Numerical Analysis

Computational Project

Noe Lomidze

Problem 2.1

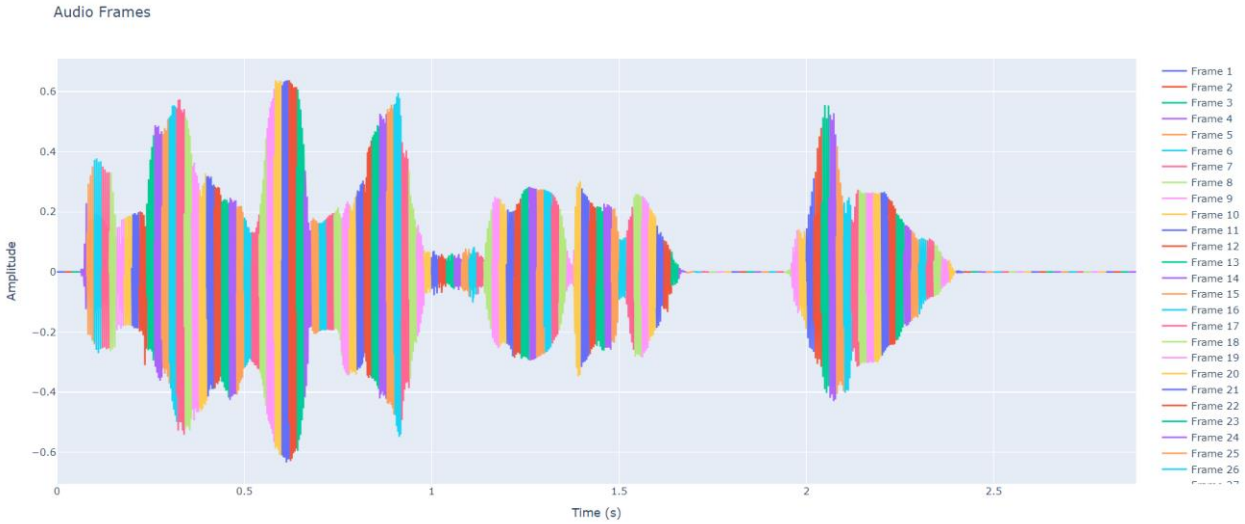
Interpolation, sound wave (6 points)

1. Consider sound as a univariate function. For example you can write text and convert it to speech. Consider application of libraries (from gtts import gTTS , import librosa). You may need to cut sound in 20 milisecond frames.
2. Explore the potential of three different interpolation methods for sound representation. Describe the methods used and justify your approach.
3. Present visual examples to illustrate both successful and unsuccessful findings.

Please use a separate slide for each of the sub-tasks mentioned above.

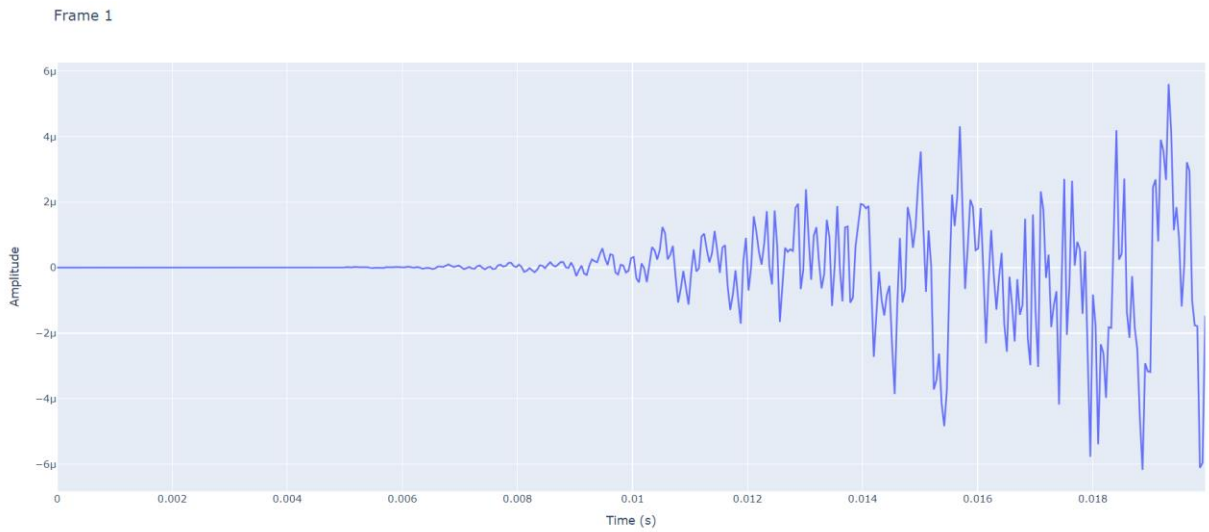
So our goal was something like that:
firstly I had to create a text to speech, but that was easy because the libraries were provided.

Then I had to choose the interpolation methods to apply to the sound function and I chose linear, quadratic and cubic for simplicity

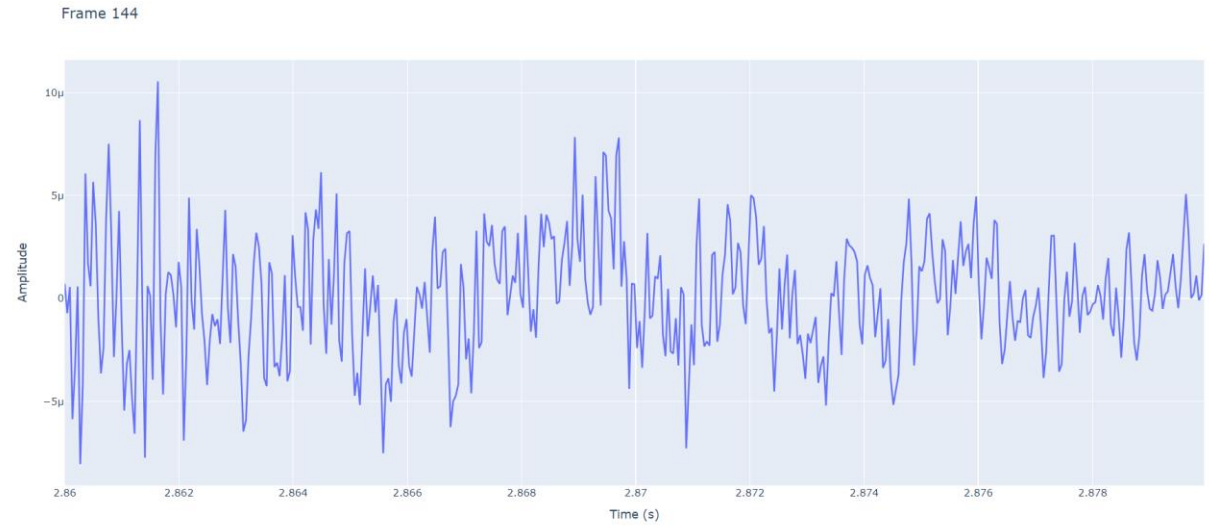


Therefore, this is the sound but cut in 20 millisecond frames as it was in the task.

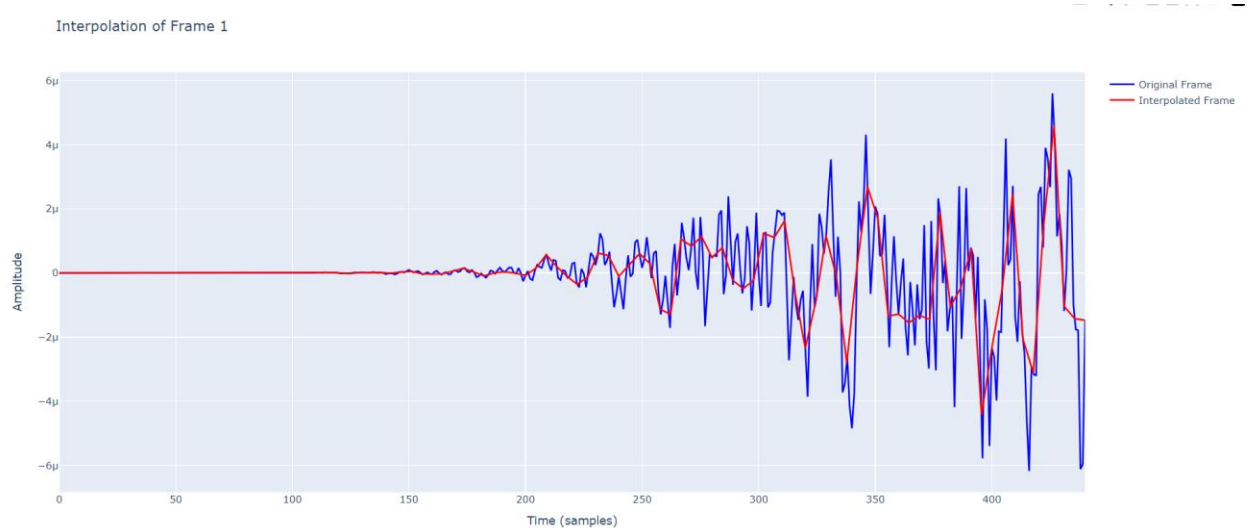
In my case the sound was 2.8 seconds long, so I got approximately 144 frames.



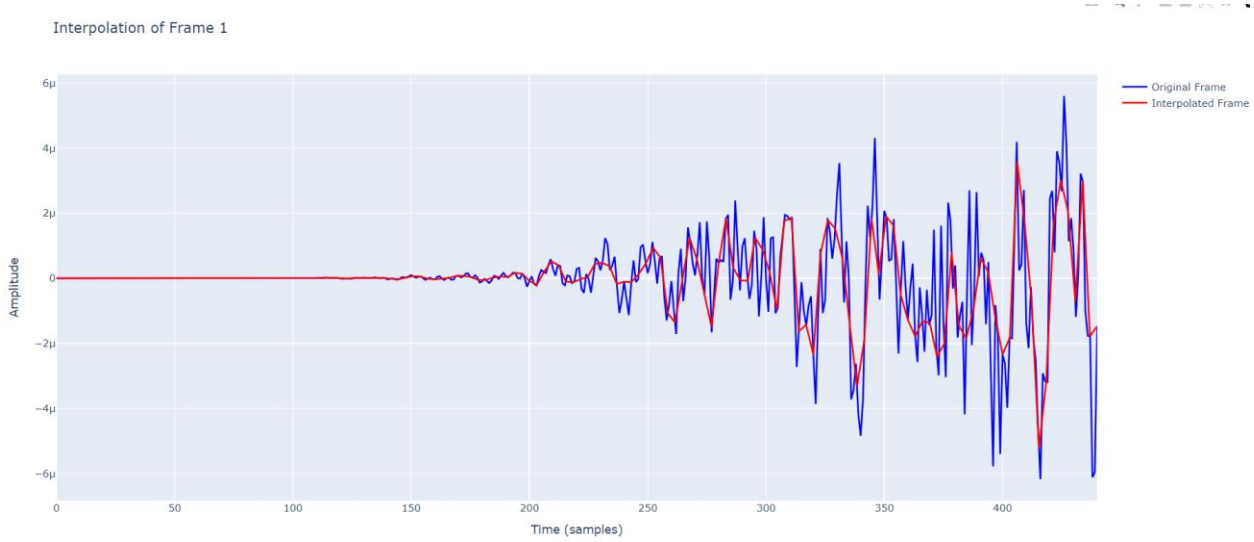
This is how the frame 1 looks like



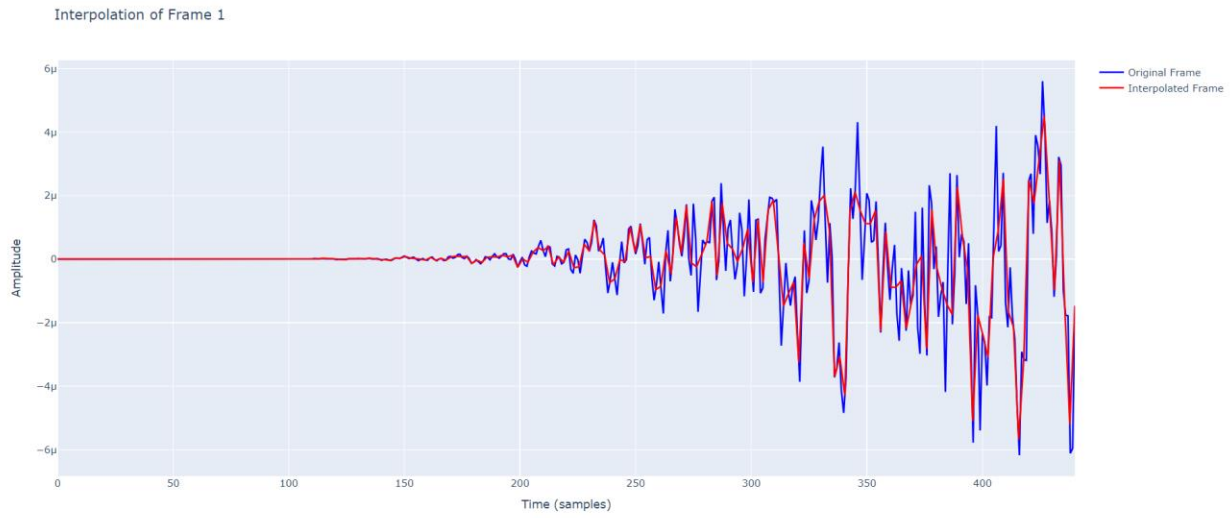
This is how the last frame looks like



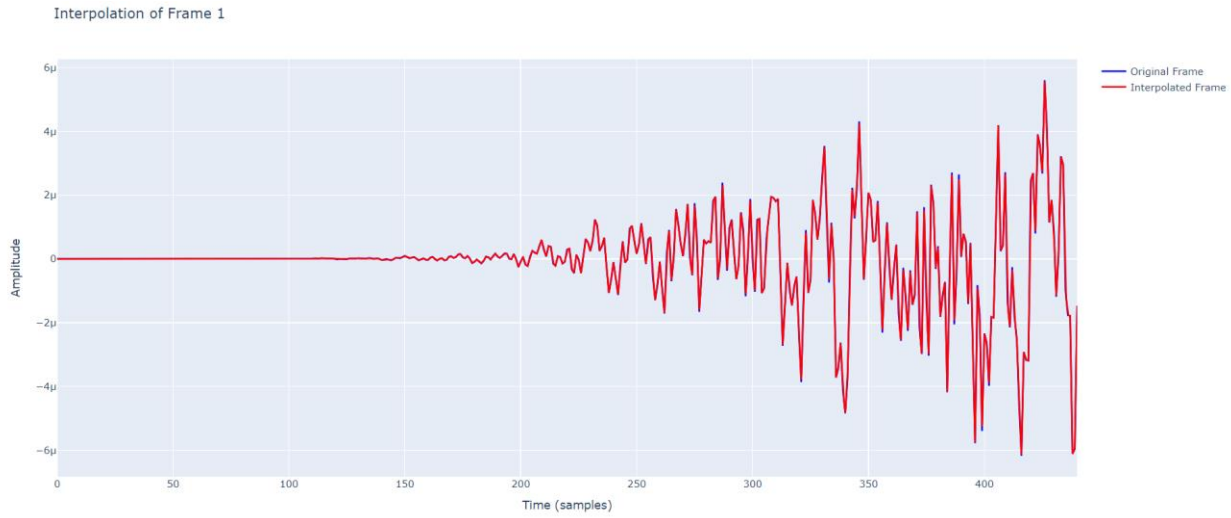
This is how the interpolated first frame looks like, using linear interpolation, with 400 control points



Here are about 550 control points, which as you see made a difference from the previous one

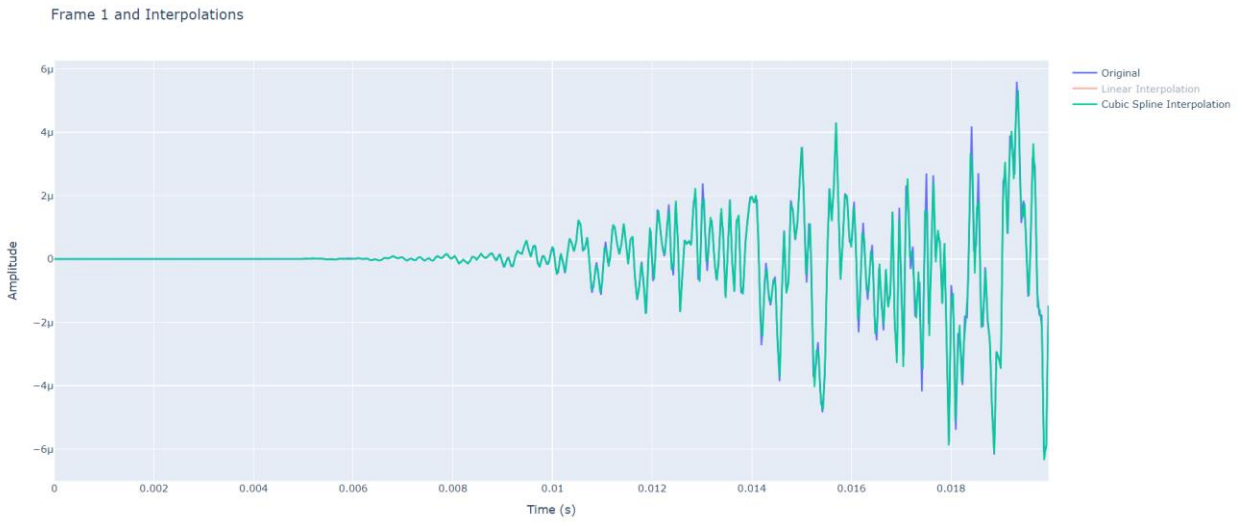


Here are even more control points, approximately 1000 or 1200

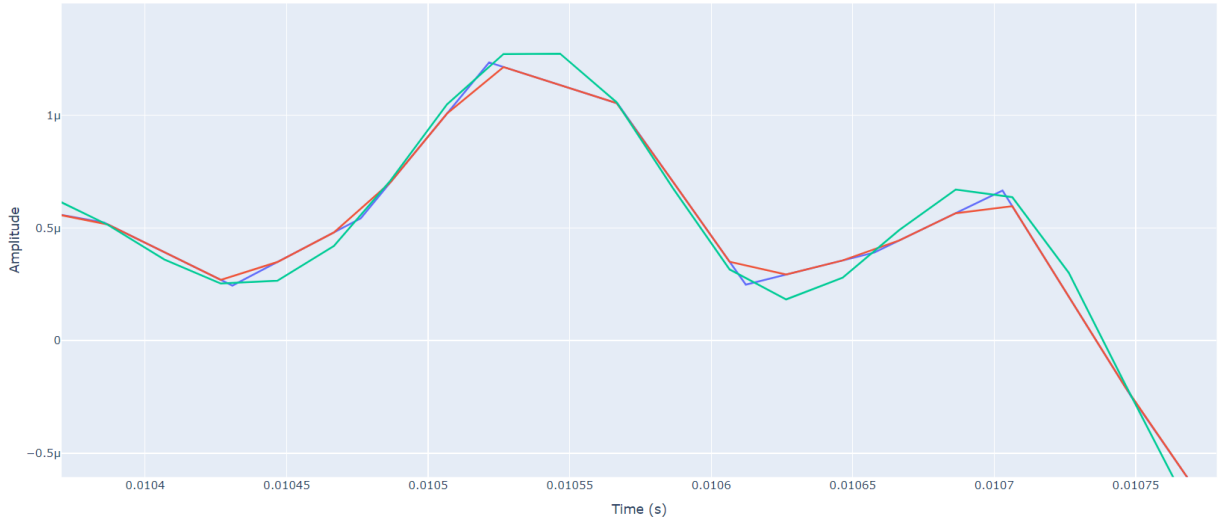


Here are twice as much points as in the previous one, 2000. Which really did its job correctly

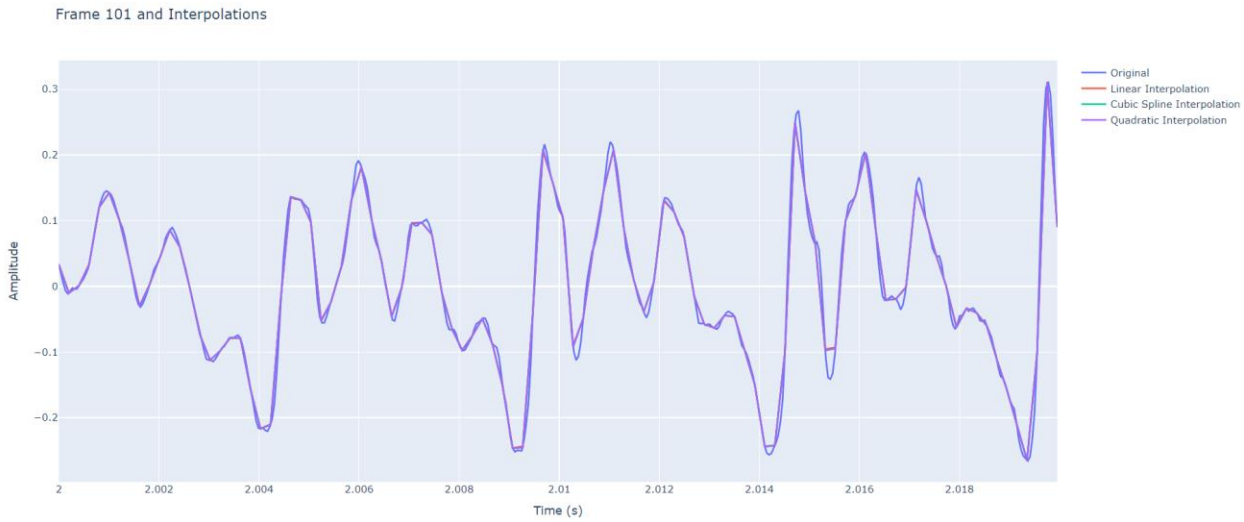
So now lets move to the cubic interpolation

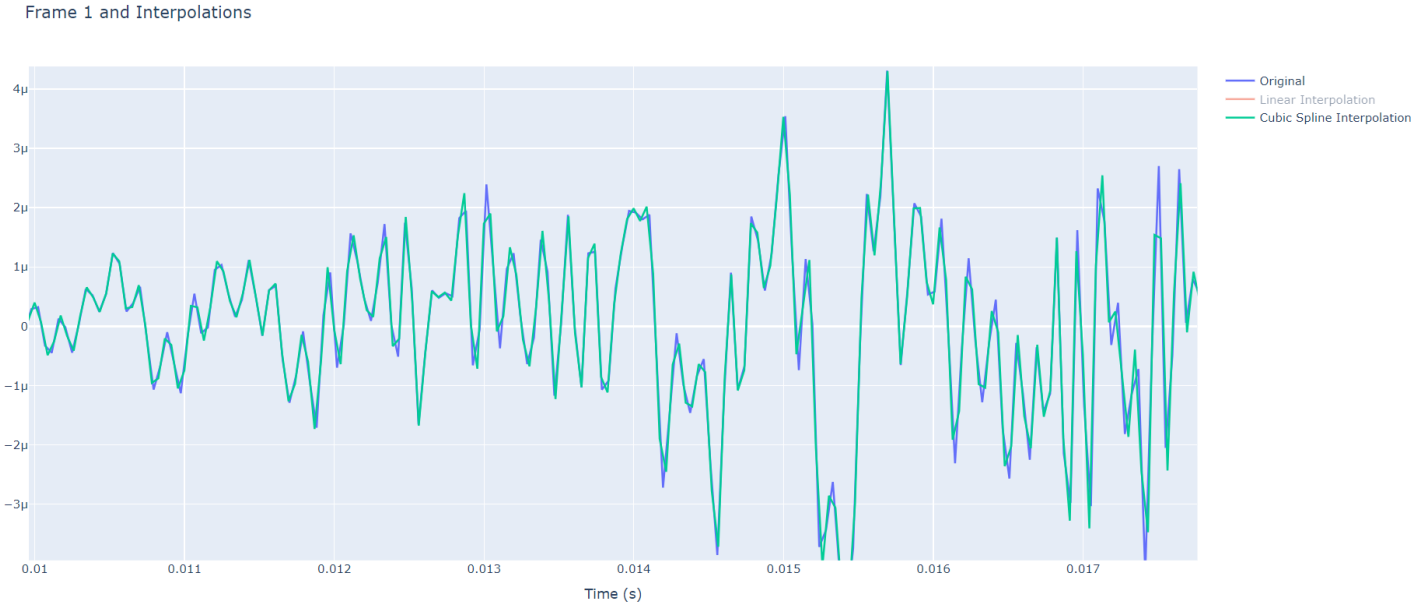


As you can see, it works way better than others

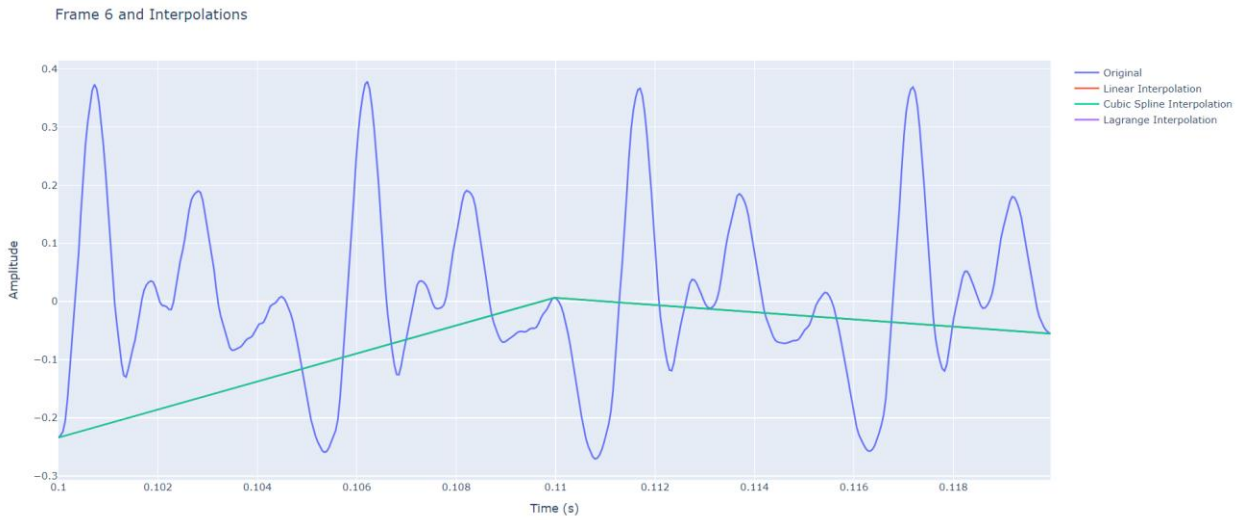


Here you can see the difference between those lines





The picture below can be seen as an unsuccessful founding



If we choose less control points, we get the bad result as you can see

Linear Interpolation Method:

Connects two known points with a straight line. Formula:

$$y = y_0 + \frac{(x - x_0)(y_1 - y_0)}{x_1 - x_0}$$

Use Case: Simple and fast, used when data points are close and the change between them is approximately linear.

Quadratic Interpolation Method:

Uses a quadratic polynomial (parabola) passing through three known points. Formula:

$$y = a * x^2 + b * x + c$$

Where a, b, and c are determined based on the three points.

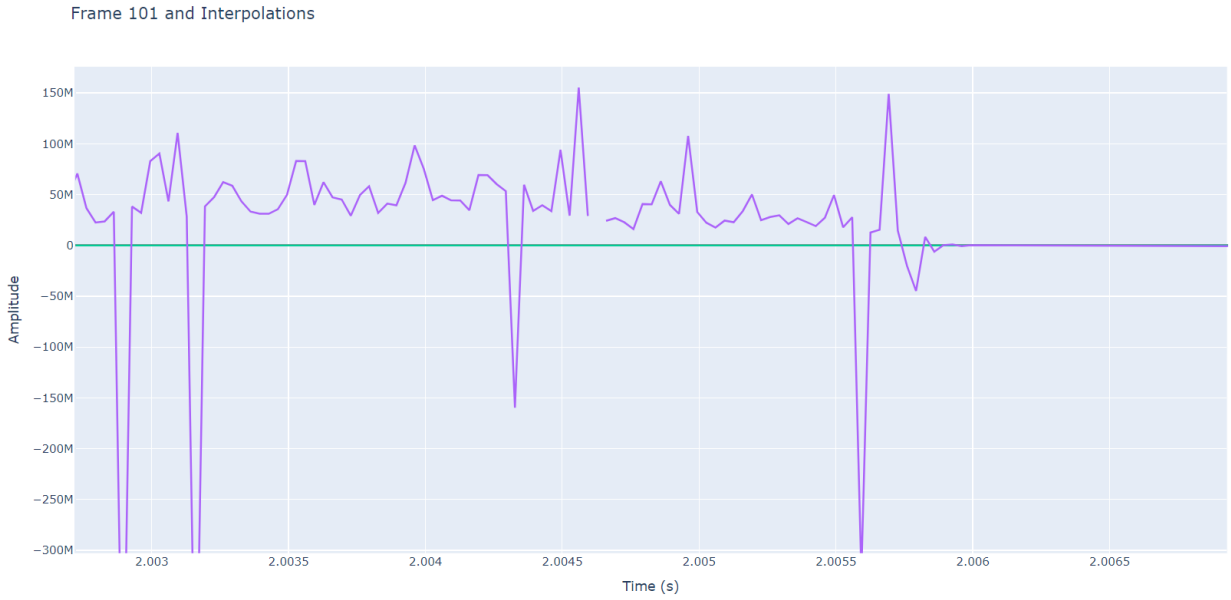
Use Case: Better for capturing curvature than linear interpolation, suitable for moderately curved data.

Cubic Interpolation Method: Uses a cubic polynomial passing through four known points. Formula:

$$y = a * x^3 + b * x^2 + c * x + d$$

Where a, b, c, and d are determined based on the four points.

Use Case: Captures curvature that is more complex and is smooth at the connections between segments, making it ideal for smoothly varying data.



Lastly I also tried Lagrange interpolation and again got unsuccessful founding for some reason

The sound files will be provided inside the zip

So now let's move to another task

Problem 2.2

Digital Images, radial basis functions, compression (6 points)

1. Consider digital images. remove some $(10\%, 25\%, 50\%, 75\%)$ pixels randomly.
2. Explore the potential of three different radial basis functions for restoring digital image. consider adjusting of shape parameter. Describe the method used and justify your approach.
3. Present visual examples to illustrate both successful and unsuccessful findings.

Please use a separate slide for each of the sub-tasks mentioned above.

Radial Basis Functions (RBFs)

are a class of mathematical functions whose value depends only on the distance from a center or reference point. These functions are commonly used in various fields, including mathematics, computer science, and engineering, due to their flexibility and ability to approximate complex relationships. The general form of an RBF can be written as:

ϕ (r) where r is the distance from the center or reference point, and ϕ is the RBF itself

There are several types of RBFs commonly used in practice. Some of the most popular ones include:

Gaussian RBF: The Gaussian RBF is perhaps the most well-known type of RBF. It is defined as:

$$\varphi(r) = e^{(-\varepsilon r)^2}$$

where ϵ is a shape parameter that controls the spread or width of the Gaussian curve. Increasing ϵ leads to a wider curve, while decreasing it results in a narrower curve.

Multiquadric RBF: The Multiquadric RBF is defined as:

$$\varphi(r) = \sqrt{1 + (\varepsilon r)^2}$$

Similar to the Gaussian RBF, ϵ controls the shape of the curve. However, the Multiquadric RBF has a different functional form, resulting in slightly different interpolation behavior compared to the Gaussian RBF.

Inverse Multiquadric RBF: This RBF is defined as:

$$\varphi(r) = \frac{1}{\sqrt{1 + (\varepsilon r)^2}}$$

Like the Multiquadric RBF, it is controlled by a shape parameter ϵ. However, it behaves differently from the Multiquadric RBF due to its inverse relationship with the distance r.

Introduction to Radial Basis Functions Definition:

An RBF is a real-valued function whose value depends only on the distance from a center point, typically denoted as

$$\varphi(\|x - c\|)$$

where x is the input variable, c is the center, and ϕ is the radial basis function.

Properties and Characteristics:

Radial Symmetry: The function's value depends solely on the distance from a center point, making it symmetric around the center.

Flexibility: RBFs can model complex and non-linear relationships due to their non-linear nature.

Smoothness: Many RBFs, such as Gaussian and Multiquadric, are infinitely differentiable, providing smooth interpolations.

Now lets talk about applications

Interpolation: RBFs are widely used for interpolating scattered data points in multi-dimensional space. They provide a smooth approximation that passes through the given data points.

Machine Learning: In neural networks, RBFs are used as activation functions in Radial Basis Function Networks (RBFNs), which are used for classification and regression tasks.

Image Processing: RBFs are used for tasks like image reconstruction, inpainting, and warping due to their ability to handle spatial relationships and interpolate missing or damaged parts of images.

Surface Reconstruction: RBFs are utilized in reconstructing surfaces from point clouds in computer graphics and 3D modeling.

Mathematical Formulation

RBF Interpolation: Given a set of data points, the goal is to find a function such that

$$s(x_i) = y_i$$

The interpolant s (x) s(x) is often formulated as:

$$s(x) = \sum w_i \varphi(\|x - x_i\|)$$

where w are the weights determined by solving the system of linear equations derived from the interpolation conditions.

Advantages and Disadvantages

Advantages:

Flexibility and Smoothness: Can model complex relationships smoothly.
Dimensionality: Handles high-dimensional data naturally.

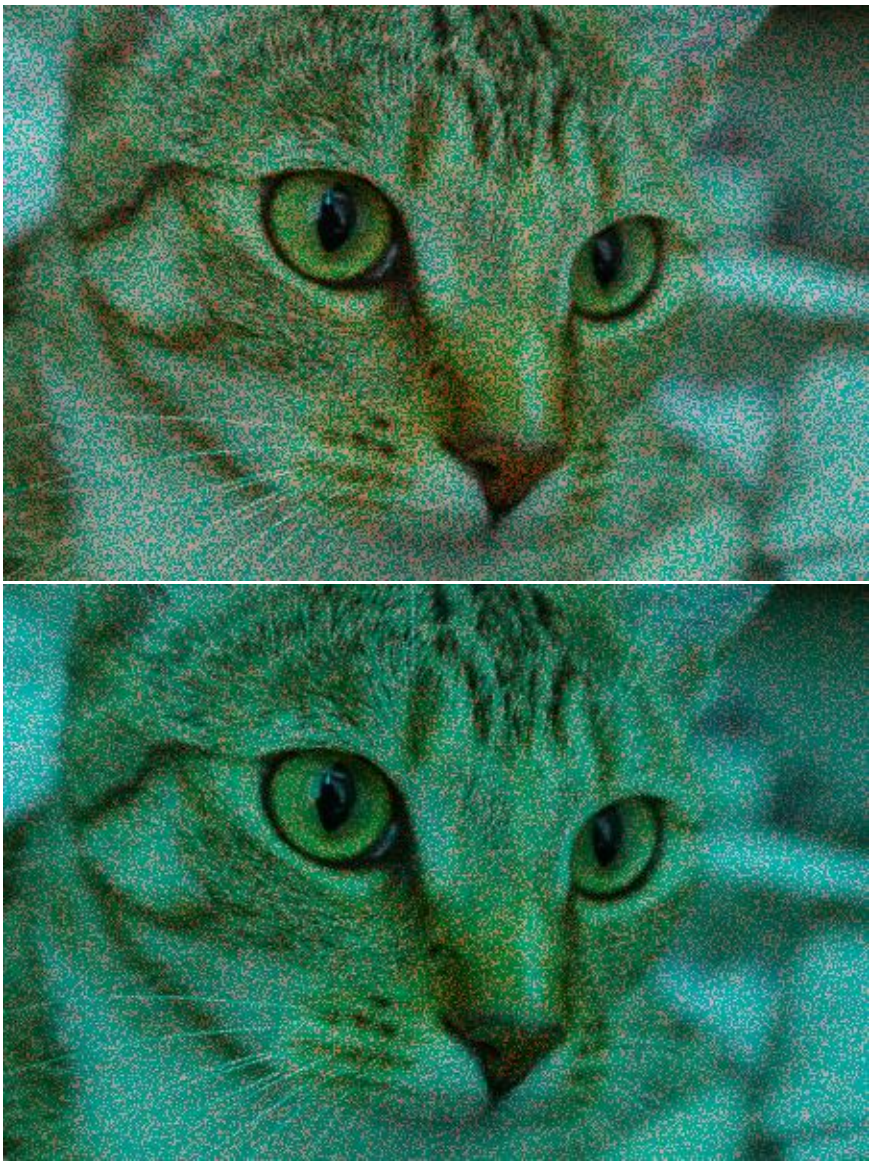
Locality: Some RBFs have local support, meaning changes in input affect only nearby outputs, which can be beneficial for certain applications.

Disadvantages:

Computational Cost: Solving the linear system can be computationally expensive for large datasets.

Scalability: The performance and memory requirements can become prohibitive as the number of data points increases.

Now let us get into work, firstly remove 10%, 25%, 50%, 75% of the pixels and then apply those three methods:



As you can see the pixels get removed so now lets start to apply the methods

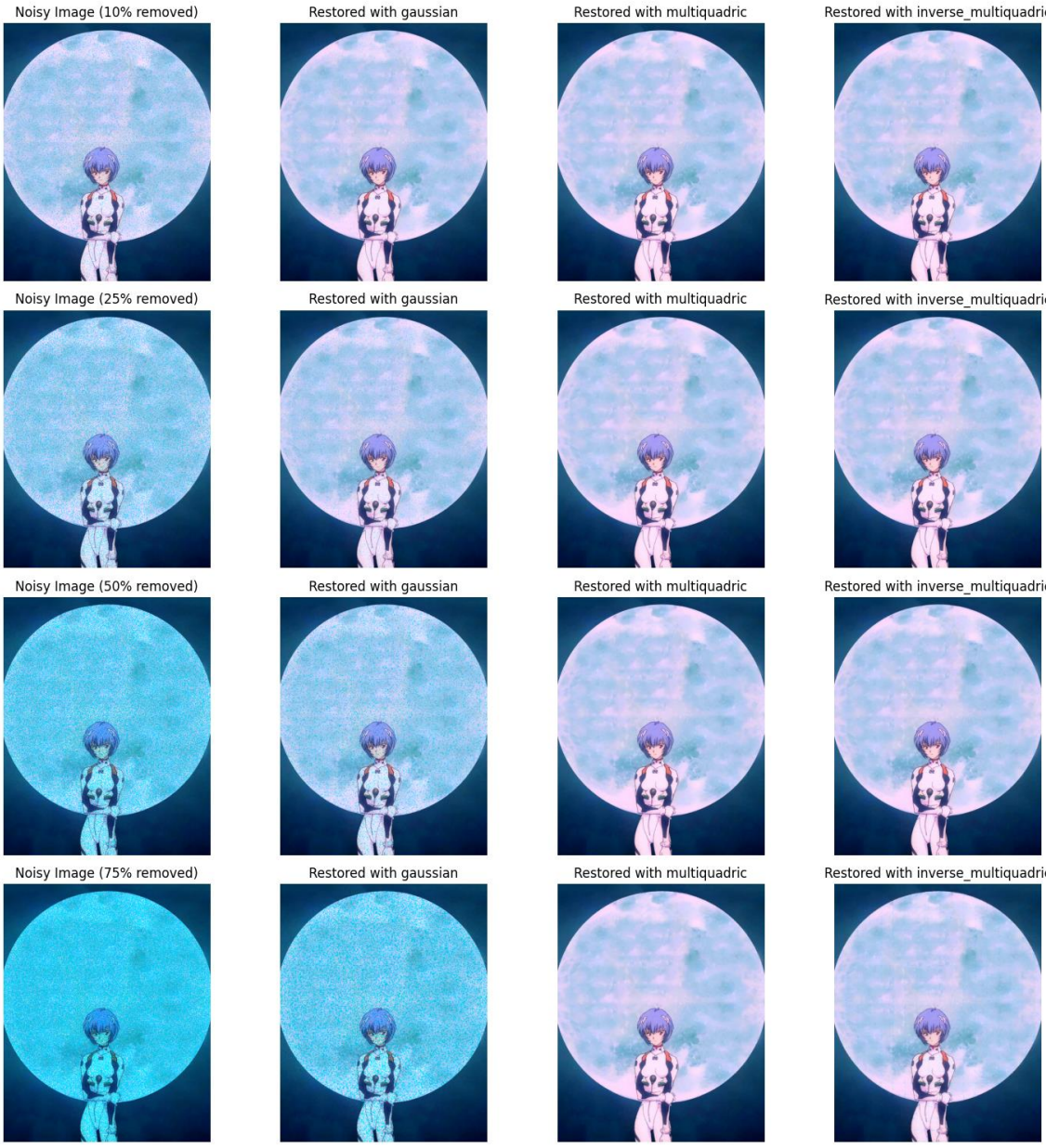
This is restored by Gaussian

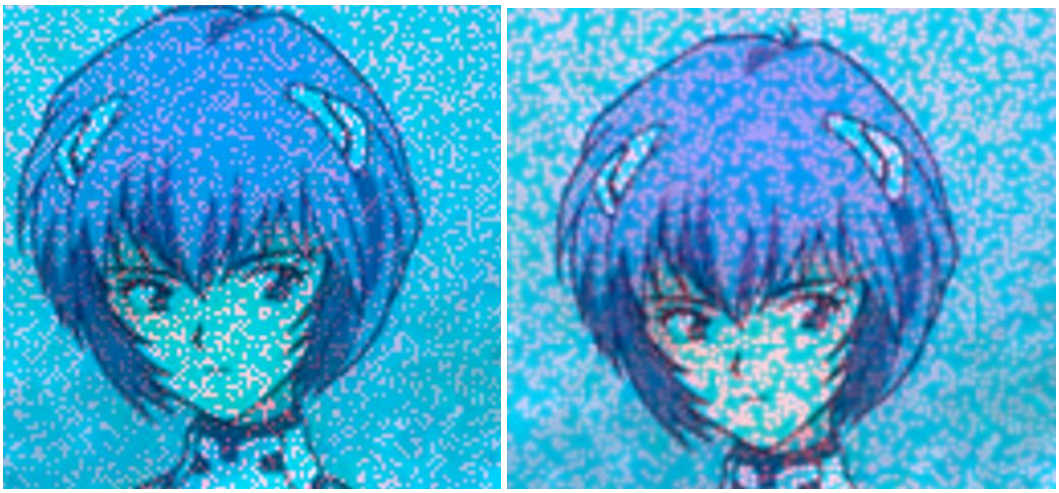
This is restored by inverse multiquadratic and next to it its multiquadratic

As you can see, by the results, both multiquadratic and inverse multiquadratic are good, which is not the case in Gaussian, especially when 75% of the pixels are removed.

Now lets move to another example:

This is the original image:

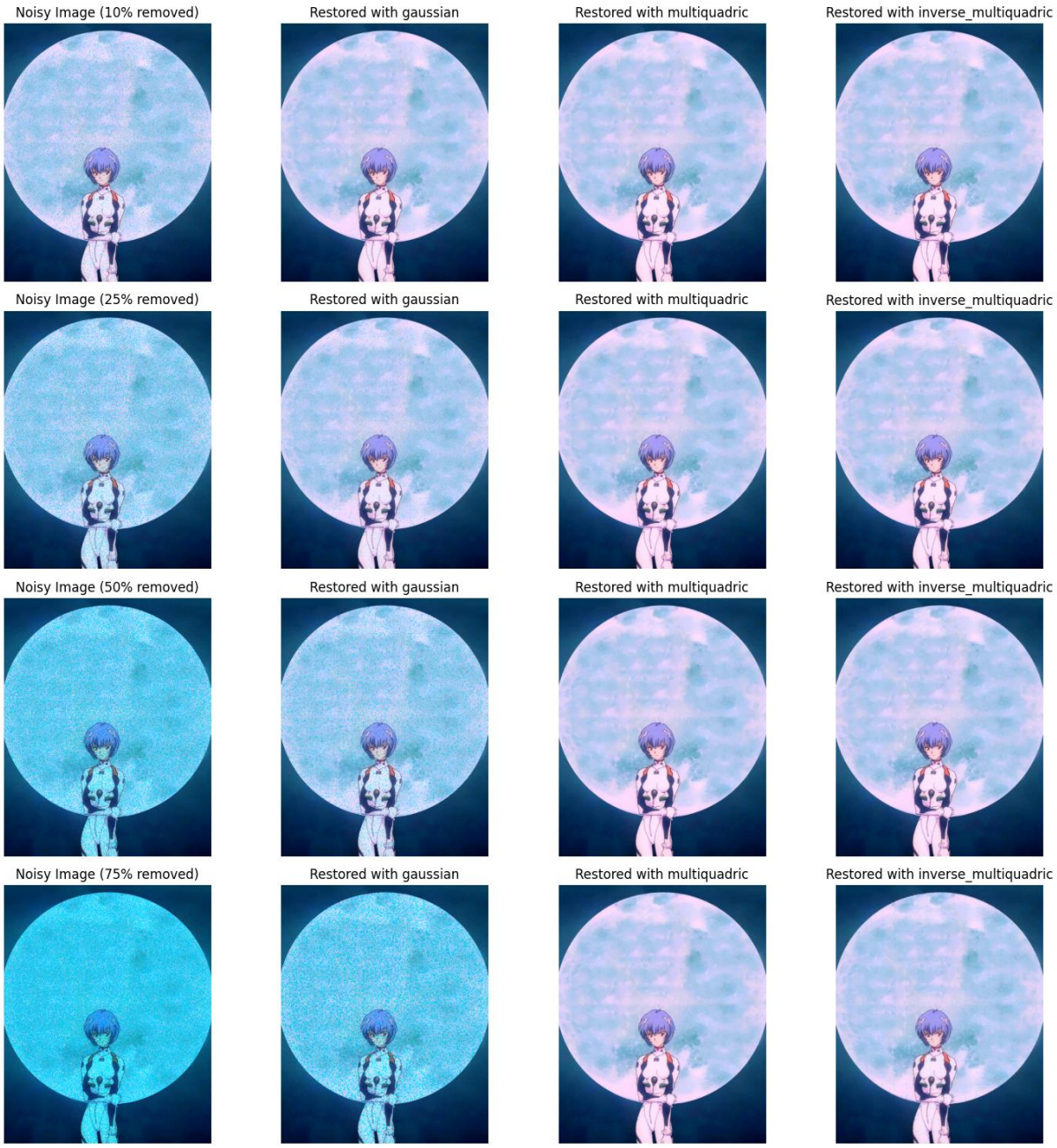

Noisy Image (10% removed)
Restored with gaussian
Restored with multiquadric
Restored with inverse_multiquadric
Noisy Image (25% removed)
Restored with gaussian
Restored with multiquadric
Restored with inverse_multiquadric
Noisy Image (50% removed)
Restored with gaussian
Restored with multiquadric
Restored with inverse_multiquadric
Noisy Image (75% removed)
Restored with gaussian
Restored with multiquadric
Restored with inverse_multiquadric



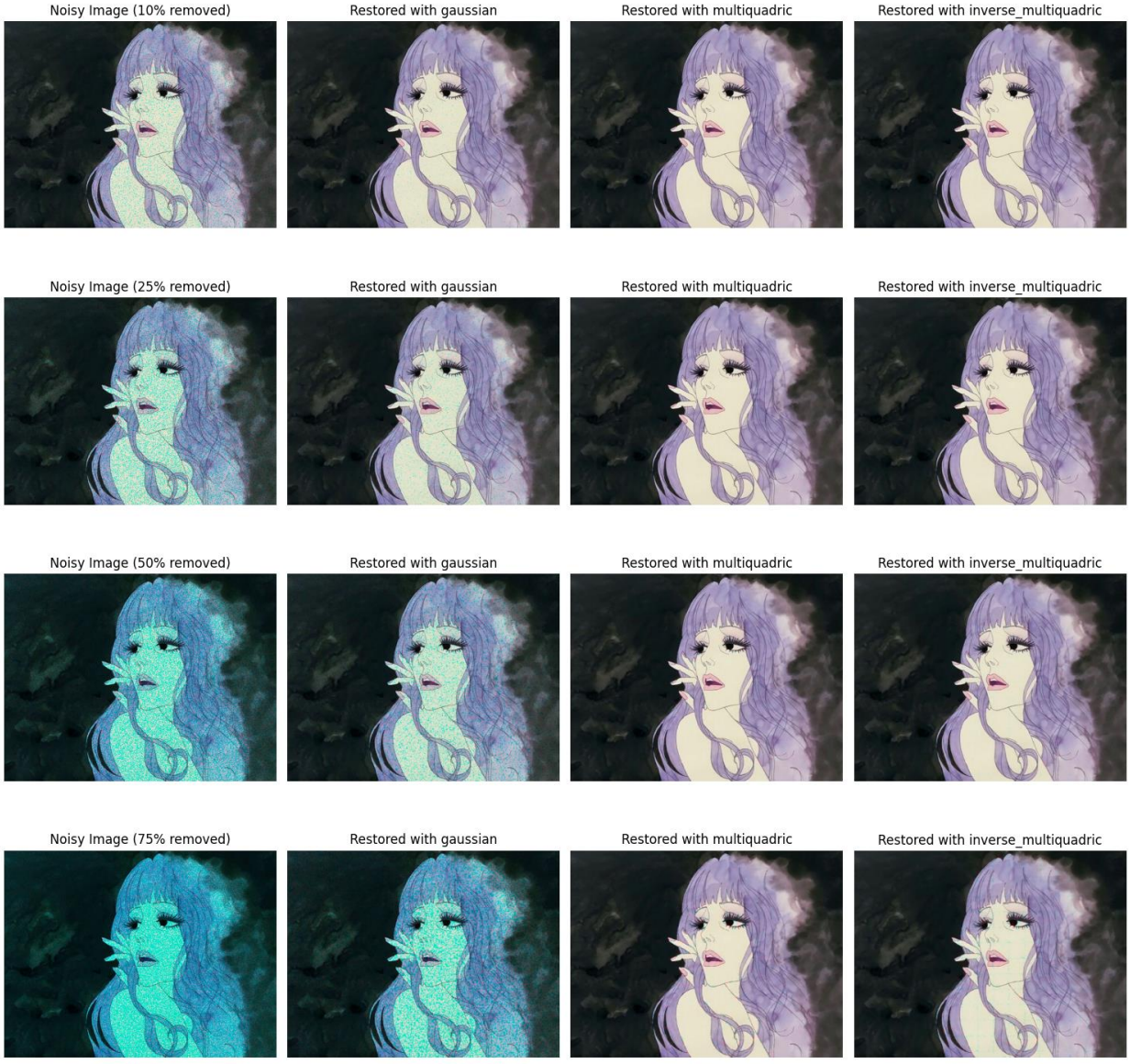
This is the result from 75% removed pixels by Gaussian

This is the result from again 75% removed but now with multiquadratic

I am sure, you can clearly see the difference



Now let us move to another example



Btw, I recommend everyone to watch this anime, because it is really well drawn.

Thank you for your consideration