Worksheet 2 - Separability of filters, Median filters

Mark Belanger

October 4, 2024

Problem 1: Separability of filters - Show with an example that a Gaussian filter can be separable.

a) Choose an image

```
arcaid = imread("ARCAid.jpg");
imshow(arcaid);
title("ARCAid Sign")
```

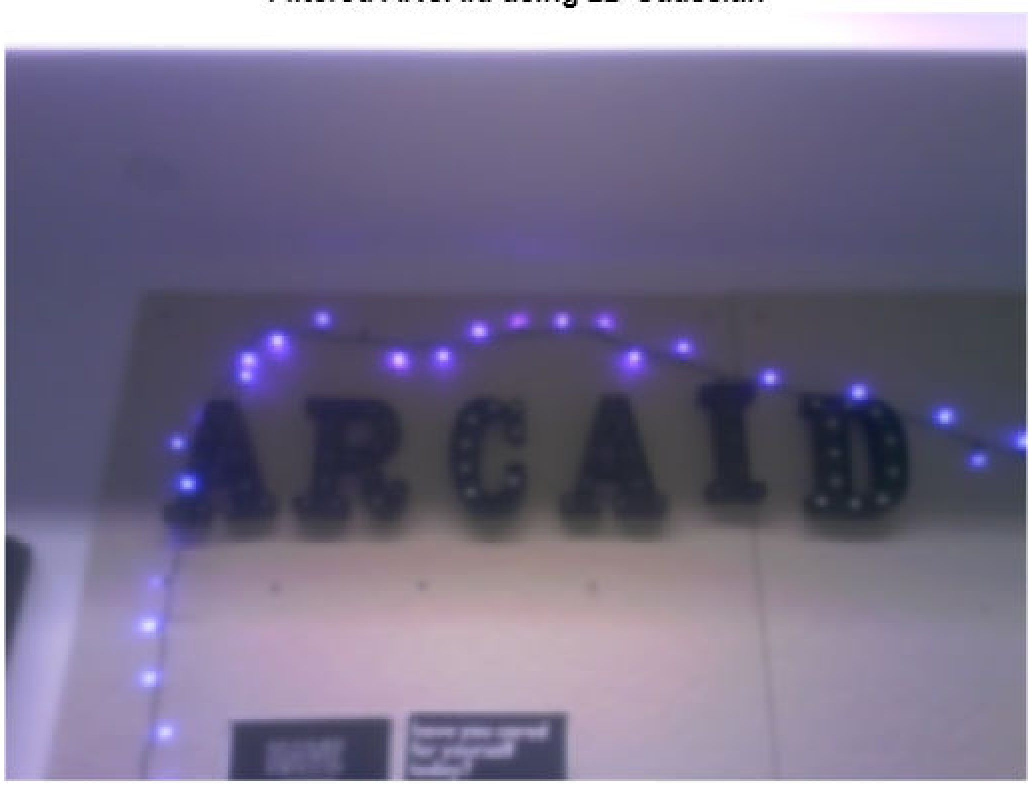
ARCAid Sign

b) Apply 2-dimensional Gaussian filter of your choice to it

```
sigma = 2;
filter_size = 6 * sigma;

[x, y] = meshgrid(-floor(filter_size/2):floor(filter_size/2), -floor(filter_size/
2):floor(filter_size/2));
```

```
two_dim_gauss = (1 / (2 * pi * sigma^2)) * exp(-(x.^2 + y.^2) / (2 * sigma^2));

two_dim_gauss = two_dim_gauss / sum(two_dim_gauss(:));

% red_arcaid = arcaid(:,:,1);
% green_arcaid = arcaid(:,:,2);
% blue_arcaid = arcaid(:,:,3);
%
% red_filtered = conv2(red_arcaid, two_dim_gauss);
% green_filtered = conv2(green_arcaid, two_dim_gauss);
% blue_filtered = conv2(blue_arcaid, two_dim_gauss);
%
% filtered_arcaid_2D = cat(3, red_filtered, green_filtered, blue_filtered);

filtered_arcaid_2D = imfilter(arcaid, two_dim_gauss, "replicate");

imshow(uint8(filtered_arcaid_2D))
title("Filtered ARCAid using 2D Gaussian")
```

Filtered ARCAid using 2D Gaussian



c. Separate the 2-dimensional Gaussian filter you used in 1(b) into two 1-dimensional filters

d. Apply the two 1-d filters to your image

```
sigma = 2;
```

```
filter_size = 6 * sigma;

filter_range = -filter_size/2:filter_size/2;
one_dim_gauss = (1/(sqrt(2*pi)*sigma)) * exp(-filter_range.^2 / (2*sigma^2));
one_dim_gauss = one_dim_gauss / sum(one_dim_gauss(:));

filtered_rows = imfilter(arcaid, one_dim_gauss, "replicate");

filtered_arcaid_1D = imfilter(filtered_rows, one_dim_gauss', "replicate");

imshow(filtered_arcaid_1D)
title("Filtered ARCAid using Separated 1D Gaussian");
```

Filtered ARCAid using Separated 1D Gaussian

e. Show that results of 1(b) and 1(d) are the same.

```
figure
subplot(1, 2, 1)
imshow(uint8(filtered_arcaid_2D))
title("2D Process")
subplot(1, 2, 2)
imshow(uint8(filtered_arcaid_1D))
title("1D Process")
```

2D Process

1D Process

The images look identical on visual inspection but we can analyze the pixel values and compare aswell.

```
result = mean(filtered_arcaid_1D == filtered_arcaid_2D, "all") * 100
```

```
result = 91.4561
```

This is comparing all the indices where the matrixes are equivalent and it indicates that they are about 91% the same. The differences are likely due to the conversions between double and uint8 in addition to the different padding that can result.

Problem 2: Median – write code in MATLAB or Python to:

a. Create a random 3x3 matrix

```
random_matrix = 100 * rand(3)
```

```
random_matrix = 3×3
   70.9365   67.9703   11.8998
   75.4687   65.5098   49.8364
   27.6025   16.2612   95.9744
```

b. Sort the elements of this matrix

```
random_matrix_reshaped = reshape(random_matrix, 1,size(random_matrix, 1) *
size(random_matrix, 2))
```

```
random_matrix_reshaped = 1×9
   70.9365   75.4687   27.6025   67.9703   65.5098   16.2612   11.8998   49.8364 ···
```

```
random_matrix_sorted = sort(random_matrix_reshaped)
```

```
random_matrix_sorted = 1×9
   11.8998   16.2612   27.6025   49.8364   65.5098   67.9703   70.9365   75.4687 ⋯
```

c. Take median of the matrix

```
median_num = median(random_matrix_sorted)
```

```
median_num = 65.5098
```

d. Assign the median to the middle location of the matrix

```
random_matrix(2, 2) = median_num;

random_matrix
```

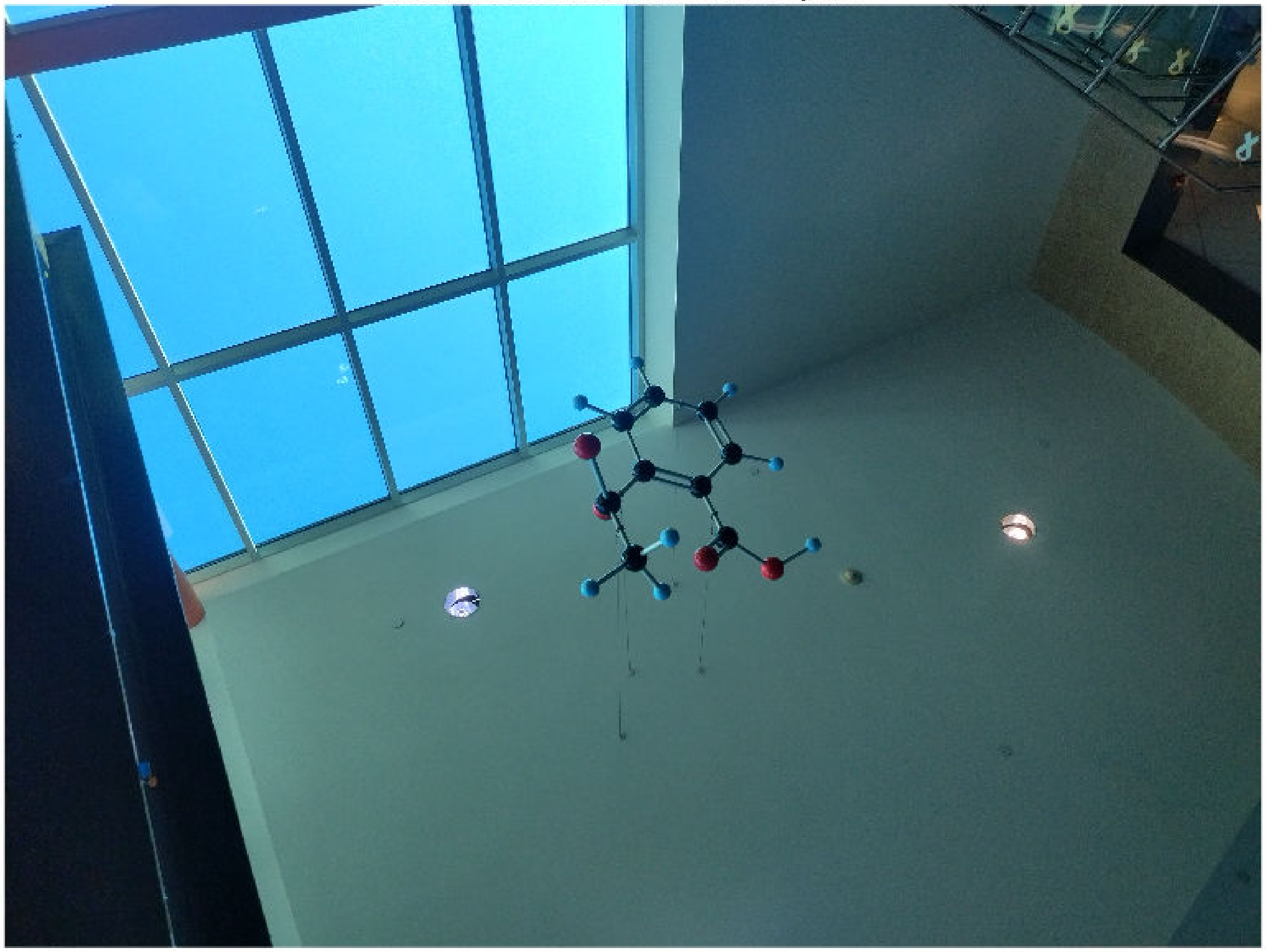
```
random_matrix = 3×3
   70.9365   67.9703   11.8998
   75.4687   65.5098   49.8364
   27.6025   16.2612   95.9744
```

Problem 3: Just playing with different noise in MATLAB/Python:

a. Choose an image

```
molecule = imread("molecule.jpg");
figure
imshow(molecule)
title("A Picture I Took of a Molecule Sculpture ")
```

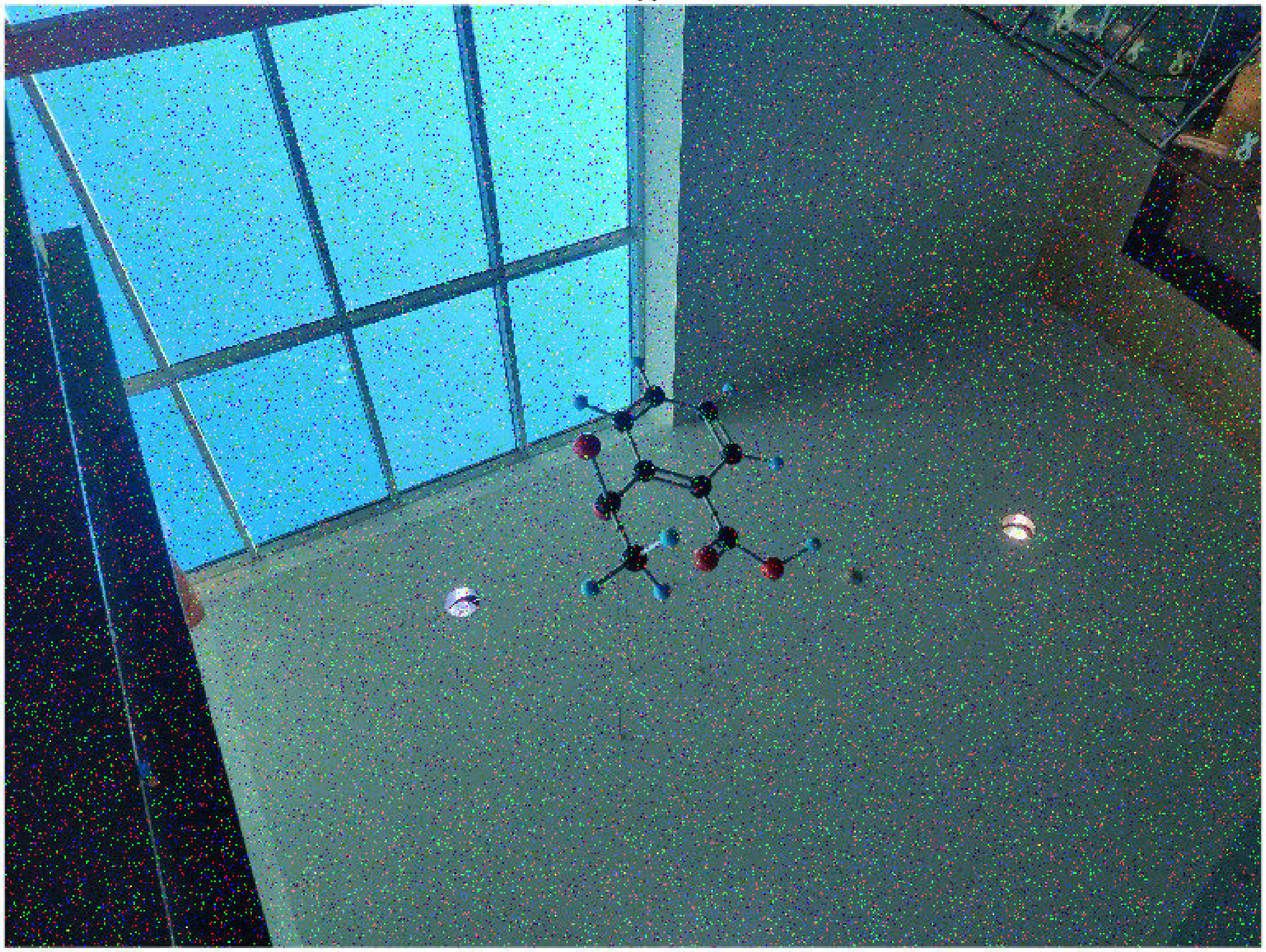
A Picture I Took of a Molecule Sculpture



b. Add any 3 types of noise to it – show each as a separate image. (You can use the imnoise function in MATLAB, or its equivalent in Python).

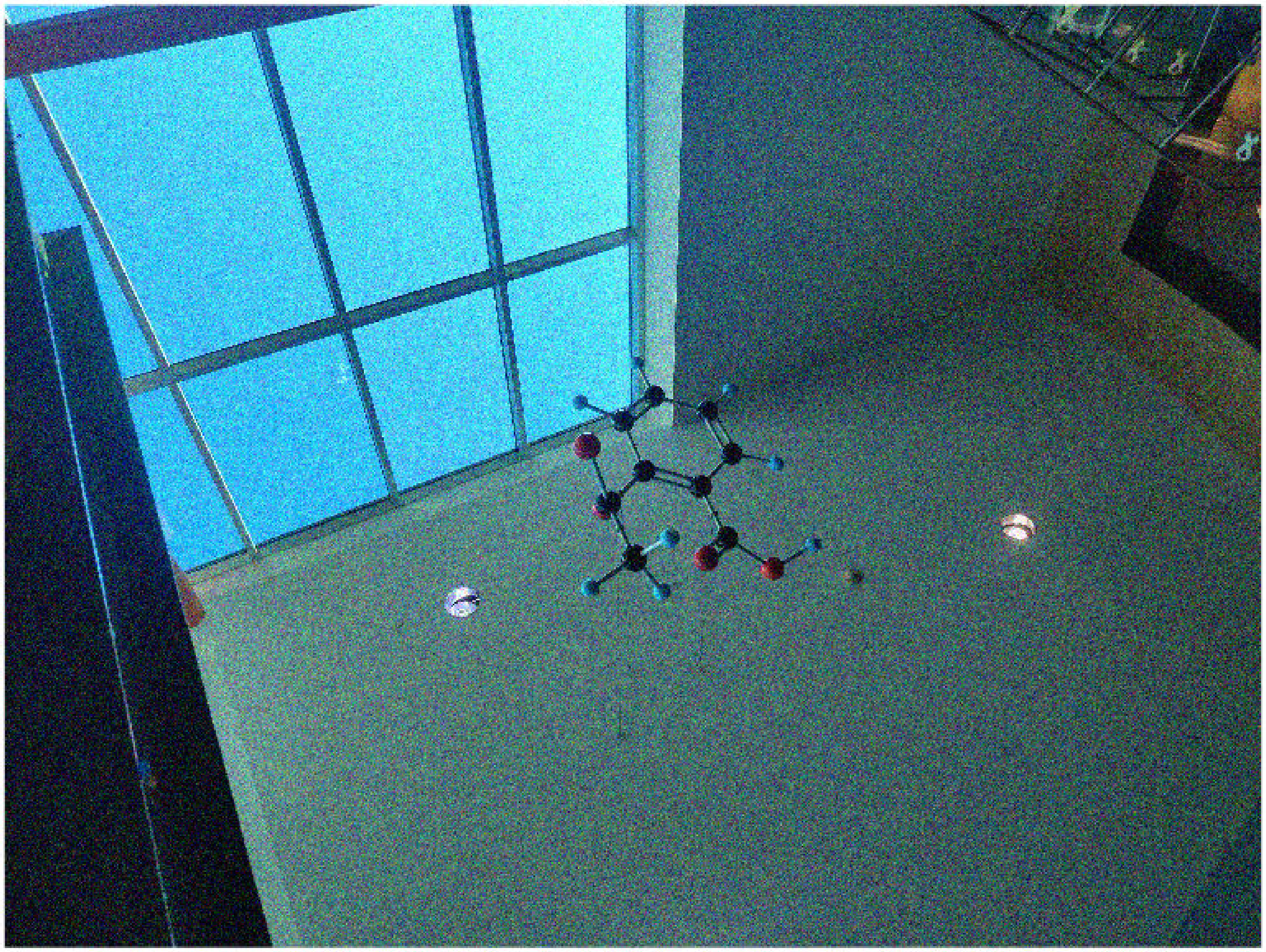
```
molecule_sp = imnoise(molecule, "salt & pepper");
figure
imshow(molecule_sp);
title("Salt and Pepper Noise")
```

Salt and Pepper Noise



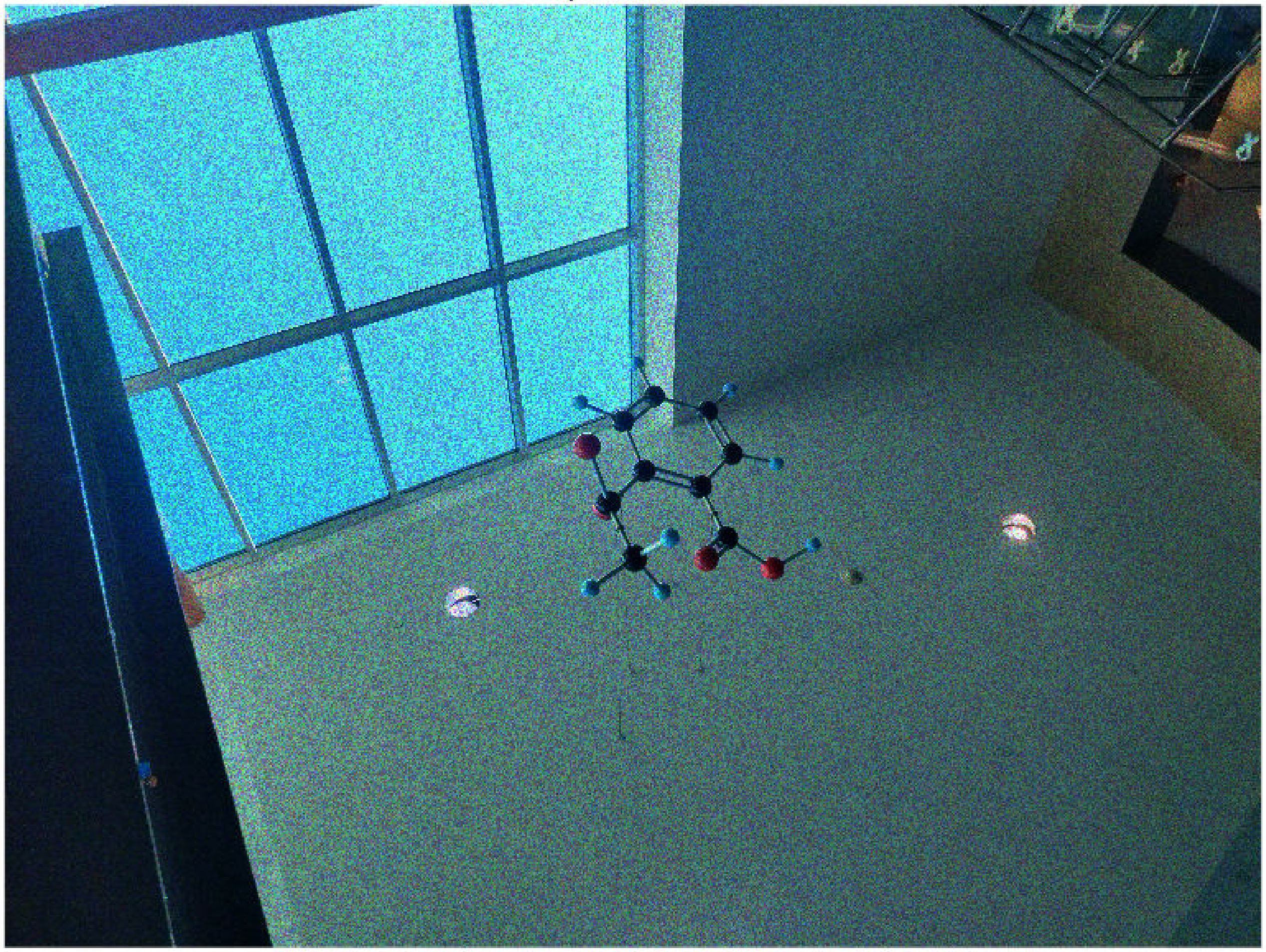
```
molecule_gauss = imnoise(molecule, "gaussian");
figure
imshow(molecule_gauss);
title("Gaussian Noise")
```

Gaussian Noise



```
molecule_speckle = imnoise(molecule, "speckle");
figure
imshow(molecule_speckle)
title("Speckle Noise")
```

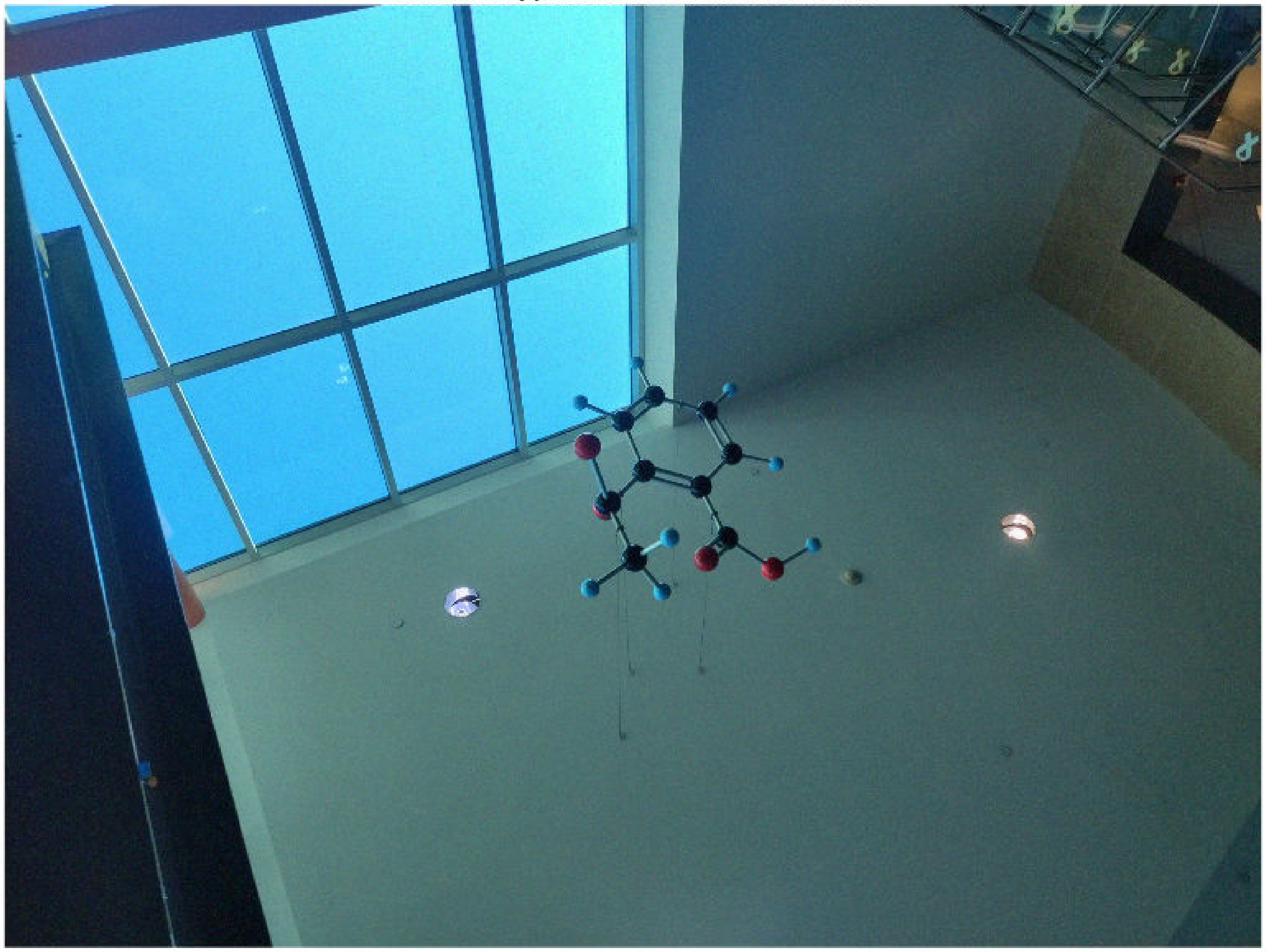
Speckle Noise



c. Try Gaussian filtering on these images. (You can use imgaussfilt function in MATLAB, or its equivalent in Python). Show your results

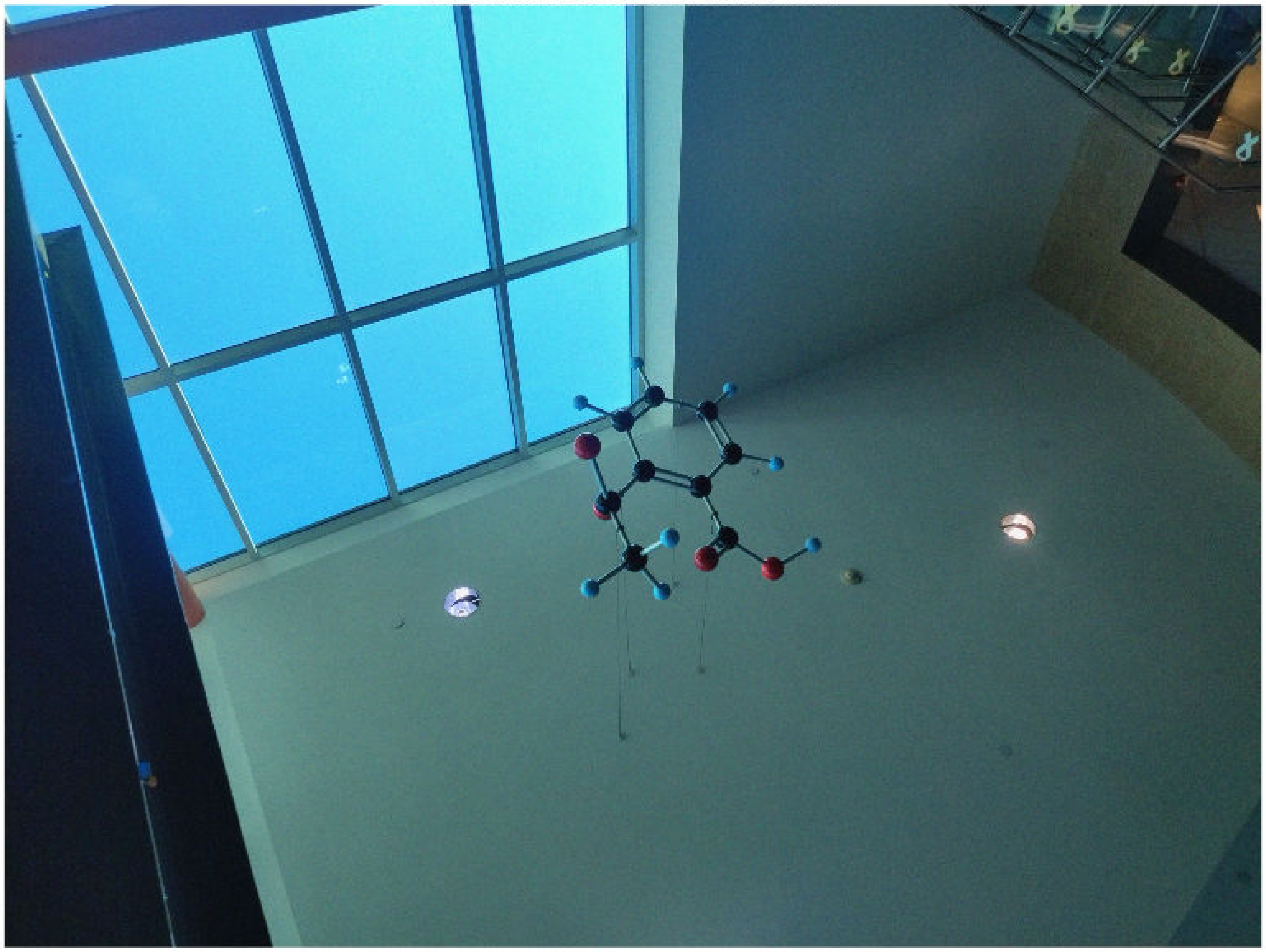
```
sigma = 2;

molecule_sp_gauss = imgaussfilt(molecule_sp, sigma);
figure
imshow(molecule_sp_gauss);
title("Salt and Pepper Noise with Gaussian Filter")
```

Salt and Pepper Noise with Gaussian Filter



```
molecule_gauss_gauss = imgaussfilt(molecule_gauss, sigma);
figure
imshow(molecule_gauss_gauss);
title("Gaussian Noise with Gaussian Filter")
```

Gaussian Noise with Gaussian Filter



```
molecule_speckle_gauss = imgaussfilt(molecule_speckle, sigma);
figure
imshow(molecule_speckle_gauss);
title("Speckle Noise with Gaussian Filter")
```

Speckle Noise with Gaussian Filter

The filtering did a very good job of removeing the noise. I arbitrarily chose a σ value of 2 and this value removed almost all the noise without noticeably decreasing the quality of the photo. Some noise is still visable when looking closely at the window in the picture. Using a larger value such as 20 does blur the image though. This is a large image compared to images we worked with in the past so this may affect the results.