Assignment 1

Part 1

1.1. Calculating the magnitude.

In section 1.1. we calculated magnitude by using imaginary and real values generated by the fft()

Code

```
SDoublePlane fft_magnitude(const SDoublePlane &fft_real, const SDoublePlane
&fft_imag)
{
        // TODO
        // Create an SDoublePlain for storing the magnitude values
        SDoublePlane magnitude(fft_real.rows(), fft_imag.cols());

        // Get start address pointers.
        double *magnitude_ptr = magnitude[0];
        double *real_part_data = fft_real[0];
        double *imag_part_data = fft_imag[0];

        int sz = fft_real.rows()*fft_imag.cols();

        // Calculating magnitude.
        for (int i = 0; i < sz; i++)
        {
                magnitude_ptr[i] = log(sqrt(real_part_data[i] *
real_part_data[i] + imag_part_data[i] * imag_part_data[i]));
        }

        return magnitude;
}
```

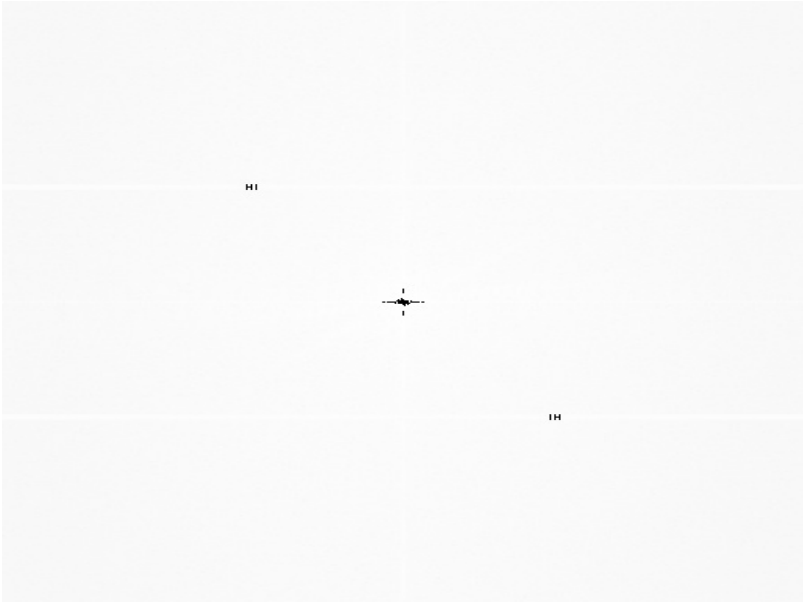


fig. Spectrum - With noise

fig. Actual Image with noise.

1.2. Fixing the noise in Image.

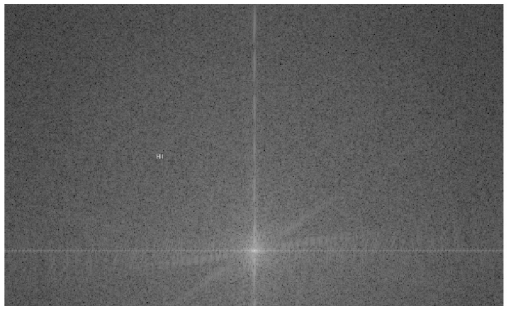
In section 1.2. we tried to fix the noise in the Image.

These are analysis we have performed on the image. Observed the following things.

Code

```
SDoublePlane remove_interference(const SDoublePlane &fft_real, const
SDoublePlane &fft_imag){

        //TODO
```

```
        SDoublePlane output_real(fft_real.rows(),fft_imag.cols());
        SDoublePlane magnitude(fft_real.rows(),fft_imag.cols());

        int j,k;

        // Removing from the Spectrum.
        for( j = 156 ; j < 161 ; j++){
                for(  k = 0 ; k < fft_real.cols() ; k++){
                        fft_real[j][k] = 0;
                    fft_imag[j][k] = 0;
                  }
                }

        // Removing from the Spectrum.
        for( j = 352 ; j < 357 ; j++){
                for(  k = 0 ; k < fft_real.cols() ; k++){
                        fft_real[j][k] = 0;
                    fft_imag[j][k] = 0;
                  }
                }

        double *magnitude_ptr = magnitude[0];
        double *real_part_data = fft_real[0];
        double *imag_part_data = fft_imag[0];

        int sz = fft_real.rows()*fft_imag.cols();

        // Calculating magnitude to generate spectrum of no noise image.
        for (int i = 0; i < sz; i++)
        {
                magnitude_ptr[i] = log(sqrt(real_part_data[i] *
real_part_data[i] + imag_part_data[i] * imag_part_data[i]));
        }

        // Writing Image output.
        SImageIO::write_png_file("updated_spectrum.png",magnitude,magnitude,ma
gnitude);
        ifft(fft_real,fft_imag, output_real);

        return output_real;
}
```



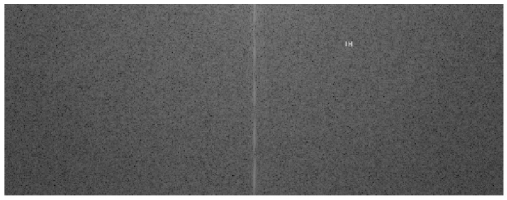


fig. Spectrum - With Noise [generated in matlab]

fig. Actual Image with noise.

fig. Actual Image with no noise.