Measuring Modulation Transfer Function

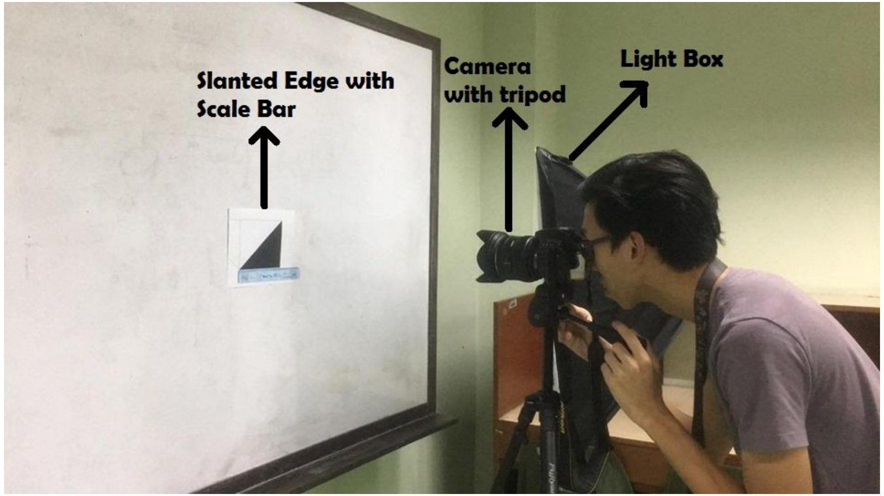
Domingo, Kenneth V.
Estrada, Viron Gil
Fernan, Charles Louie
Gaffud, Ymmanuel
Go, Mary Chris

To be able to measure the modulation transfer function of an imaging system

Goal

Methodology

- In this activity, we used the slanted edge technique to measure the MTFs of a camera with different lenses.
- A Nikon D3400 was positioned on a tripod and a slanted edge image was placed at a fixed distance away from it. A lightbox was used to illuminate the slanted edge image.
- An image of the slanted edge was captured for each of the lenses that the group has.
- The slanted edge images were then loaded and processed in *Python*.



Procedure Setup

Methodology:Programming Process

- Using *Python*, the intensity profiles of the line crossing the edge were determined.
- The derivative of the line, together with the Fourier transform of the derivative, was also determined.
- The positive half of the modulus of FFT was kept.
- The MTF was normalized and determined.
- This was done for all the lenses used.

1

```python
class MeasureMTF:
    def __init__(self, params_dict, plot=False, save=False):
        file = params_dict['file']
        if type(file) == str:
            img_raw = raw.imread(file)
            img = img_raw.postprocess(use_camera_wb=True,
                                      output_bps=8,
                                      no_auto_bright=True)
            self.image = cv.cvtColor(img, cv.COLOR_RGB2GRAY)
            self.filename = file.split('/')[-1].split('.')[0][4:]
        elif type(file) == np.ndarray:
            self.image = file
        else:
            raise NotImplementedError
        self.name = params_dict['name']
        self.rot = params_dict['rot']
        self.bbox = params_dict['bbox']
        self.len_px = params_dict['len_px']

        if plot:
            plt.imshow(self.image, 'gray')
            plt.grid(0)
            plt.tight_layout()
            if save and type(file) == str:
                plt.savefig(self.filename + '_gray.png', dpi=300, bbox_inches='tight')
            plt.show()

    def selectROI(self, show=False, save=False):
        rois = ROISelect(self.image)
        self.roi = rois.get_ROI()
        self.droi_dx = np.gradient(self.roi)
        self.droi_dx = np.hypot(self.droi_dx[0], self.droi_dx[1])

        if show:
            fig, ax = plt.subplots(1, 2, figsize=(16/2, 9/2))
            ax[0].imshow(self.roi, 'gray', vmin=self.roi.min(), vmax=self.roi.max())
            ax[0].grid(0)
            ax[0].set_title('ROI')
```

2

```python
            ax[1].imshow(self.droi_dx, 'gray')
            ax[1].grid(0)
            ax[1].set_title(r'$\dv{}{x}\mathrm{ROI}$')

            plt.tight_layout()
            if save:
                plt.savefig(self.filename + '_deriv.png', dpi=300, bbox_inches='tight')
            plt.show()

    def straighten(self, show=False, save=False):
        h,w = self.roi.shape
        M = cv.getRotationMatrix2D((w//2, h//2), self.rot, 1.0)
        roi_rot = cv.warpAffine(self.roi, M, (self.roi.shape))

        t, b, l, r = self.bbox
        self.roi_croprot = roi_rot[t:b, l:r]

        if show:
            fig, ax = plt.subplots(1, 2, figsize=(16/2, 9/2))
            ax[0].imshow(self.roi_croprot, 'gray', vmin=self.roi.min(), vmax=self.roi.max())
            ax[0].grid(0)
            ax[0].set_title('straightened ROI')
            ax[1].imshow(abs(fft.fftshift(fft.fft(self.roi_croprot))), 'hot')
            ax[1].grid(0)
            ax[1].set_title('ROI FT')

            plt.tight_layout()
            if save:
                plt.savefig(self.filename + '_roi.png', dpi=300, bbox_inches='tight')
            plt.show()

    def project(self, show=False, save=False):
        roi_hproj = self.roi_croprot.mean(axis=0)
        roi_hproj /= roi_hproj.max()
        roi_lsf = np.gradient(roi_hproj)
        roi_lsf /= roi_lsf.max()
        self.roi_lsf = roi_lsf
```

3

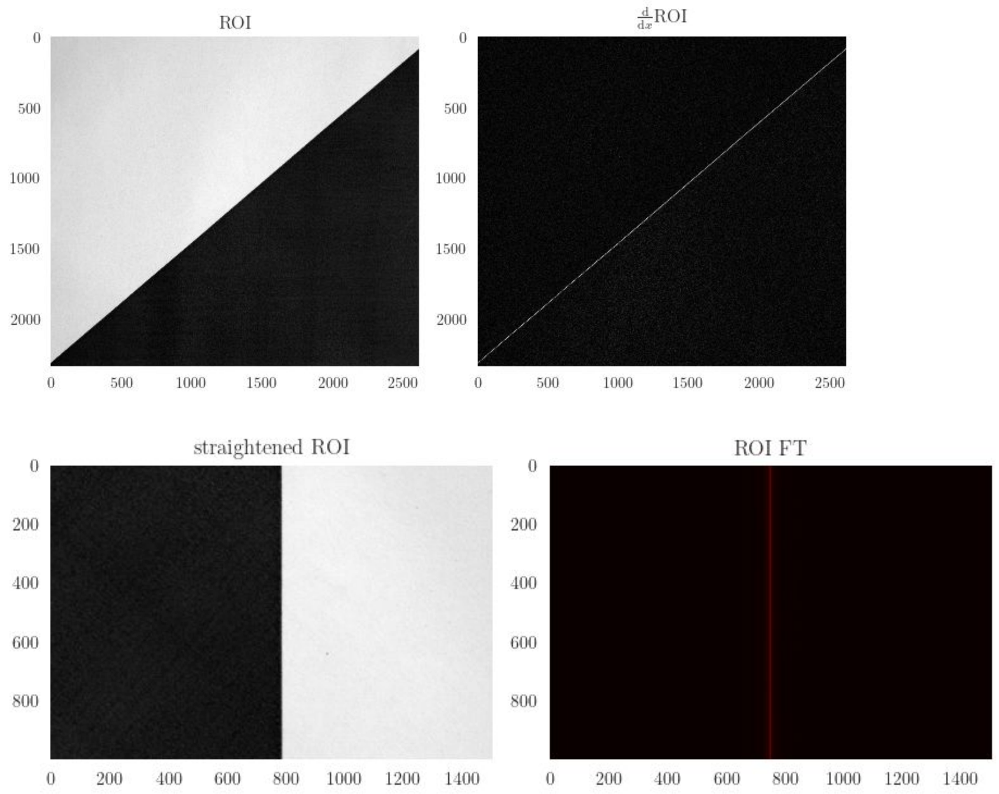
```python
        if show:
            plt.plot(roi_hproj, lw=1)
            plt.plot(roi_lsf, lw=1)
            plt.legend(['ESF', 'LSF'])
            plt.tight_layout()
            if save:
                plt.savefig(self.filename + '_sf.png', dpi=300, bbox_inches='tight')
            plt.show()

    def calcMTF(self, len_mm=133, show=False, save=False):
        self.px2mm = lambda px: px * len_mm/self.len_px
        N = len(self.roi_lsf)
        self.mtf = abs(fft.fft(self.roi_lsf))[:N//2]
        self.mtf /= self.mtf.max()
        cpx = np.linspace(0, 1, self.mtf.size)
        mtf50 = cpx[np.argmin(abs(0.5 - self.mtf))]
        lpm_factor = mtf50/(23.5/6000)
        lpm = cpx * lpm_factor
        self.mtf50 = lpm[np.argmin(abs(0.5 - self.mtf))]

        if show:
            plt.plot(lpm, self.mtf, label=r'$\mathrm{MTF50 = %i}$ lp/mm' %(np.round(self.mtf50)))
            plt.xlabel('lp/mm')
            plt.ylabel('MTF')
            plt.title(self.name)
            plt.legend()
            plt.tight_layout()
            if save:
                plt.savefig(self.filename + '_mtf.png', dpi=300, bbox_inches='tight')
            plt.show()

    def main(self, len_mm=113, show=False, save=False):
        self.selectROI(show, save)
        self.straighten(show, save)
        self.project(show, save)
        self.calcMTF(len_mm, show, save)
```

```python
prep_dict = dict({ 'file': ['raw/_KVD7149.NEF', 'raw/_KVD7150.NEF', 'raw/_KVD7153.NEF', 'raw/_KVD7155.NE
F'],
                   'name': ['Nikon D3400 + SIGMA 24-70mm@70, f/2.8', 'Nikon D3400 + NIKKOR 70-300@70, f/4.
5',
                            'Nikon D3400 + NIKKOR 50mm, f/2.8', 'Nikon D3400 + NIKKOR 18-55mm@35, f/4.5'],
                   'rot': 49.5+180,
                   'bbox': (1000, 2000, 500, 2000),
                   'len_px': [4324-1541, 4455 - 1533, 4120 - 1244, 4345 - 1573]
                  })
```

4

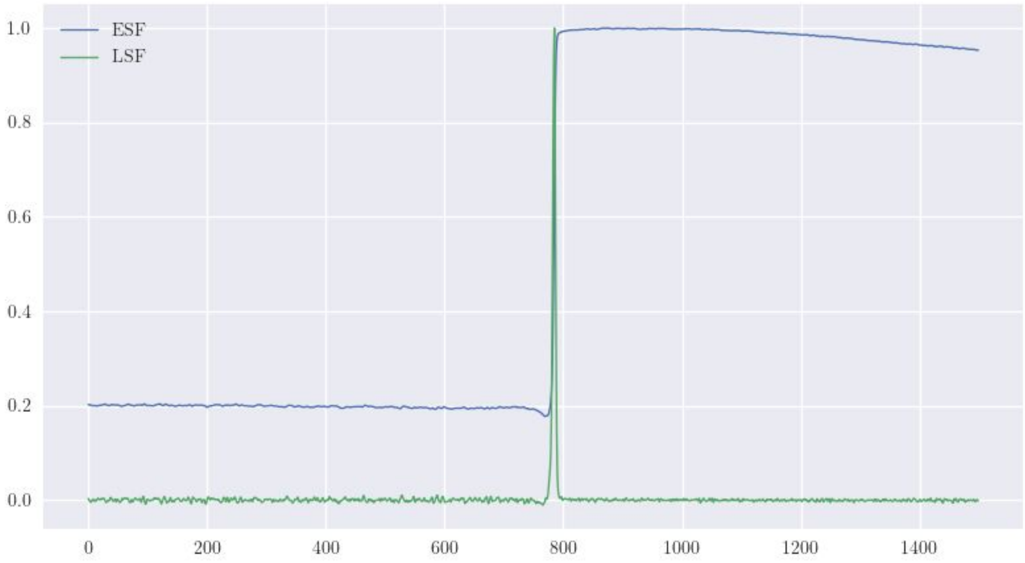
```python
for i in range(len(prep_dict['file'])):
    params_dict = dict({'file': prep_dict['file'][i],
                        'name': prep_dict['name'][i],
                        'rot' : prep_dict['rot'],
                        'bbox': prep_dict['bbox'],
                        'len_px': prep_dict['len_px'][i]
                       })
    cal = MeasureMTF(params_dict)
    cal.main(show=True, save=True)
```

Results:



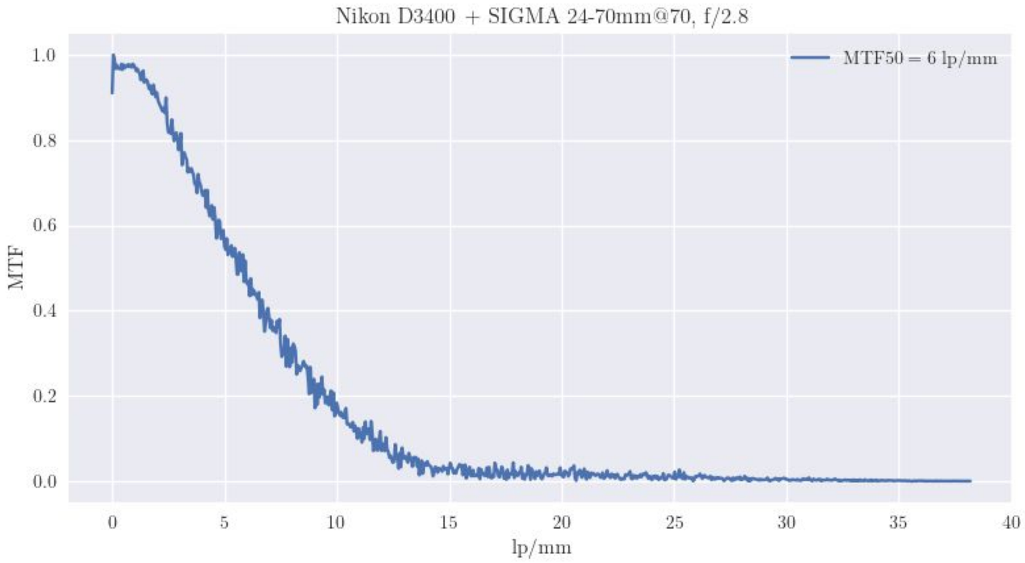
Region of interest (ROI) of the slanted edge. The Fourier Transform of the derivative of the line edge.

Results:



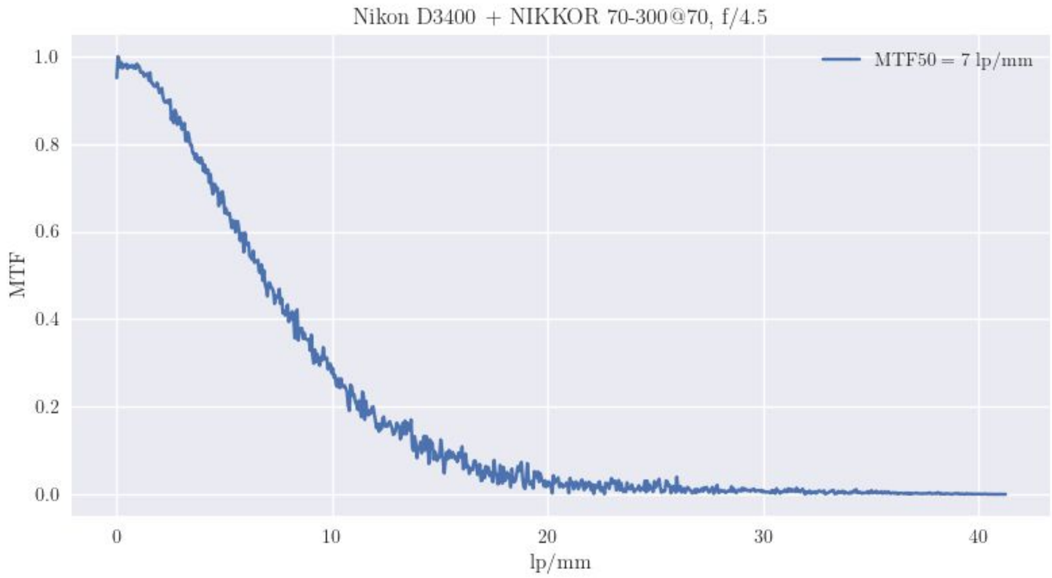
Evaluating the Edge spread function (ESF) and line spread function (LSF) of Nikon D3400 + SIGMA 24-70mm @70, f/2.8

Results: MTF of Nikon D3400 + SIGMA 24-70mm @ 70



The calculated MTF for Nikon D3400 + SIGMA 24-70mm @ 70 is **6 lp/mm**

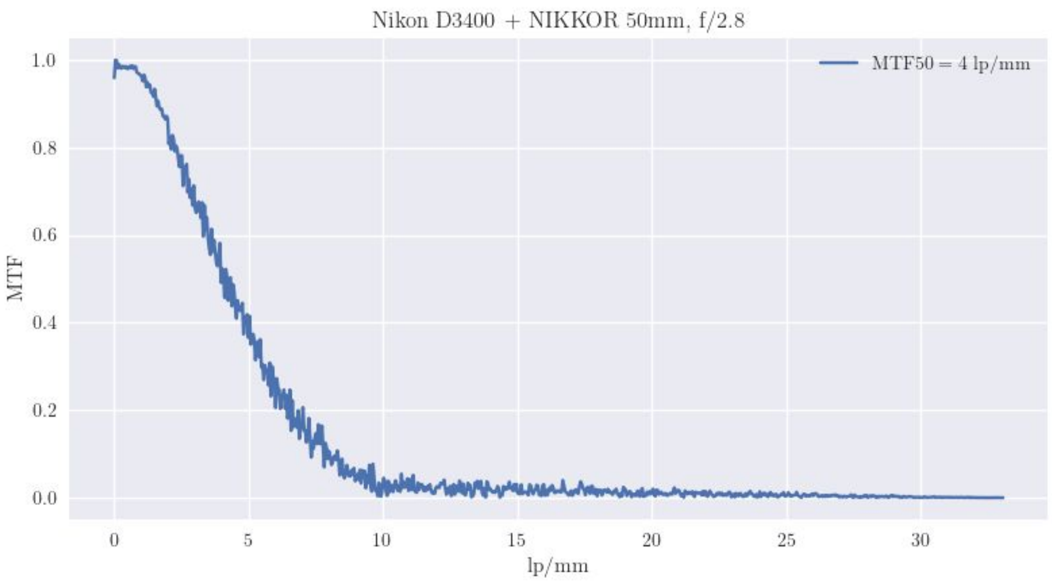
Results: MTF of Nikon D3400 + NIKKOR 70-300@70, f/4.5



Same procedure is done but for different specification.

The calculated MTF for Nikon D3400 + NIKKOR 70-300@70, f/4 is **7 lp/mm**

Results: MTF of Nikon D3400 + NIKKOR 50mm, f/2.8



Same procedure is done but for different specification.

The calculated MTF for Nikon D3400 + NIKKOR 50mm, f/2.8 is **4 lp/mm**

Discussion

Nikon D3400 + SIGMA 24-70mm @ 70 f2/8 - **6 lp/mm**

Nikon D3400 + NIKKOR 70-300@70, f/4.5 - **7 lp/mm**

Nikon D3400 + NIKKOR 50mm, f/2.8 - **4lp/mm**

The Nikon D3400 with NIkkor lens set at 70mm and aperture of f/4.5 obtained the highest MTF.

Summary

The modulation transfer function of various lenses were obtained through capturing a slanted edge and using python. A camera of Nikon D3400 was used. Lenses such as Sigma 24-70mm, Nikkor 70-300mm, and Nikkor 50mm, were tested.

The intensity profile of the line crossing the edge were determined using python. Through this, the MTF was determined. This process was done to all lenses. Our group found out that Nikkor lens set at 70mm obtained the highest MTF. This made sense because this lens is known to have a better quality in showing contrast in the original object.

In a nutshell, we were able to achieve the goal of finding out the MTF of lenses.

References

1. https://www.edmundoptics.com/resources/application-notes/optics/introduction-to-modulation-transfer-function/
2. https://www.optikos.com/modulation-transfer-function/

Contribution

Domingo - actual experiment, coding

Estrada - actual experiment, results and discussion

Fernan - actual experiment, results

Gaffud - actual experiment, methodology

Go - actual experiment, summary