1. **Preliminary (preliminary.py)**

Experiments
Side-by-side comparisons of original image with grayscale image

houndog1.png

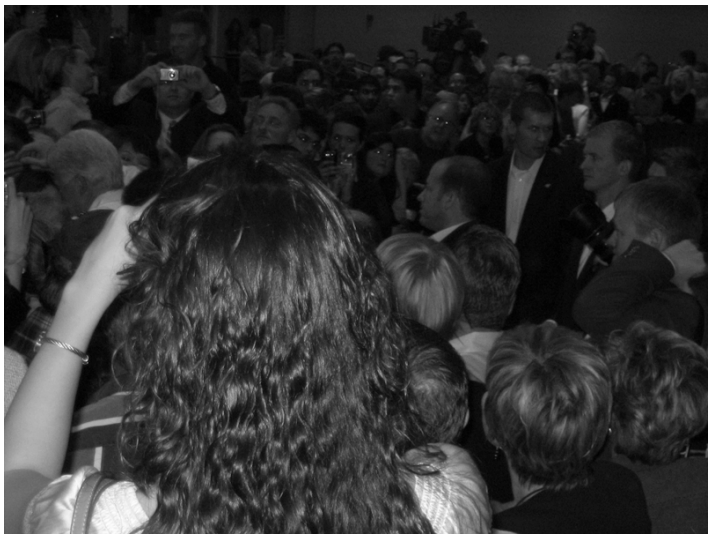
ezito.jpg

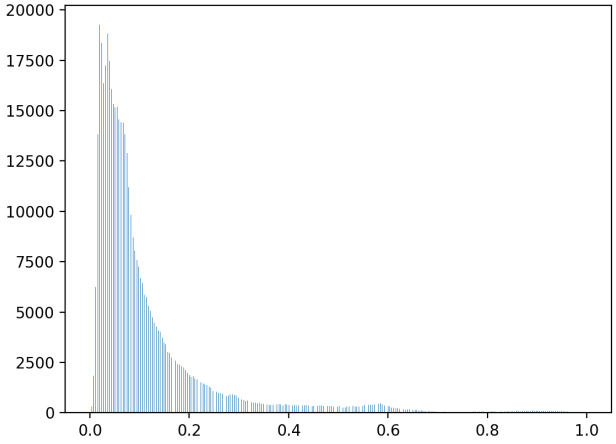
2. Build a histogram (build_histogram.py)

A histogram can be built by bucketing the intensity values of the image into bins. The number of bins can be specified by the user to visualize the spread of the intensities. It's then possible to threshold pixels based of their position in the histogram which can be used to find specific parts of the image.

Experiments

crowd.tif





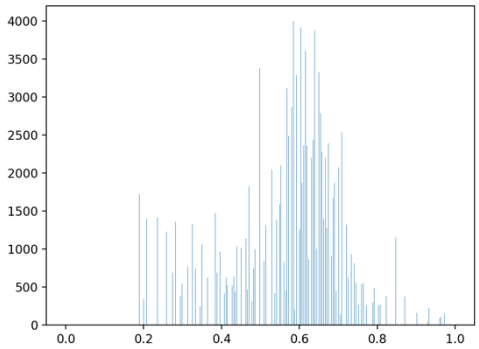
The histogram shows that a large section of the image has very low intensities compared to the rest of the image.

For # of buckets = 1000

what regions/objects are what part of the histogram?

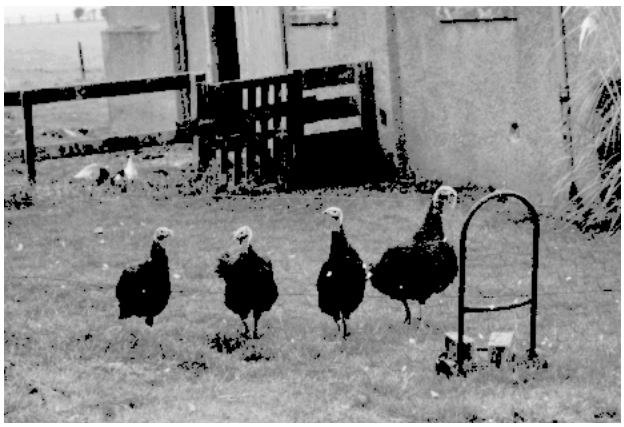
Below a threshold of 0.1, the areas that are not caught in the cameras flash show up. Above this threshold the remaining crowd that's caught in the cameras flash can be seen.

turkeys.tif



The histogram shows that most of the image falls in the middle with gray pixels. There is a cluster of gray values as most of the picture is gray.
For # of buckets = 1000

what regions/objects are what part of the histogram?
The darker bodies of the turkeys show up as well as the shadowed part of the fence below the 0.45 threshold of the histogram. The grass, shed, fence show up above this threshold.



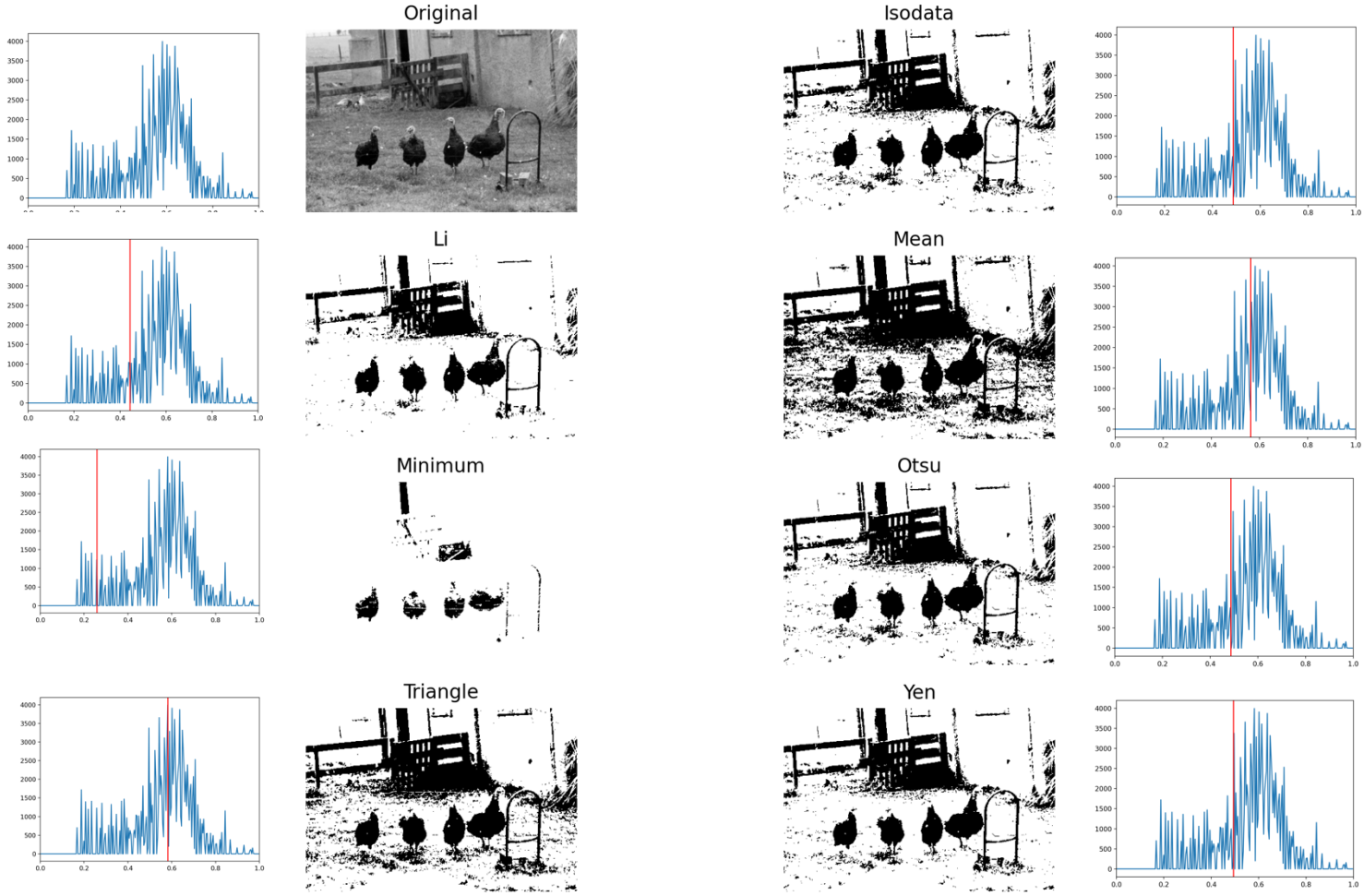
3. **Connected components and region (components.py)**
   The image can be thresholded using the histogram of the image. The threshold can be determined using different algorithms. From the thresholded image, it is possible to determine the connected components of the image using thresholded image.

   Performing flood fill on the thresholded image generated can give the connected components of the image.

   Parameters

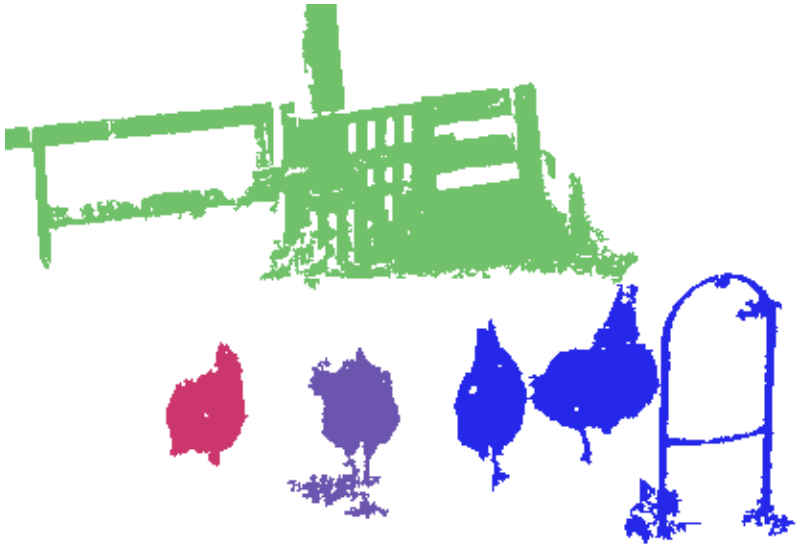
1. # of bins: The number of bins used to generate the histogram
2. Connected component size: The number of pixels that the connected component should be
3. Adjacent pixel distance: skimage can limit the neighboorhood of a pixel when using flood_fill

Experiments: turkeys.tiff



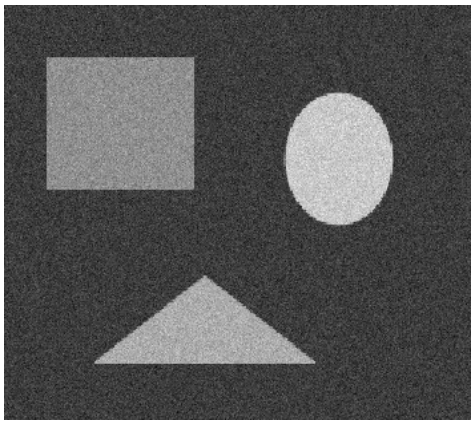
The threshold provided by Otsu's algorithm seems to give the best results in terms of categorizing different regions in the image. Using the Otsu thresholded image to perform flood fill with the below params

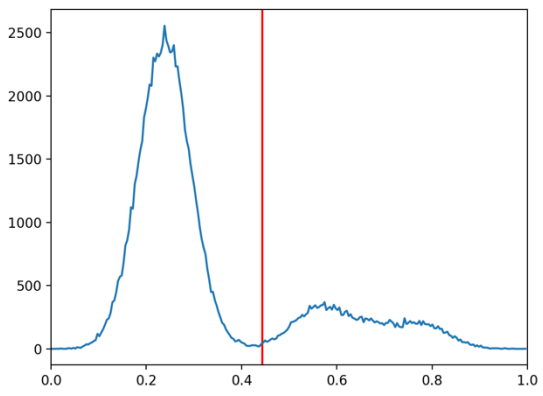
# of bins: 256
Connected component size: 1000 pixels
Adjacent pixel distance: 1



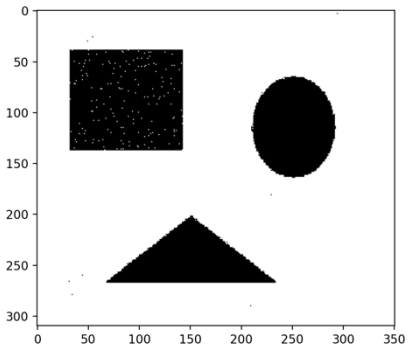
shapes_noise.png
Params:

Threholding: Otsus algorithm
# of bins: 256
Connected component size: 1000 pixels
Adjacent pixel distance: 1

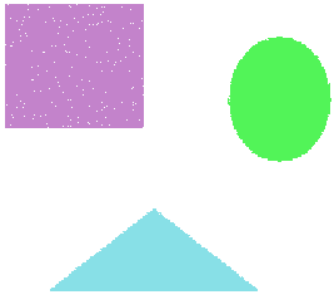




Thresholded image (high)



Connected components



coins.png

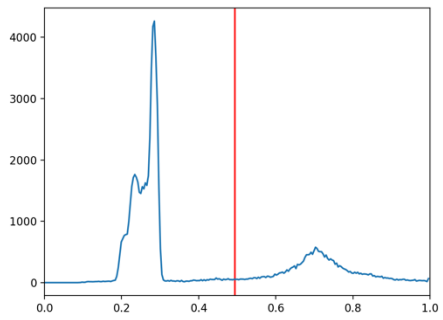
Params:

Threholding: Otsus algorithm

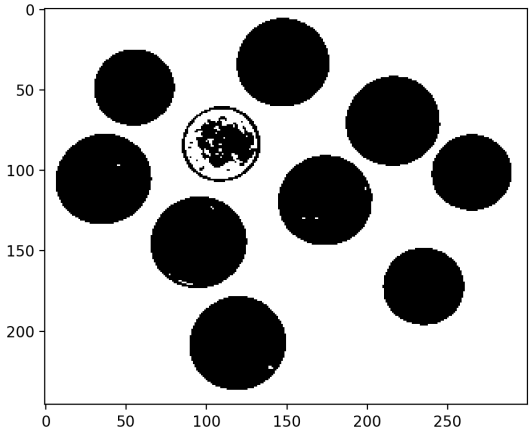
# of bins: 256

Connected component size: 100 pixels

Adjacent pixel distance: 1



Thresholded image (high)



Connected components

Note: Due to the gray in one of the coins the pixels within the coin are thresholded up to 1 using Otsu's algorithm. This affects flood fill and the coin doesn't completely show up as a connected component.

4. **Histogram equalization (histogram_equlization.py)**
   Using the in-built libraries in skimage, it's possible to run histogram equalization algorithms to help with improving contrast.

Important parameters for histogram equalization

1. Number of bins: The number of bins to bucket the intensities for constructing the cumulative distribution function.
2. Mask: A mask in the shape of the image can be passed to the equalize histogram function wherein only pixels that are true will be equalized.

Important parameters for adaptive histogram equalization

1. Kernel_size: Size of the window to be used when applying CLAHE.
2. Clip_limit: The clip limit is used to clip the histogram before computing the CDF. The higher the value the more contrast can be seen in the resulting image.
3. Number of bins: The number of gray bins to bucket the values used in the histogram.

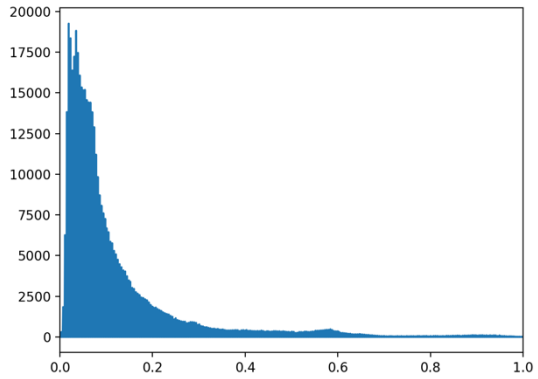
Important parameters for local histogram equalization

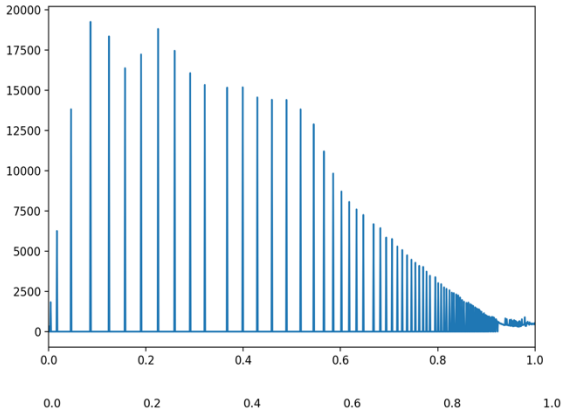
1. Mask: Used to specify what part of the image to equalize.
2. neighborhood: The neighborhood around the pixel to be used.

<u>Experiments</u>
crowd.tif

Before any histogram equalization
Number of bins = 1000

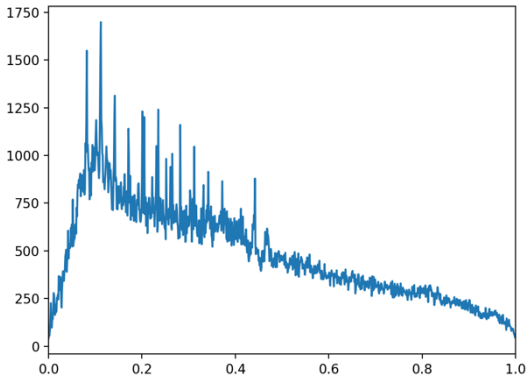
After histogram equalization
number of bins = 1000
mask=default(size of image)

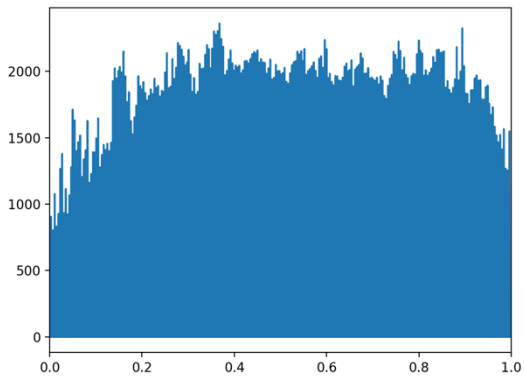




After contrast adaptive histogram equalization
Kernel_size = 1/8th of image height/width.
Clip_limit = 0.1
Number of buckets = 1000

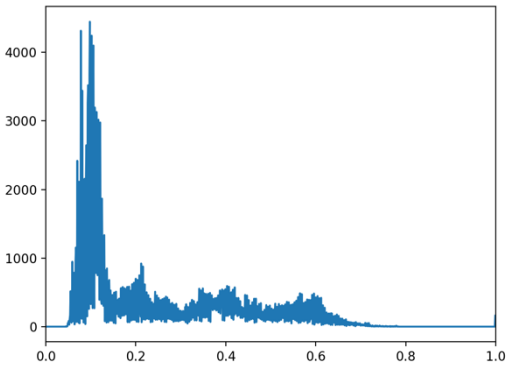
After local histogram equalization
neighborhood= 200x200
mask = default, entire image



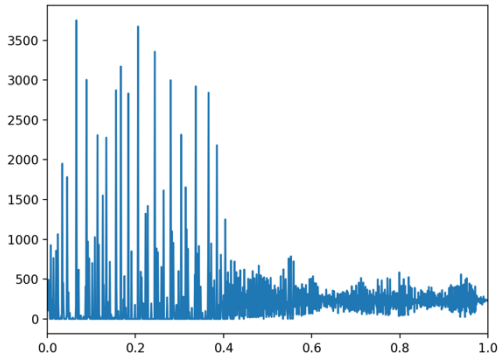


xray.png

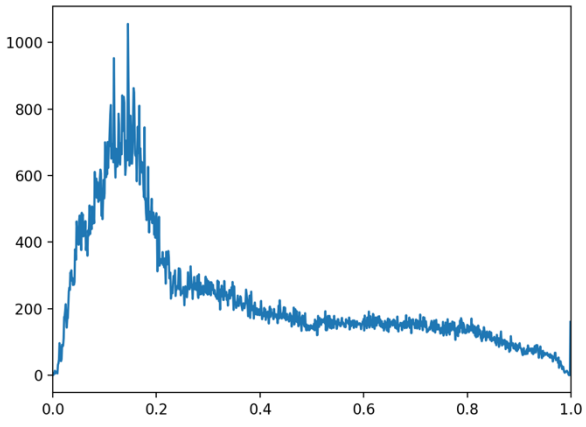
Before any histogram equalization
Number of bins = 1000



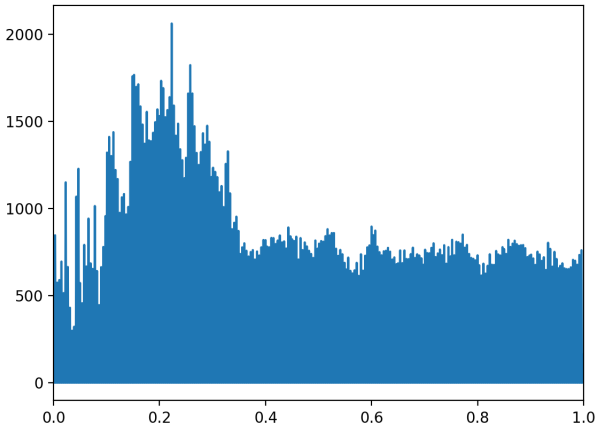
After histogram equalization
number of bins = 1000
mask=default(size of image)



After contrast adaptive histogram equalization
Kernel_size = 1/8th of image height/width.
Clip_limit = 0.01
Number of buckets = 1000



After local histogram equalization
neighborhood= 583x393
mask = default, entire image



Conclusion
From the perspective of visibility CLAHE seems to provide good results.

References

1. https://datacarpentry.org/image-processing/05-creating-histograms/
2. https://scikit-image.org/docs/dev/auto_examples/color_exposure/plot_equalize.html
3. https://scikit-image.org/docs/dev/auto_examples/color_exposure/plot_local_equalize.html
4. https://scikit-image.org/docs/0.13.x/auto_examples/xx_applications/plot_thresholding.html