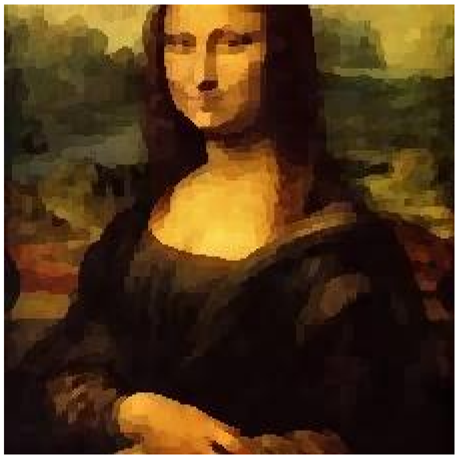
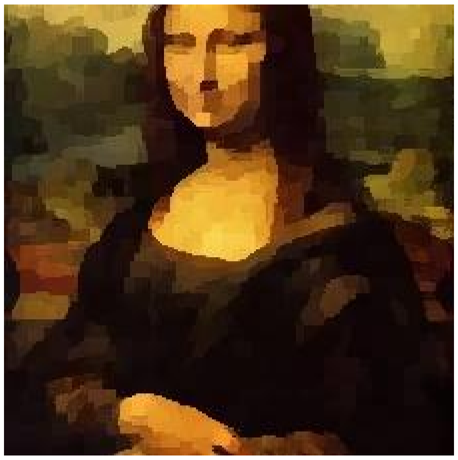
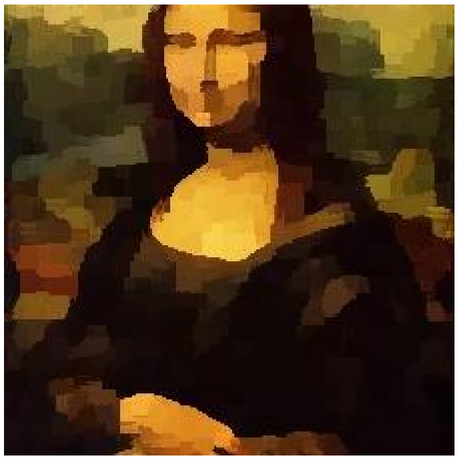


Example 3

original

3x3 5x5

7x7 9x9

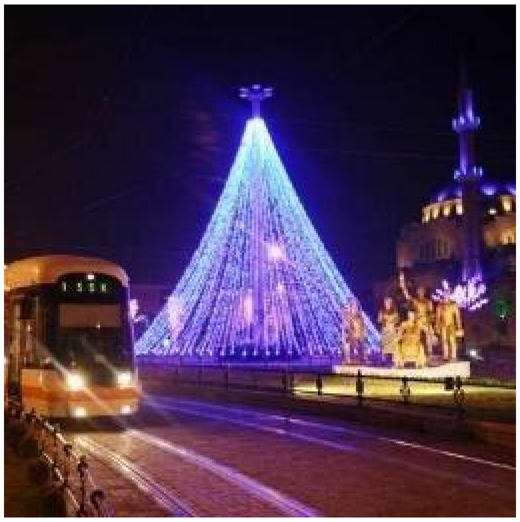





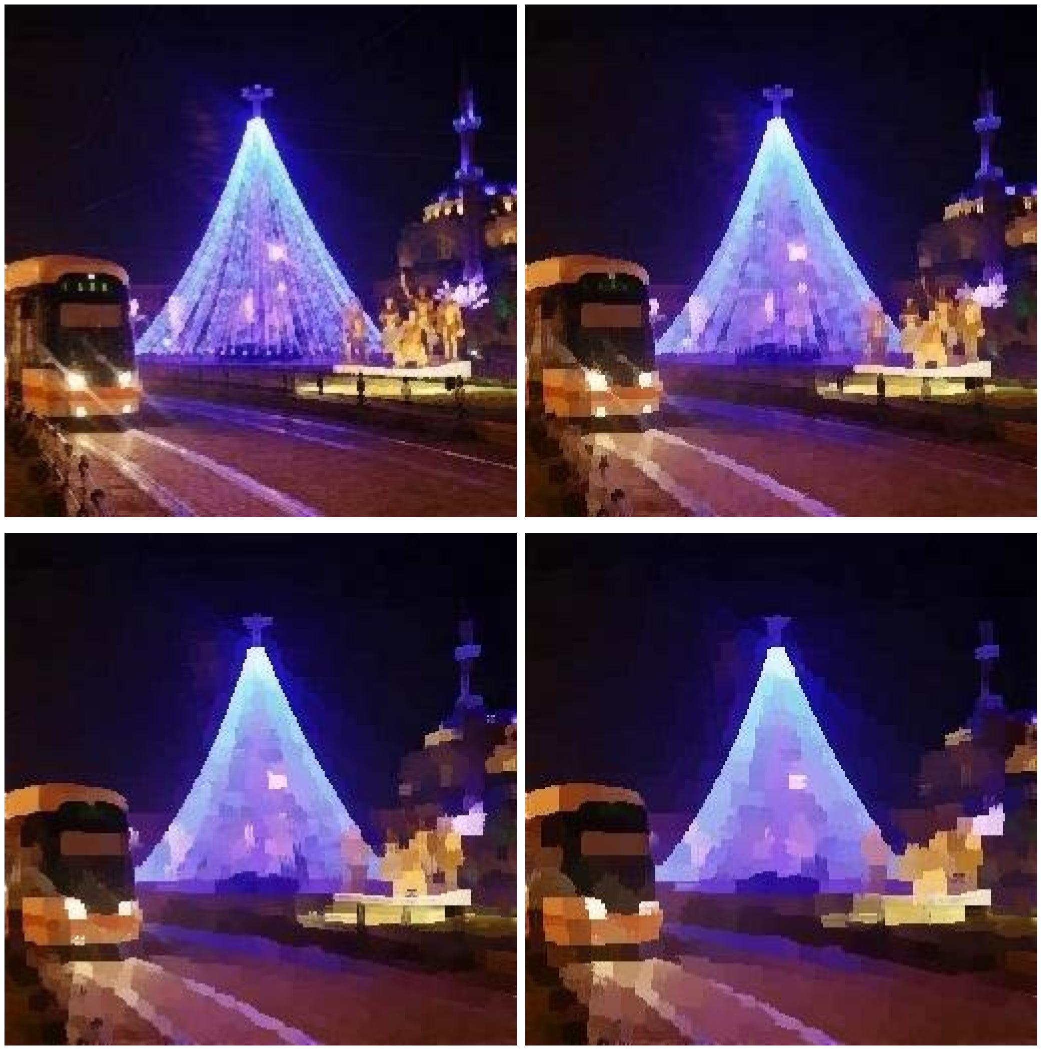
Example 4

original

3x3 5x5

7x7 9x9





Example 5

original

3x3 5x5

7x7 9x9