slit_conrner_detection

June 19, 2019

1 Overview

This notebook is used for the development of the auto slit conrner detection algorithm, which is critical for the automated detector drift correction.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import medfilt
from scipy.signal import medfilt2d
from scipy.signal import find_peaks
from skimage import exposure
from skimage.transform import match_histograms

from scipy.ndimage import gaussian_filter1d
from tomoproc.util.npmath import discrete_cdf
```

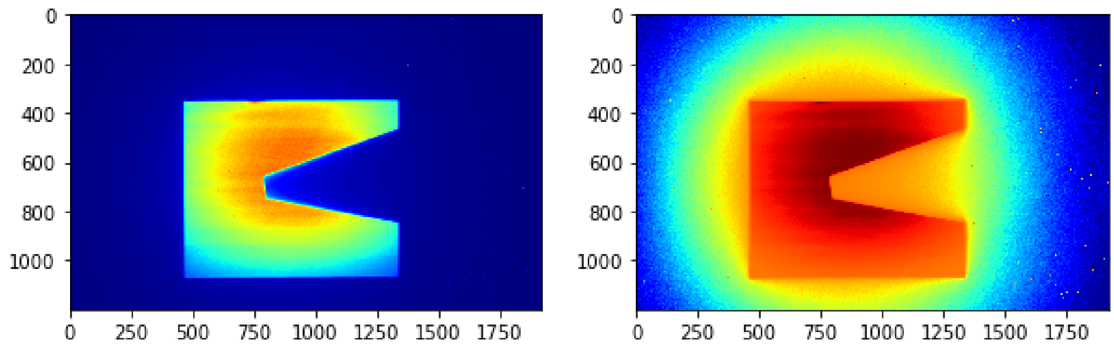
[2]:
```
val_atPercent = lambda ar, p: np.sort(ar.flatten())[int(np.prod(ar.shape)*p)]
wgt_histequal = lambda ar: (np.sort(ar.flatten()).searchsorted(ar) + 1)/np.
 ↪prod(ar.shape)
wgts_binned   = lambda wgts, bins: np.int64(np.floor(wgts * bins)).astype(wgts.
 ↪dtype) / bins
```

2 Initial test with the first image

let's import one images and start with the code we submitted to tomopy a long time ago.

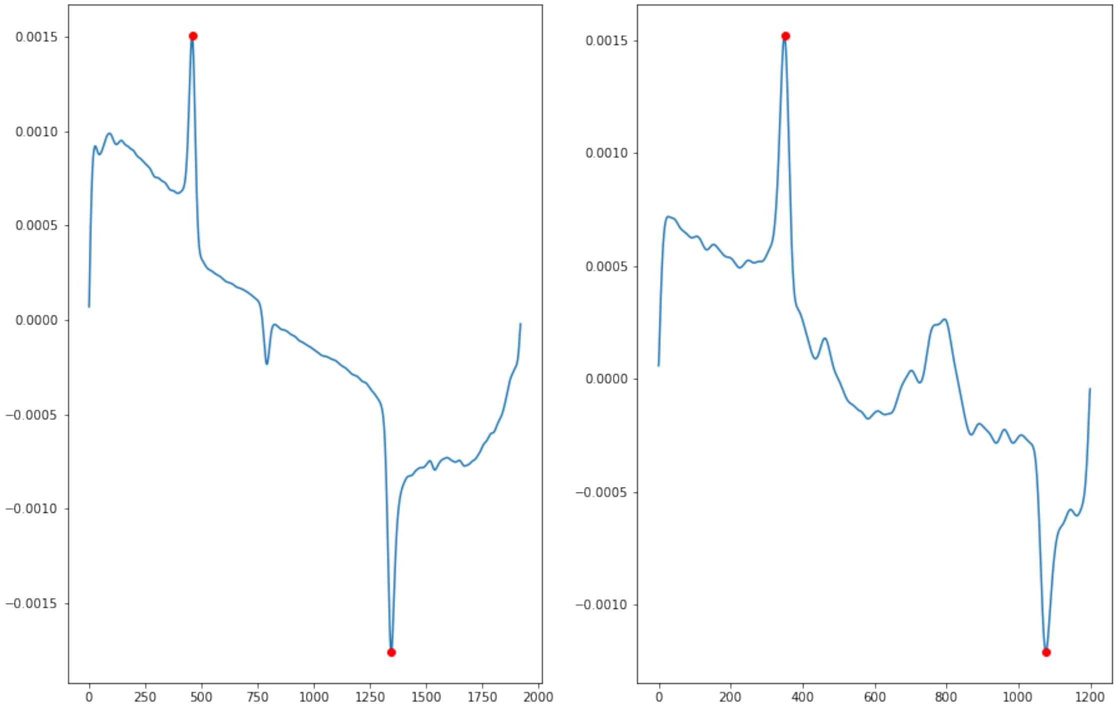
[3]:
```
rawimg = plt.imread('data/test_midregion_1.tif')

fig, axes = plt.subplots(1,2,figsize=(10, 5))

img = exposure.equalize_hist(rawimg)

axes[0].imshow(rawimg, 'jet')
axes[1].imshow(img, 'jet')
```

[3]:
```
<matplotlib.image.AxesImage at 0x1c19624748>
```



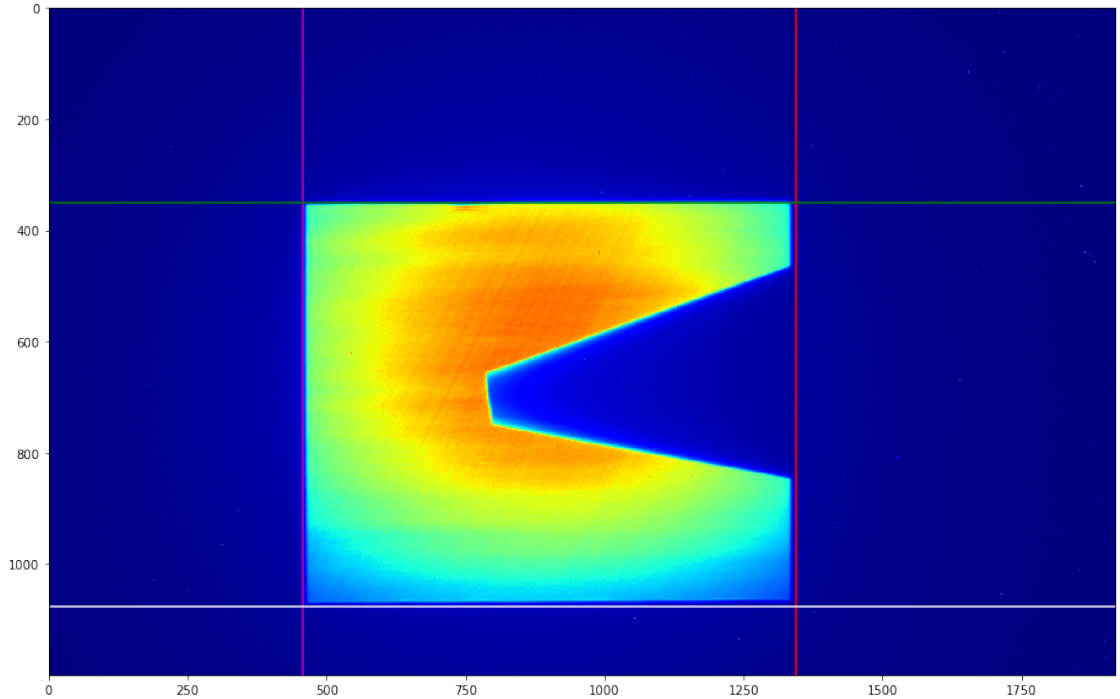
[4]:
```
tmp = np.log(medfilt2d(img.astype(float))+1)

col_prof = gaussian_filter1d(np.average(tmp, axis=0), sigma=11)
dot_col_prof = np.gradient(col_prof)

left = np.argmax(dot_col_prof)
right = np.argmin(dot_col_prof)

row_prof = gaussian_filter1d(np.average(tmp, axis=1), sigma=11)
dot_row_prof = np.gradient(row_prof)

top = np.argmax(dot_row_prof)
bot = np.argmin(dot_row_prof)

fig, axes = plt.subplots(1, 2, figsize=(15, 10))

axes[0].plot(dot_col_prof)
axes[0].plot(left, dot_col_prof[left], 'ro')
axes[0].plot(right, dot_col_prof[right], 'ro')

axes[1].plot(dot_row_prof)
axes[1].plot(top, dot_row_prof[top], 'ro')
axes[1].plot(bot, dot_row_prof[bot], 'ro')
```

[4]:
```
[<matplotlib.lines.Line2D at 0x1c1a03af28>]
```



[5]:
```
fig, ax = plt.subplots(1,1,figsize=(18, 10))

ax.imshow(rawimg, 'jet')

ax.axvline(left, color='m')
ax.axvline(right, color='r')

ax.axhline(top, color='g')
ax.axhline(bot, color='w')
```

[5]:
```
<matplotlib.lines.Line2D at 0x1c1a68af60>
```

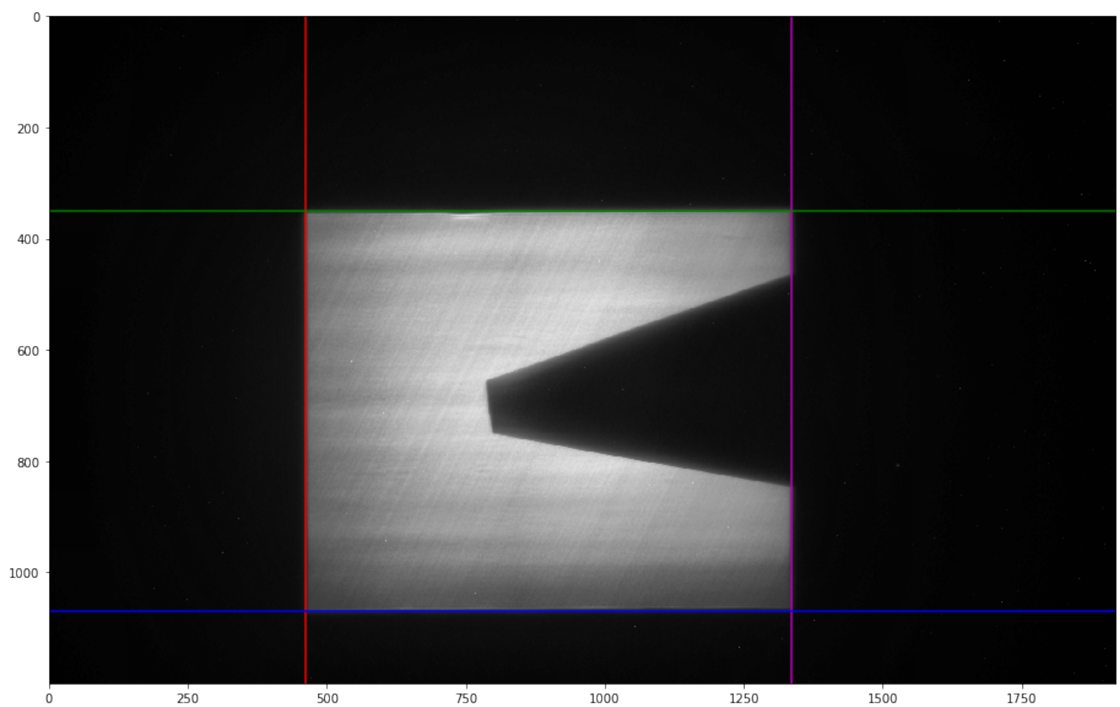


Making it into a function

[7]:
```
def guess_slit_box(img):

    # Contrast stretching
    pl, ph = np.percentile(img, (2, 98))
    img = exposure.rescale_intensity(img, in_range=(pl, ph))

    # equilize hist
    img = exposure.equalize_adapthist(img)

    # map to log to reveal transition box
    img = np.log(medfilt2d(img.astype(float))+1)

    # get row and col profile gradient
    pdot_col = np.gradient(gaussian_filter1d(np.average(img, axis=0), sigma=11))
    pdot_row = np.gradient(gaussian_filter1d(np.average(img, axis=1), sigma=11))

    return {
        'left':  np.argmax(pdot_col),
        'right': np.argmin(pdot_col),
        'top':   np.argmax(pdot_row),
        'bot':   np.argmin(pdot_row),
    }
```

```python
# ---------------
# testing
rawimg = plt.imread('data/test_midregion_1.tif')

edges = guess_slit_box(rawimg)

fig, ax = plt.subplots(1,1,figsize=(18, 10))

# ax.imshow(rawimg, 'gray')
# ax.imshow(exposure.equalize_hist(rawimg), 'gray')
ax.imshow(exposure.equalize_adapthist(rawimg), 'gray')

ax.axvline(edges['left'],  color='r')
ax.axvline(edges['right'], color='m')
ax.axhline(edges['top'],   color='g')
ax.axhline(edges['bot'],   color='b')

print(edges)
```

```
{'left': 461, 'right': 1335, 'top': 350, 'bot': 1069}
```

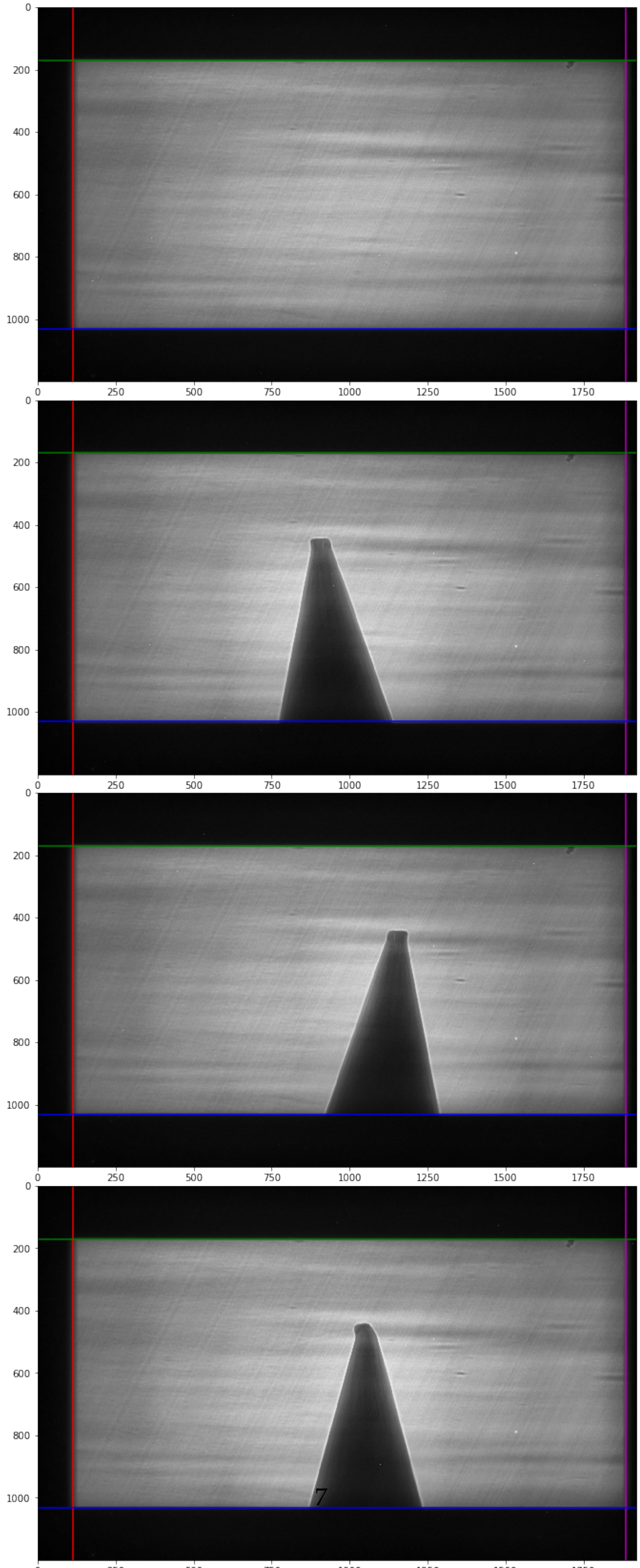


3 Systematic test under different conditions

3.1 Standard case

In most cases, the majority of detector should be used for imaging.

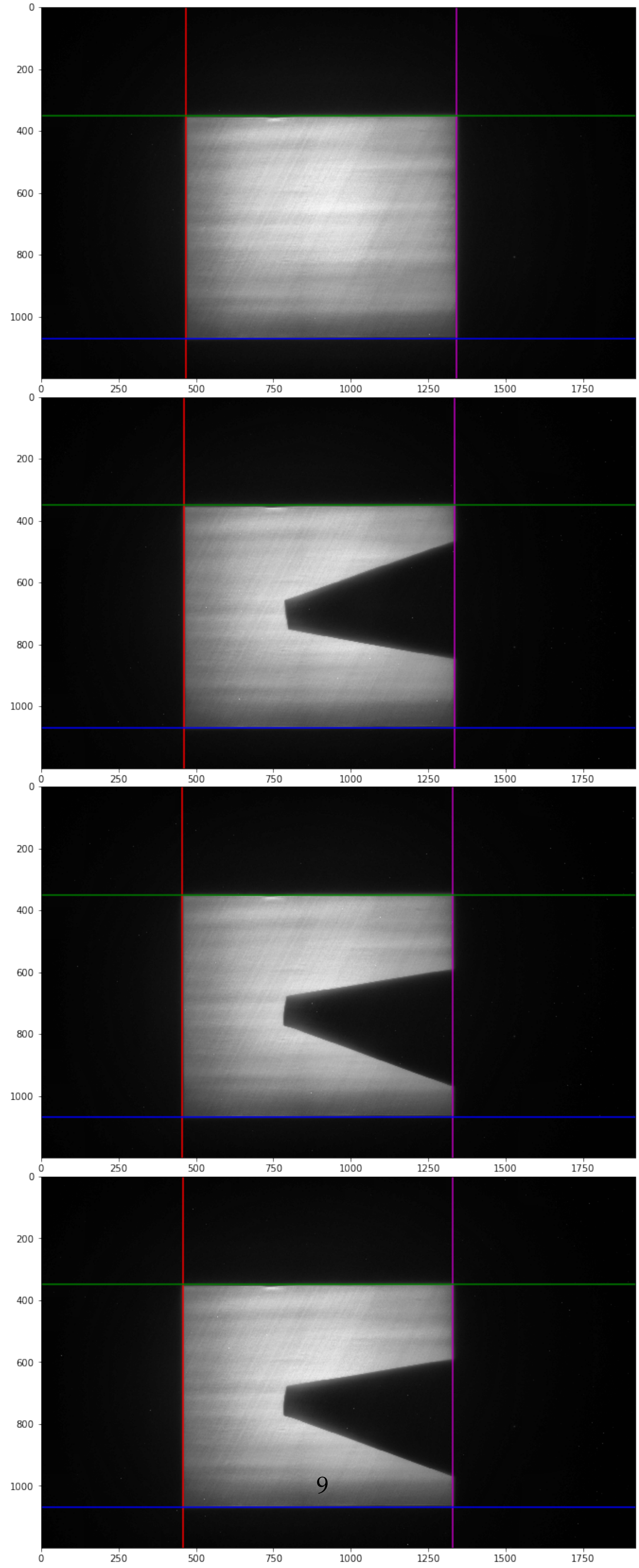
[16]:
```
# we have 0-3, 4 regular cases here

fig, axs = plt.subplots(4, 1, figsize=(15, 30))

for i in range(4):
    img = plt.imread(f'data/test_bigregion_{i}.tif')
    edges = guess_slit_box(img)
    axs[i].imshow(exposure.equalize_adapthist(img), 'gray')
    axs[i].axvline(edges['left'],  color='r')
    axs[i].axvline(edges['right'], color='m')
    axs[i].axhline(edges['top'],   color='g')
    axs[i].axhline(edges['bot'],   color='b')

plt.subplots_adjust(wspace=0, hspace=0.05)
```


0
200
400
600
800
1000
0 250 500 750 1000 1250 1500 1750
0
200
400
600
800
1000
0 250 500 750 1000 1250 1500 1750
0
200
400
600
800
1000
0 250 500 750 1000 1250 1500 1750
0
200
400
600
800
1000
0 250 500 750 1000 1250 1500 1750

3.2 Mid region case

In some cases, a good region of the detector is purposely selected for imaging.

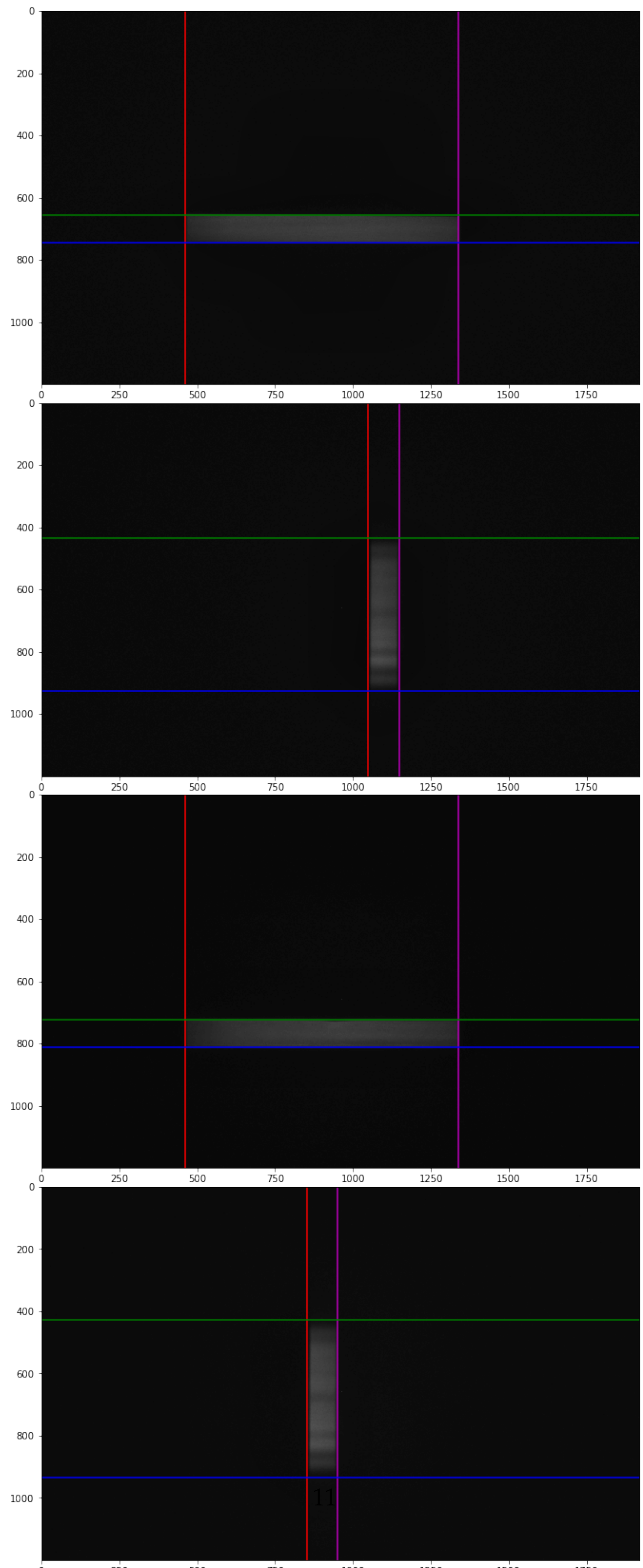
[17]:
```python
# we have 0-3, 4 mid region cases here

fig, axs = plt.subplots(4, 1, figsize=(15, 30))

for i in range(4):
    img = plt.imread(f'data/test_midregion_{i}.tif')
    edges = guess_slit_box(img)
    axs[i].imshow(exposure.equalize_adapthist(img), 'gray')
    axs[i].axvline(edges['left'],  color='r')
    axs[i].axvline(edges['right'], color='m')
    axs[i].axhline(edges['top'],   color='g')
    axs[i].axhline(edges['bot'],   color='b')

plt.subplots_adjust(wspace=0, hspace=0.05)
```


0
200
400
600
800
1000
0
250
500
750
1000
1250
1500
1750

3.3 Small region cases

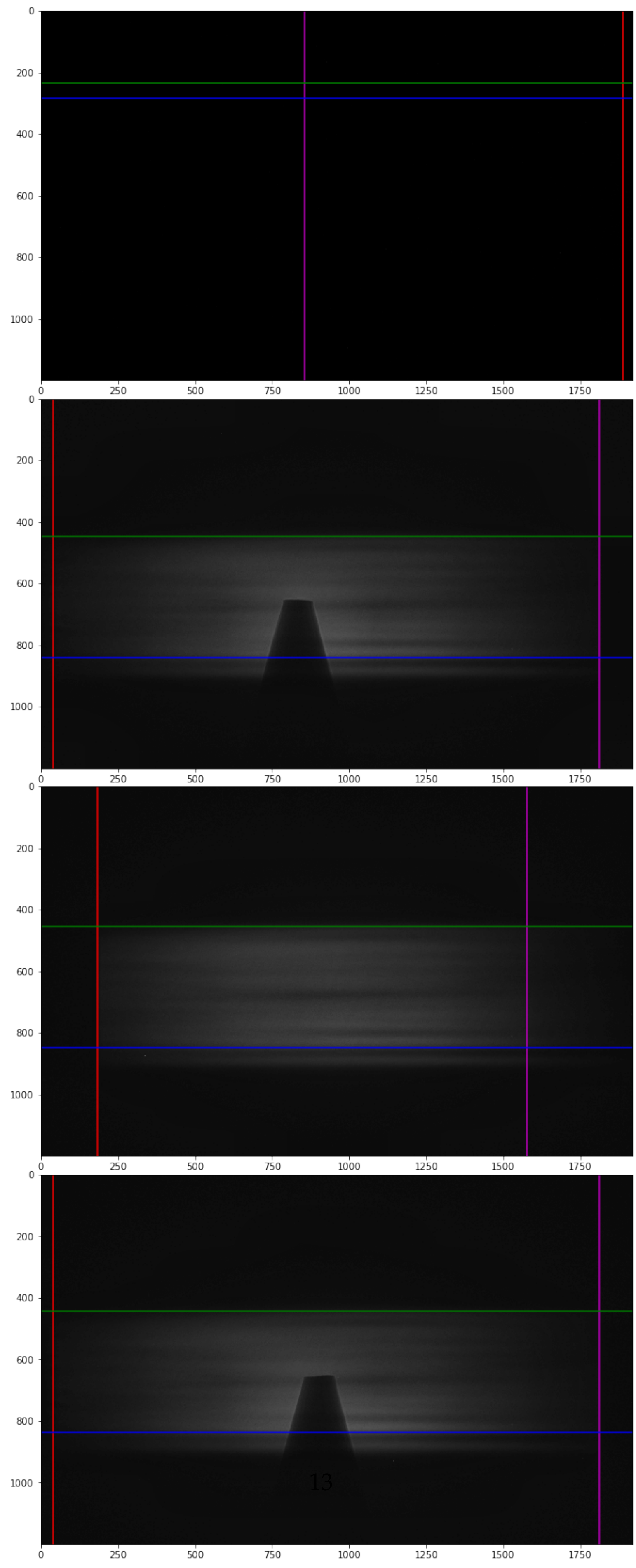
In very rare situation, the imaging region is shrinked to a really small region on the detector.

[18]:
```
# we have 0-3, 4 small region cases here

fig, axs = plt.subplots(4, 1, figsize=(15, 30))

for i in range(4):
    img = plt.imread(f'data/test_smallregion_{i}.tif')
    edges = guess_slit_box(img)
    axs[i].imshow(exposure.equalize_adapthist(img), 'gray')
    axs[i].axvline(edges['left'],  color='r')
    axs[i].axvline(edges['right'], color='m')
    axs[i].axhline(edges['top'],   color='g')
    axs[i].axhline(edges['bot'],   color='b')

plt.subplots_adjust(wspace=0, hspace=0.05)
```


0 200 400 600 800 1000
0 250 500 750 1000 1250 1500 1750
0 200 400 600 800 1000
0 250 500 750 1000 1250 1500 1750
0 200 400 600 800 1000
0 250 500 750 1000 1250 1500 1750
0 200 400 600 800 1000
0 250 500 750 1000 1250 1500 1750

Basically, as long as all four blade slits are used, the function should be able to return the approximated location of the four blades without any trouble. However, if there is no slits at all, random results will be returned.

[19]:
```
# we have 0-3, 4 no slit cases here

fig, axs = plt.subplots(4, 1, figsize=(15, 30))

for i in range(4):
    img = plt.imread(f'data/test_noslit_{i}.tif')
    edges = guess_slit_box(img)
    axs[i].imshow(exposure.equalize_adapthist(img), 'gray')
    axs[i].axvline(edges['left'],  color='r')
    axs[i].axvline(edges['right'], color='m')
    axs[i].axhline(edges['top'],   color='g')
    axs[i].axhline(edges['bot'],   color='b')

plt.subplots_adjust(wspace=0, hspace=0.05)
```



The key point here is that we should only use this slit box finder if we know that slits are used in the experiment, not the other way around.

```
[ ]:
```