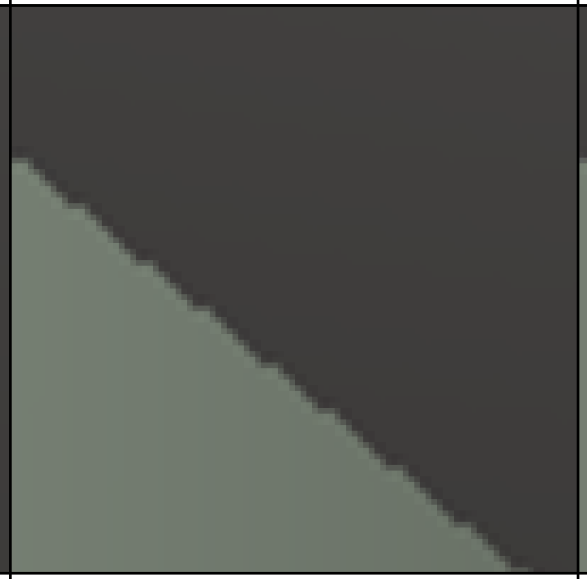
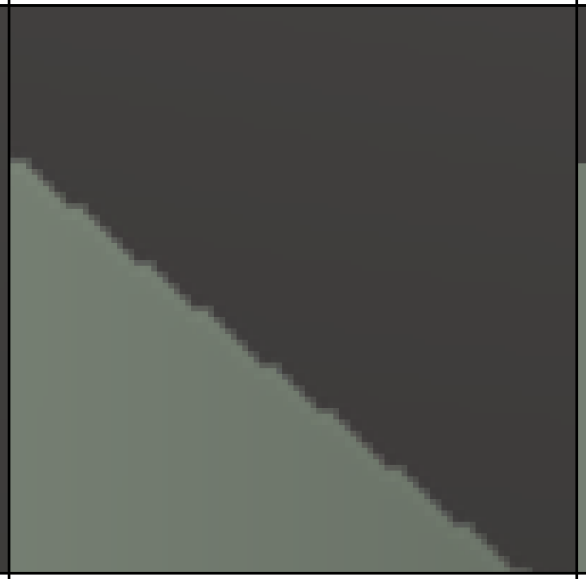
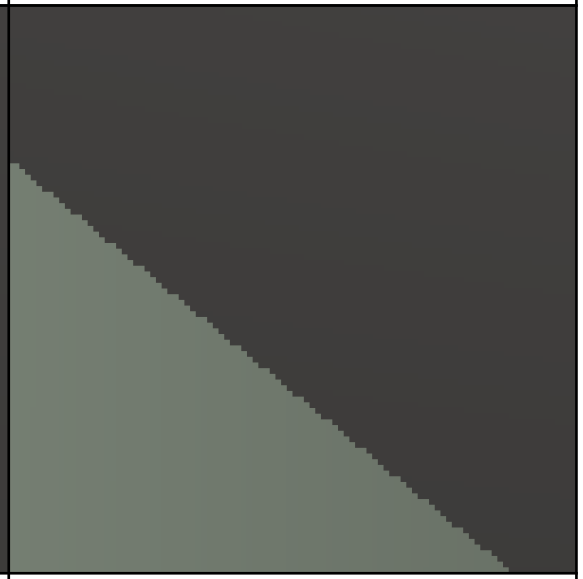
Bilateral method 분석

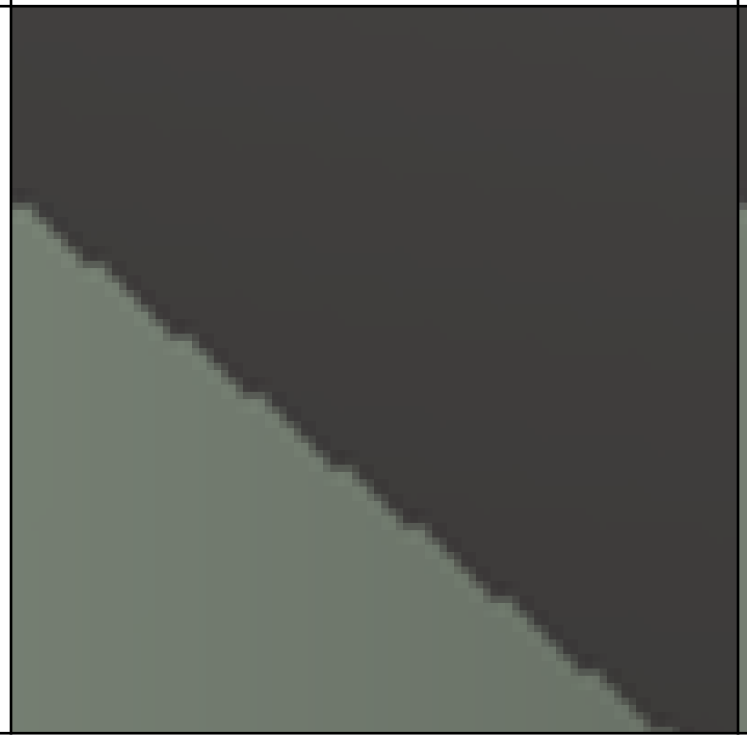
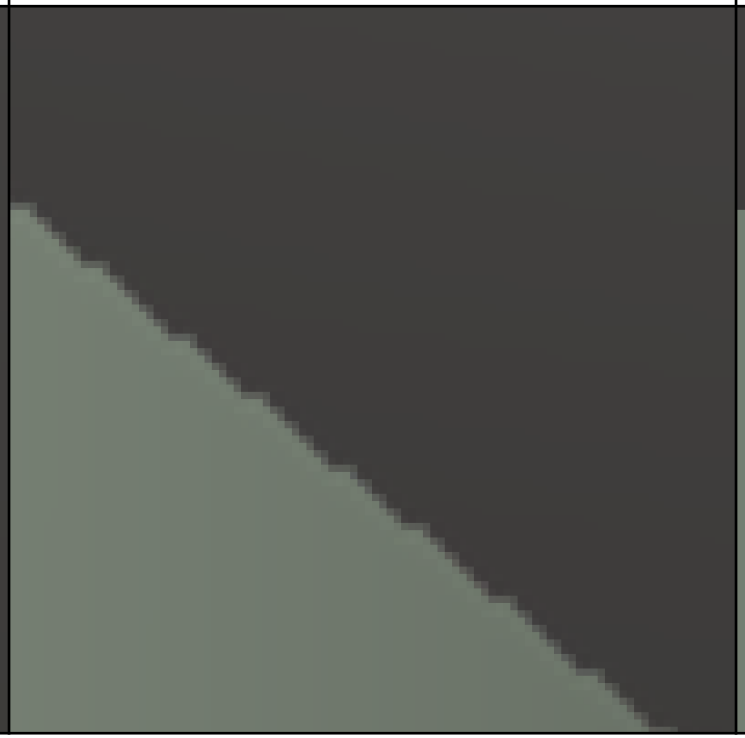
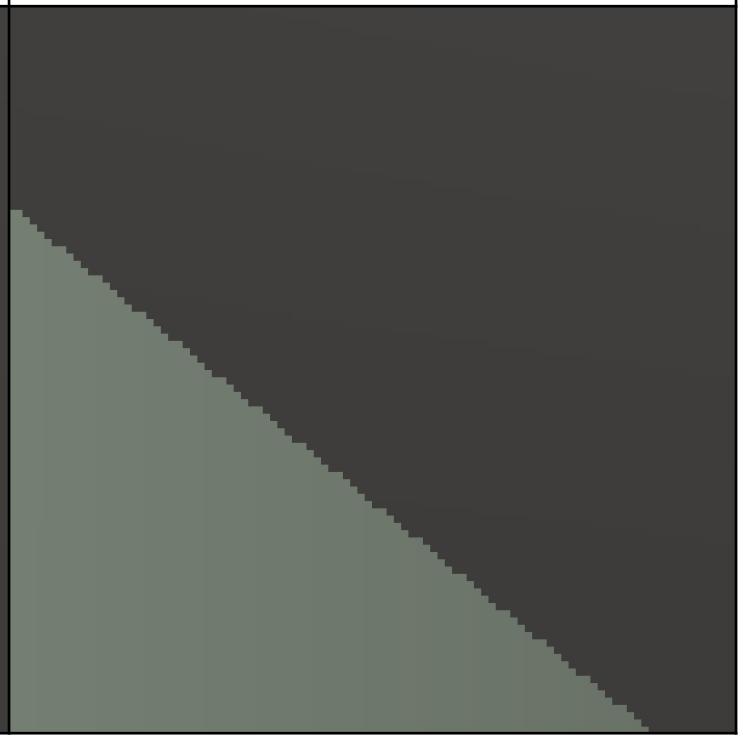
Python.ver

염지현

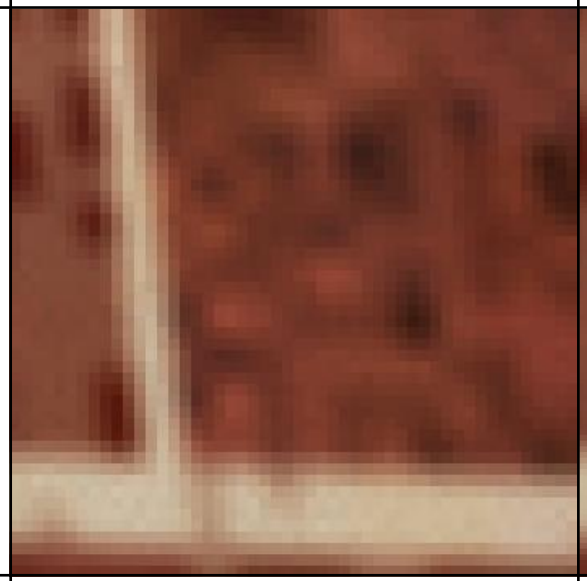
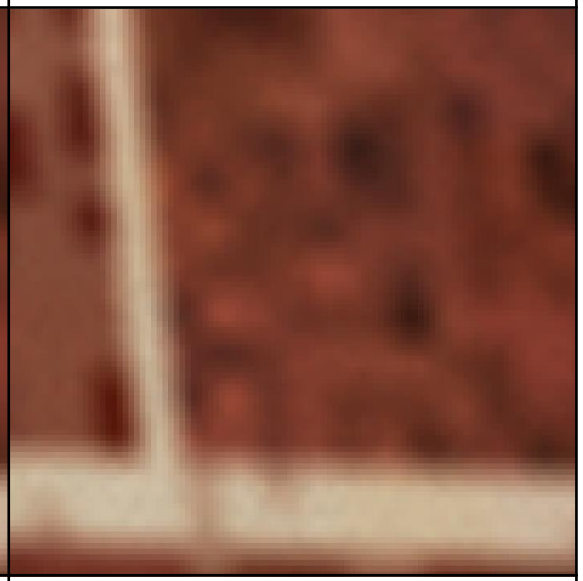
museum

Method	FHD	Bicubic	Bilateral	UHD
Image				
PSNR (RGB)	-	41.16	40.93	-
Time (초)	-	1006.49	3597.27 (Bicubic 제외)	-

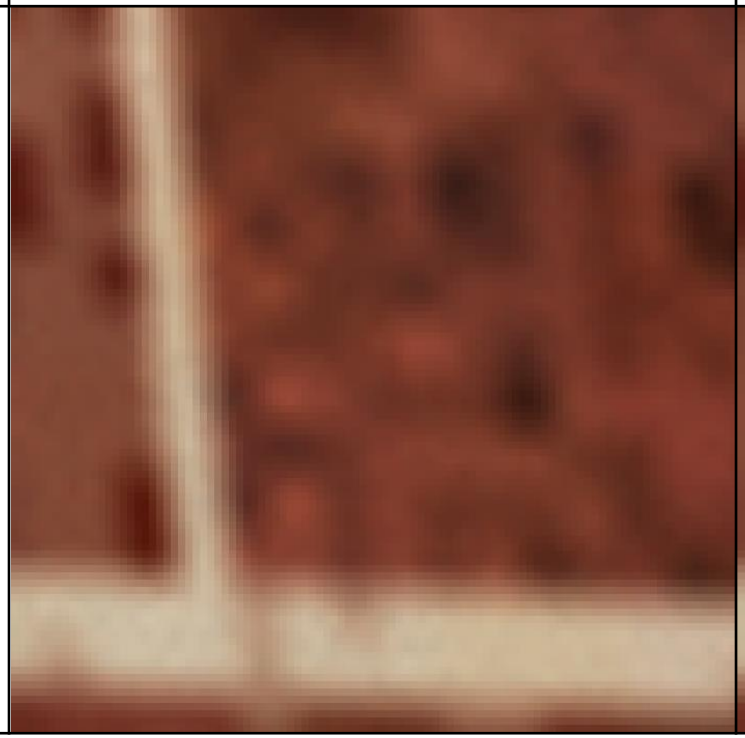
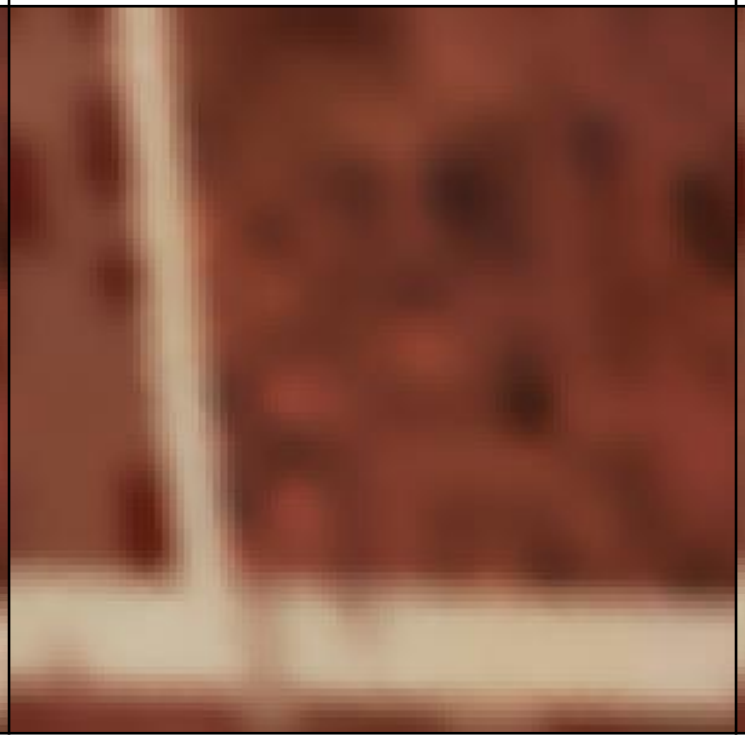
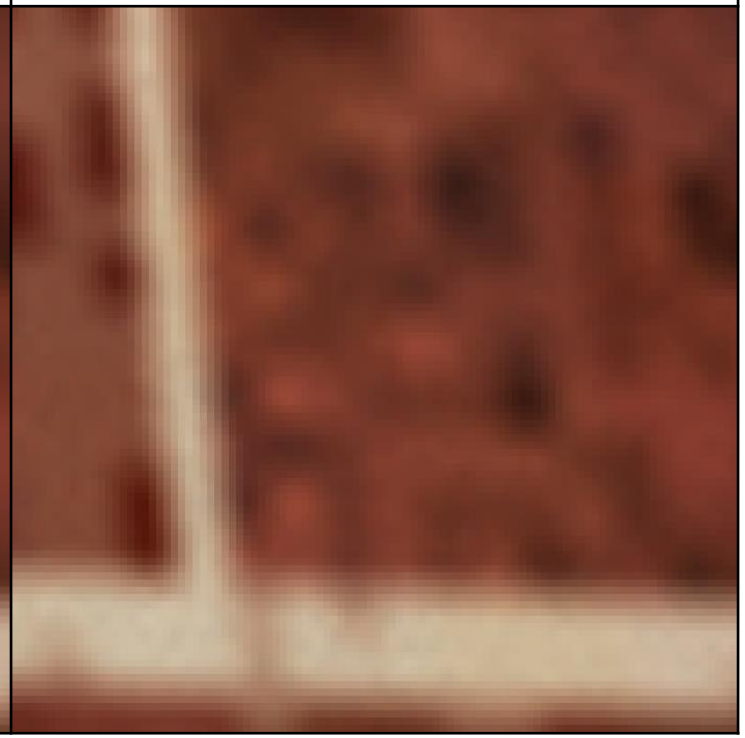
museum

Method	Bicubic	Bilateral	UHD
Image			

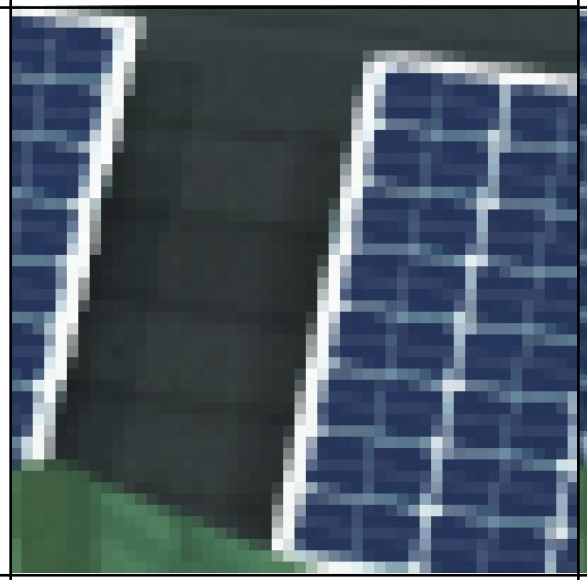
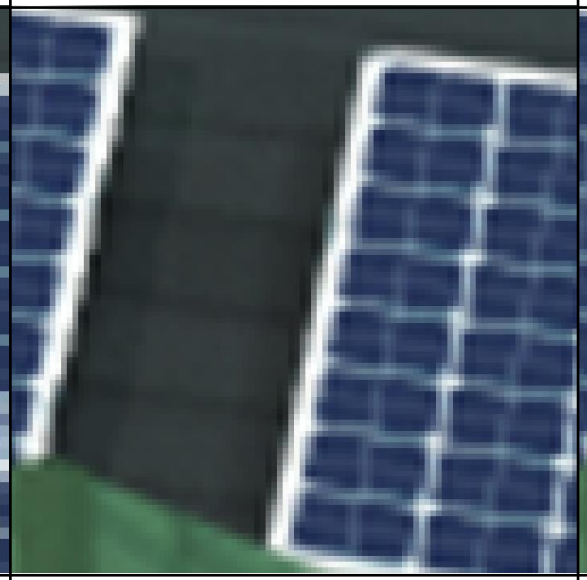
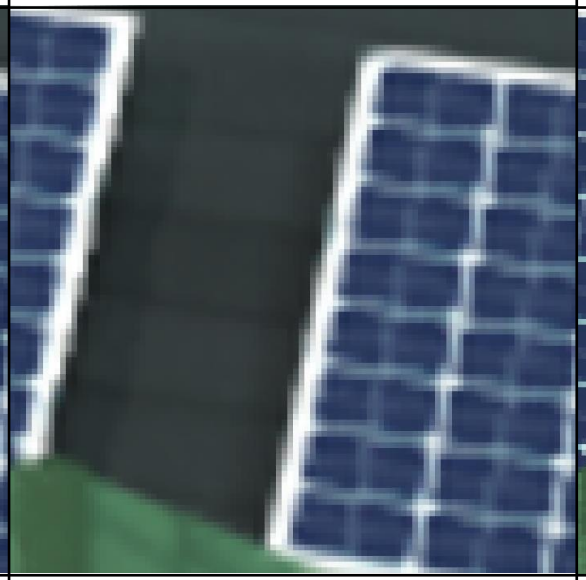
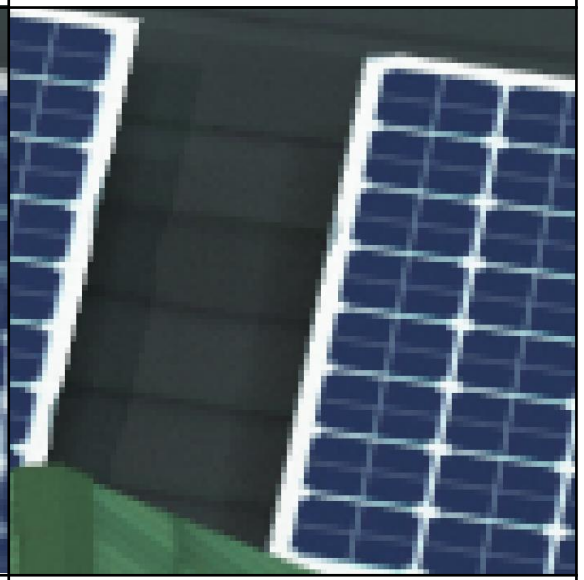
CafeInterior

Method	FHD	Bicubic	Bilateral	UHD
Image				
PSNR (RGB)	-	34.32	34.62	-
Time (초)	-	1006.49	3597.27 (Bicubic 제외)	-

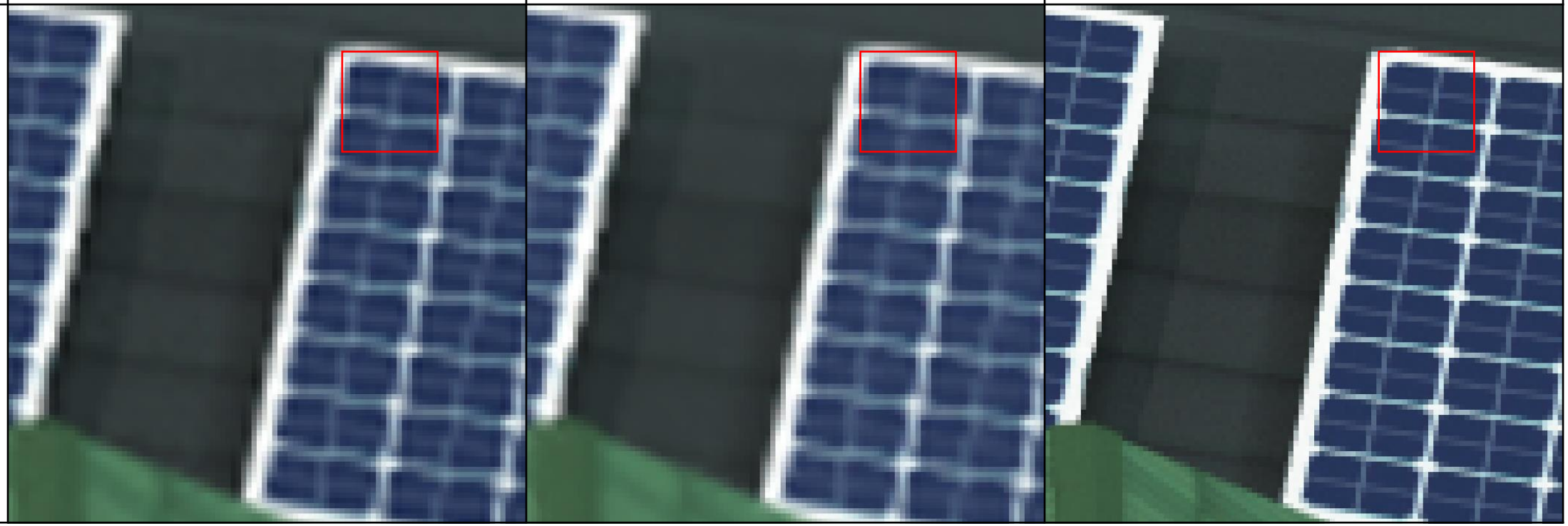
CafeInterior

Method	Bicubic	Bilateral	UHD
Image			

PolyTown

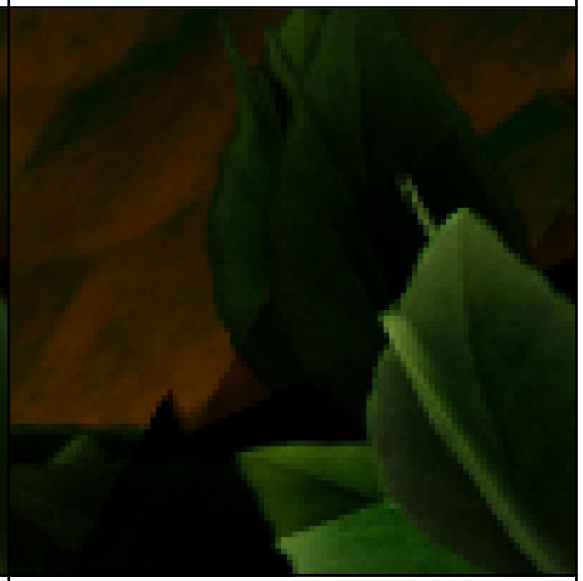
Method	FHD	Bicubic	Bilateral	UHD
Image				
PSNR (RGB)	-	38.30	37.66	-
Time (초)	-	1006.49	3597.27 (Bicubic 제외)	-

PolyTown

Method	Bicubic	Bilateral	UHD
Image			

* Bilateral 결과가 Bicubic보다 PSNR이 낮다고 분석한 이유:
Bilateral filter 자체가 노이즈를 제거를 목적으로 하므로 edge 부분 외에 smoothing을 해주는 filter라 정보를 많이 손실하게 된다고 추측

Village

Method	FHD	Bicubic	Bilateral	UHD
Image				
PSNR (RGB)	-	37.30	36.14	-
Time (초)	-	1006.49	3597.27 (Bicubic 제외)	-

Village

Method	Bicubic	Bilateral	UHD
Image			

* Bilateral 결과가 Bicubic보다 PSNR이 낮다고 분석한 이유:
Bilateral filter 자체가 노이즈를 제거를 목적으로 하므로 edge 부분 외에 smoothing을 해주는 filter라 정보를 많이 손실하게 된다고 추측

Bilateral filter 적용

In [7]:
```
import numpy as np
import cv2
import sys
import math


def distance(x, y, i, j):
    return np.sqrt((x-i)**2 + (y-j)**2)


def gaussian(x, sigma):
    return (1.0 / (2 * math.pi * (sigma ** 2))) * math.exp(- (x ** 2) / (2 * sigma ** 2))


def apply_bilateral_filter(source, filtered_image, x, y, diameter, sigma_i, sigma_s):
    hl = diameter/2
    i_filtered = 0
    Wp = 0
    i = 0
    while i < diameter:
        j = 0
        while j < diameter:
            neighbour_x = x - (hl - i)
            neighbour_y = y - (hl - j)
            if neighbour_x >= len(source):
                neighbour_x -= len(source)
            if neighbour_y >= len(source[0]):
                neighbour_y -= len(source[0])

            gi = gaussian(source[int(neighbour_x)][int(neighbour_y)] - source[x][y], sigma_i)
            gs = gaussian(distance(neighbour_x, neighbour_y, x, y), sigma_s)
            w = gi * gs
            i_filtered += source[int(neighbour_x)][int(neighbour_y)] * w
            Wp += w
            j += 1
        i += 1
    i_filtered = i_filtered / Wp
    filtered_image[x][y] = int(round(i_filtered))


def bilateral_filter_own(source, filter_diameter, sigma_i, sigma_s):
    s = time.time()
    filtered_image = np.zeros(source.shape)

    i = 0
    while i < len(source):
        j = 0
        while j < len(source[0]):
            apply_bilateral_filter(source, filtered_image, i, j, filter_diameter, sigma_i, sigma_s)
            j += 1
        i += 1
    print("bilateral time: ", time.time() - s)
    return filtered_image
```

In [26]:
```
import os
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
import torch

if torch.cuda.is_available():
    device = torch.device(f'cuda:0')
    print(f'# CUDA available: {torch.cuda.get_device_name(0)}')
else:
    device = 'cpu'


path = './sample/PolyTown/'
label_path = './sample/PolyTown/label/SR_Ref_spp32_camera'
img_list = os.listdir(path)

makedirs(path + 'bilateral_result/')

for img_name in img_list:
    if (img_name[-3:] == 'bmp'):
        # Read image
        _img = Image.open(path + img_name)
        img = np.array(_img)

        # Scale factor
        ratio = 2
        # Coefficient
        a = -1/2

#         dst = (bicubic(img, ratio, a))
        dst = Image.open(path+'org_result/result_'+img_name)

        d=psnr(np.array(Image.open(label_path + img_name[-5:])), np.array(dst))
        print("[BICUBIC] RGB채널 PSNR: ", d)

        dst = np.array(dst)
        filtered_image_own = np.zeros((2160, 3840, 3))
        filtered_image_own[:,:,0] = bilateral_filter_own(dst[:,:,0], 5, 12.0, 16.0)
        filtered_image_own[:,:,1] = bilateral_filter_own(dst[:,:,1], 5, 12.0, 16.0)
        filtered_image_own[:,:,2] = bilateral_filter_own(dst[:,:,2], 5, 12.0, 16.0)
        Image.fromarray(filtered_image_own.astype('uint8')).save(path + 'bilateral_result/' + 'result_' + img_name)
#         cv2.imwrite(path + 'bilateral_result/' + 'result_' + img_name, filtered_image_own)
        d=psnr(np.array(Image.open(label_path + img_name[-5:])), filtered_image_own)
        print("[BILATERAL] RGB채널 PSNR: ", d)
```