

# Accelerator Beam finding from YAG screens

- Problem
  - Screen Information
  - Data
    - Description
- First Pass - just files 1 and 2
- Preprocessing
  - Pipeline
    - Regression/box prediction
    - Measure accuracy
  - Results
    - Pre processing=None, files=1,2,4
    - Best Results
  - vcc median+Gaussian Blur
  - yag median+Gaussian Blur

## Problem

Below is a sketch that Siqi made on the whiteboard about the setup. Here is my understanding of the problem (I'm not an expert in the field).

- The beam goes through the DMD
- it is then split
  - first view on the VCC - virtual cathode ?
  - second, after going through nonlinear cathode and its optics, on the YAG

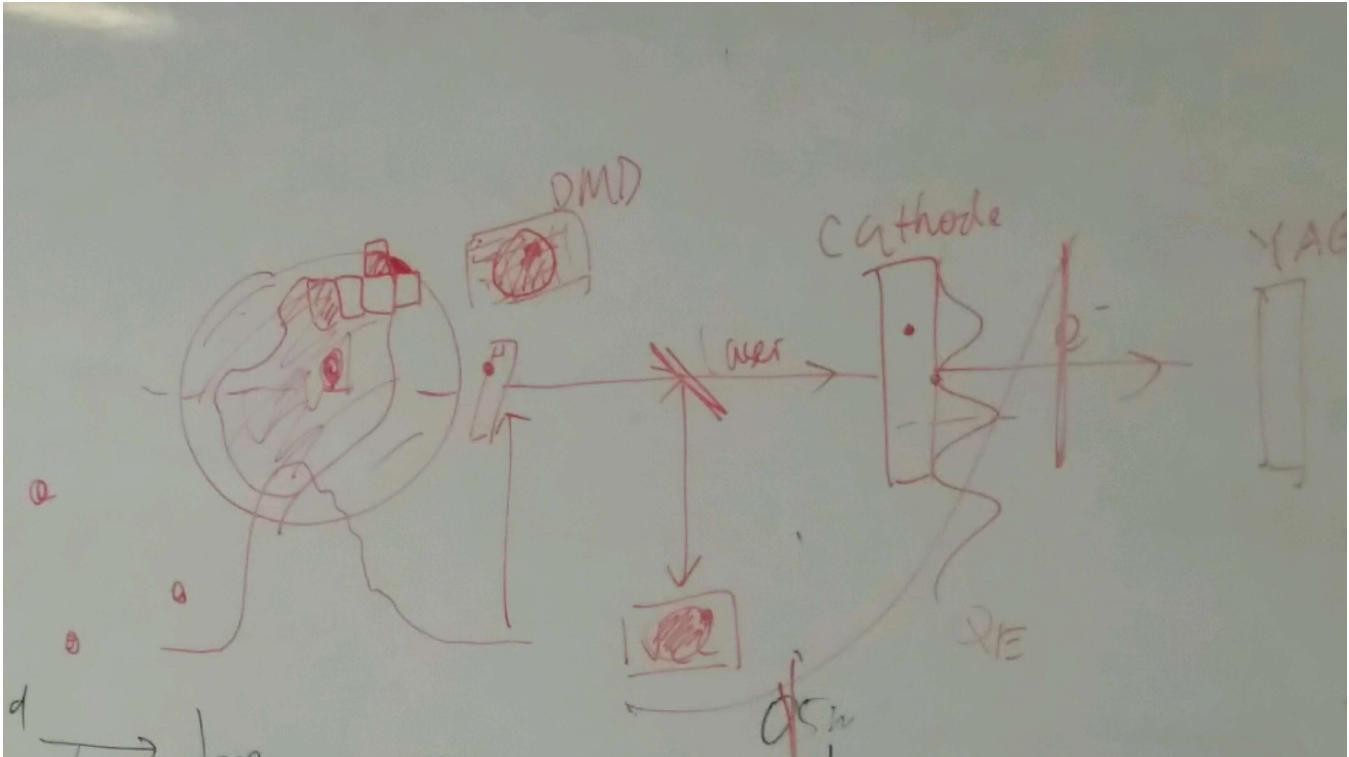
The end goal is to shape and control the beam. To do this one needs to know what happens to the beam as it goes through the cathode.

The cathode creates a non linear gain map - some parts magnify, and some parts shrink. The problem is basically calibration, figuring out what the cathode does to the beam. You can't just let the whole beam through, you need to let a little bit of the beam through at a time and see what the cathode does by looking at the YAG.

- operators can control how big of a square to open up on the DMD to let beam go through.
- A scan across the DMD is just going row by row, opening a different square of DMD pixels to let beam go through.
- One want to measure two things
  - The location of the beam on the VCC
  - The charge of the beam on the YAG
- The charge will be the sum inside the box around the beam on the YAG, so you need to know location on YAG for this

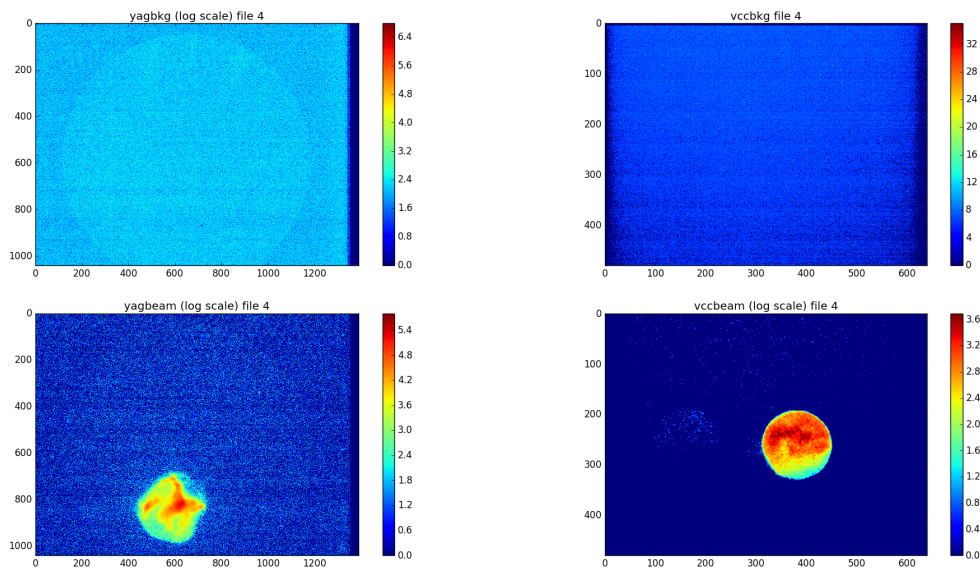
For a given file, to accurately compute charge

- first subtract the background (upper row of above plot)
- Then figure out where the beam is for YAG (bottom row, when all DMD open to let beam through)
- when you sum charge in the YAG box, restrict to the region on the YAG open beam plot,



## Screen Information

File 4 also contains the yag and vcc backgrounds, as well as shots where the entire beam is opened up (as opposed to just one location on the DMD). Here are those plots



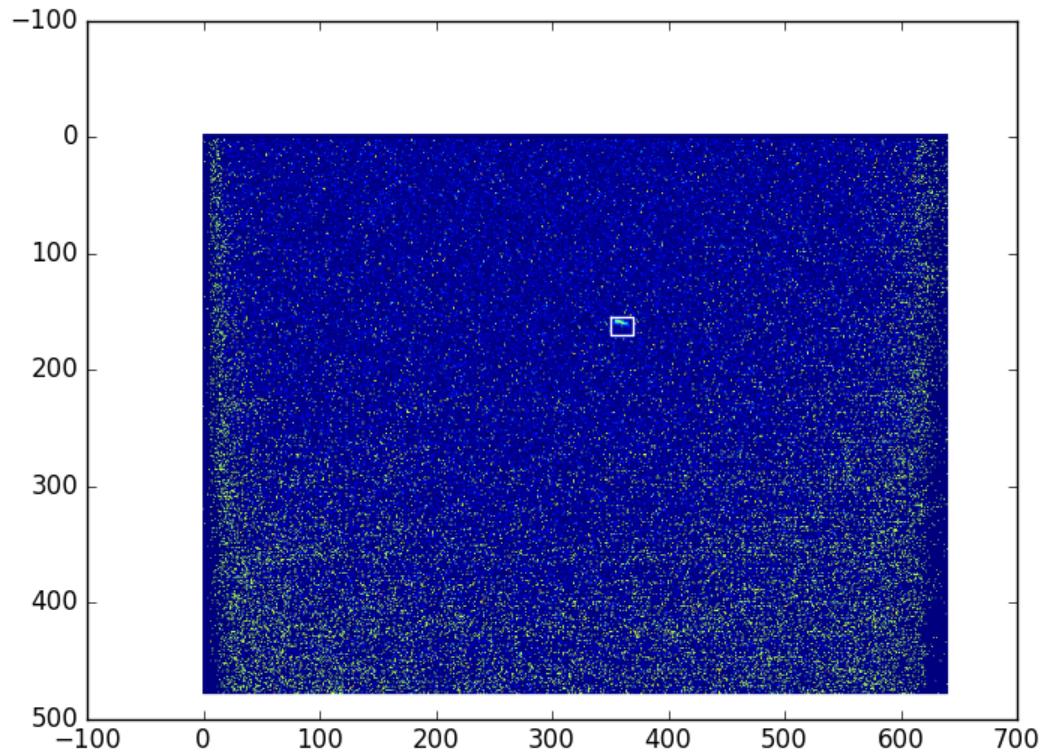
Notice the distortion on the yag. This is because the beam goes through the cathode and its nonlinear optics before showing up on the yag screen.

## Data

The page [Accelerator Beam finding - Internal Notes](#) talks about the data and code, page has restricted access.

Below is an example of the problem we are trying to solve:

- this is a vcc screen, and the location of the beam has been labeled with the white box.
- We want to use machine learning to predict these box locations.
- The dataset contains vcc screens and YAG screens - two different datasets, currently looking at training two different models - a YAG predictor and a VCC predictor.



## Description

There are 3 files, called 1,2 and 4.

- Files 1 and 2 have 142 samples. With file 4, the total number of 239 samples is.
- Each sample has a yag, vcc, and box for each
  - File 4 also contains the bkg and beam shots above.
  - Files 1 and 2 already have the bkg subtracted.
- vcc values are in  $[0, 255]$ , and the boxed beam can get quite bright
- yag values go over 1000, I think, but the boxed value is always dim, like up to 14

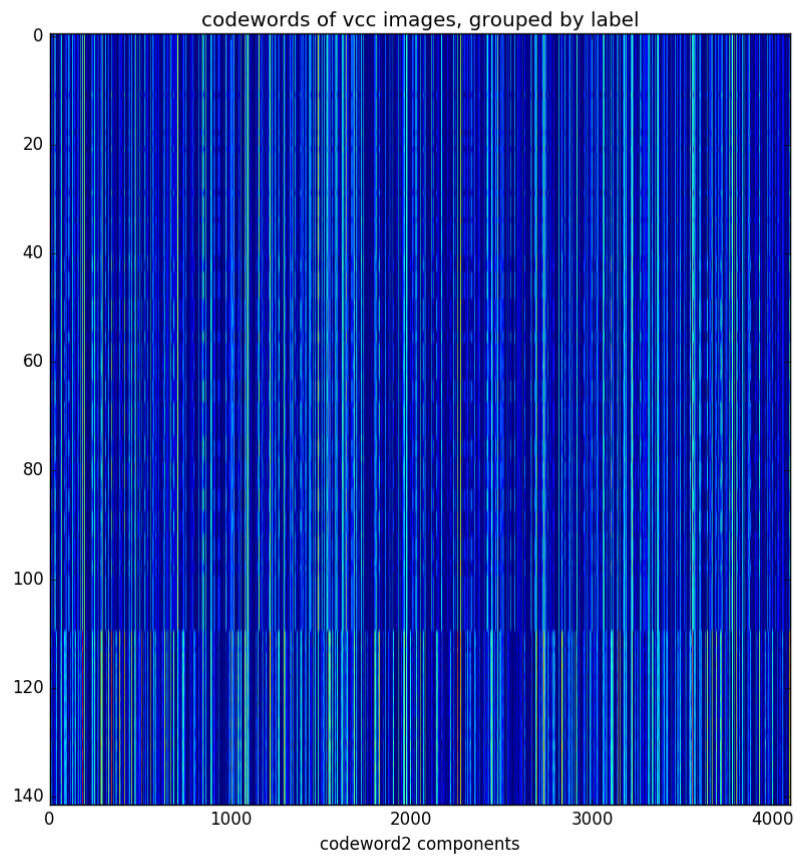
## First Pass - just files 1 and 2

Given the apparent success of using [transfer learning](#) to do [Spatial Localization](#) to find the 'fingers' in XTCAV data, we will try the same thing with the accelerator data.

We have to fit the 480 x 640 vcc images, and 1040 x 1392 yag images into the 224 x 224 x 3 RGB sized images that the [vgg16](#) convolutional neural network expects.

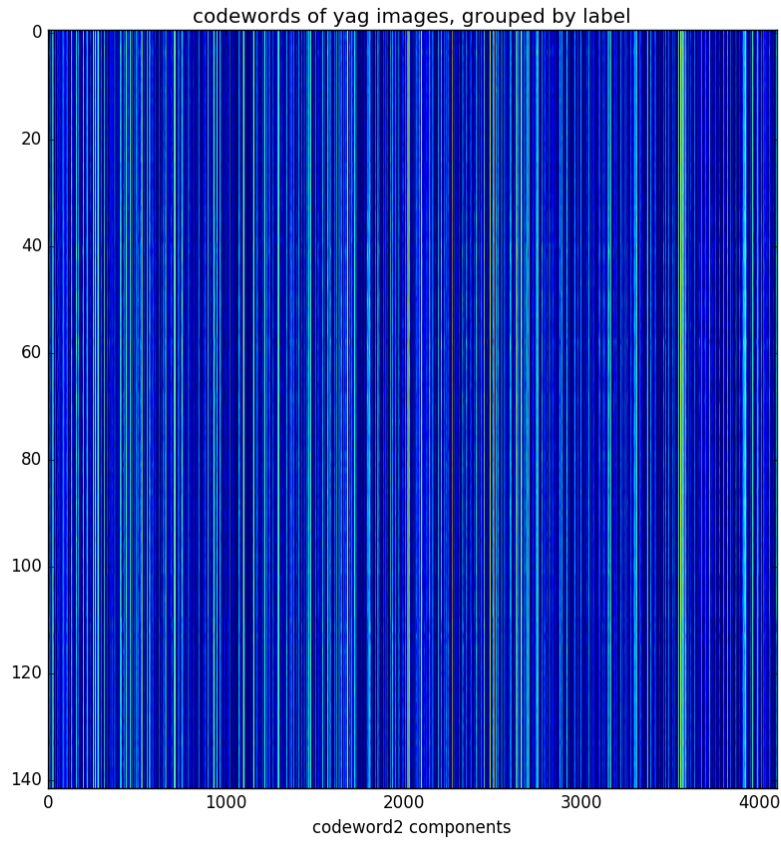
I thresholded yag at 255, then made grayscale images for each, using a `scipy imresize` option.

I generated codewords for the yag and vcc. The yag, which has bright beam, shows a lot of structure:



These are plotted with a very large aspect ratio, the bottom is the 'nobeam' images.

However with the yag images, there is very little difference between nobeam and beam:



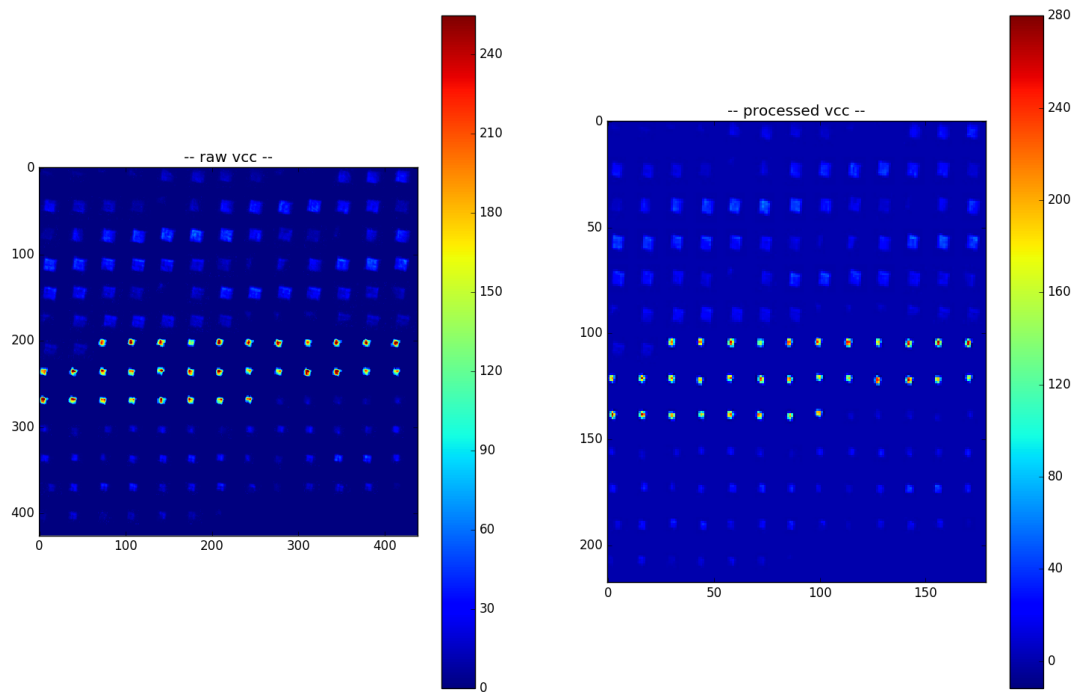
At first I thought we would not be able to do much with the yag screen codewords without more preprocessing. This may be the case for the classification problem of whether or not the beam is present, but for the regression problems of finding the box around the beam, assuming it is there, it actually does better on yag than vcc.

## Preprocessing

This problem seems harder than the [localization for lasing fingers in amo86815](#). There is more variety in the signal we are trying to find on the vcc.

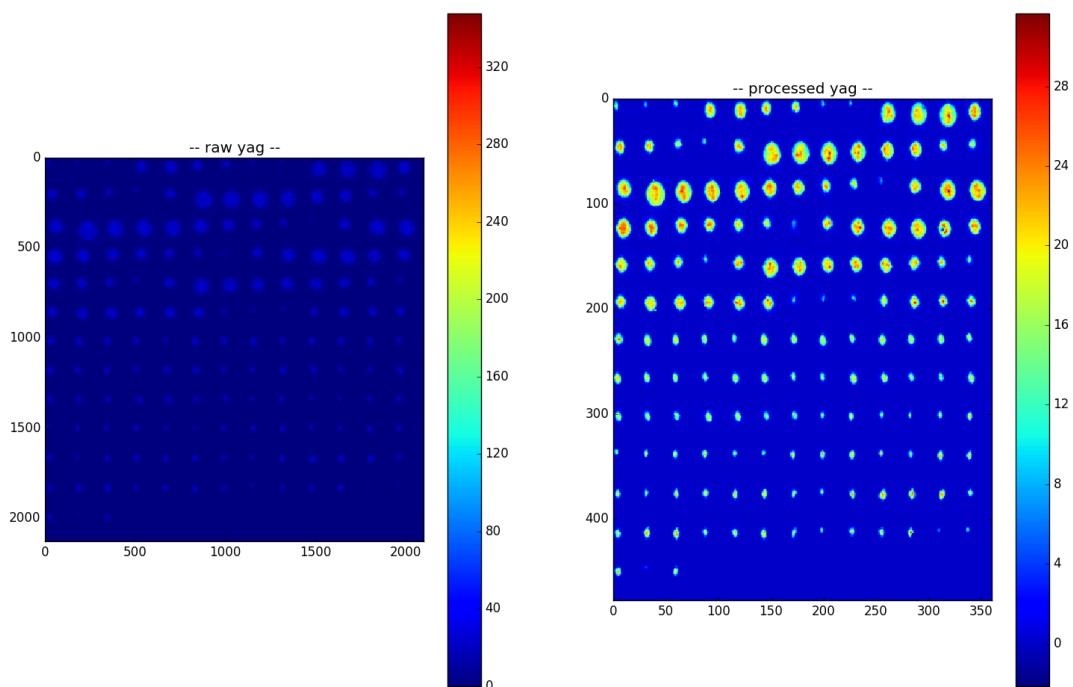
Of the 239 samples, 163 of the vcc have a labeled box. Below is a plot where we grab what is inside each box and plot it all in a grid - this is with the background subtraction for file 4. The plot on the left is before, and on the right, is after reducing the 480 x 640 vcc images to (224,224) for vgg16. We used `scipy imreduce 'lanczos'` to reduce (this calls PIL). Here, there is no preprocessing other than what the image size reduction does

pre-processing Alg: none



Here are the 159 samples of the yag with a box - here are using 'lanczos' to reduce from the much larger size of 1040 x 1392 to (224,224). It is interesting to note how the colorbar changes - the range no longer goes up to 320 - I think the 320 values were isolated pixels that get washed out? Or else there is something else I don't understand - we are doing nothing more than `scipy.misc.imresize(img,(224,224), interp='lanczos',mode='F')` but `img` is `np.uint16` after careful background subtraction - (going through `float32`, thresholding at 0 before converting back)

pre-processing Alg: none





## Pipeline

### Regression/box prediction

The processing pipeline for the regression

1. choose a preprocessing algorithm
2. create vgg16 codeword1 and 2 (8196 numbers, last two layers)
3. separately for 'yag' and 'nm'
  - a. for each of the for each of the 163 (or 159) samples,
    - i. train a linear regression classifier on the remaining 162 (or 158) samples
      1. map from 8196 variables to 4
    - ii. use it to predict a box for the omitted sample
    - iii. optionally - limit input features - reject features with variance less than threshold
      1. (maybe they are noisy and throwing off classifier)

### Measure accuracy

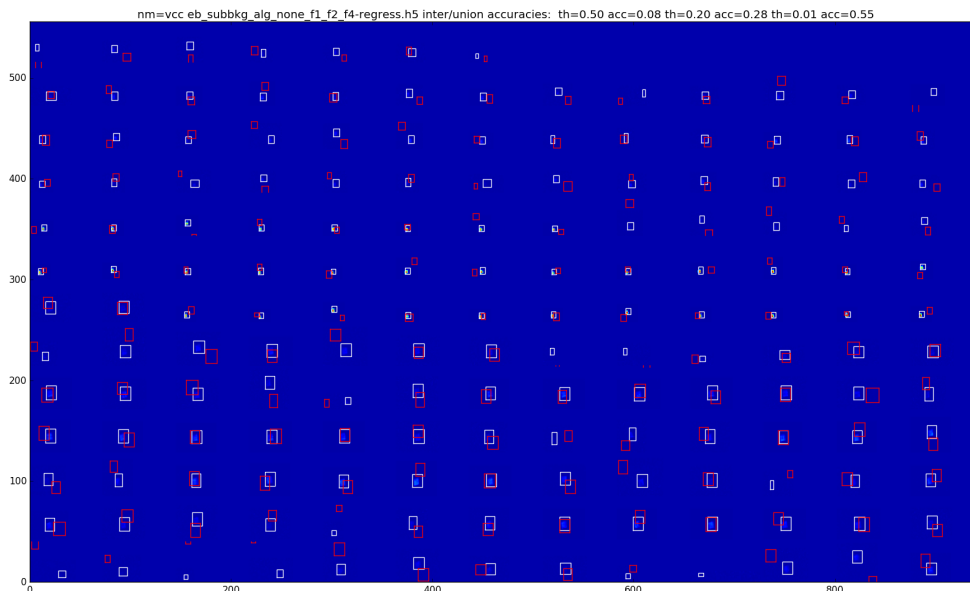
For localization - on a shot by shot basis, where we are comparing boxA to boxB, one typically calculated the ratio of area of the intersection to area of the union. For imagenet competitions, one gets success on a shot /image of  $\text{inter/union} \geq 0.5$ , those predictions look quite good! One can then come up with a overall accuracy based on the inter/union threshold. Below, we report on accuracy for different thresholds, .5, .2 and .01 - the latter is to see how accurate we are at getting any overlap.

To visualize the results, we make a similar plot to above, but plot the truth box in white, and the predicted box in red.

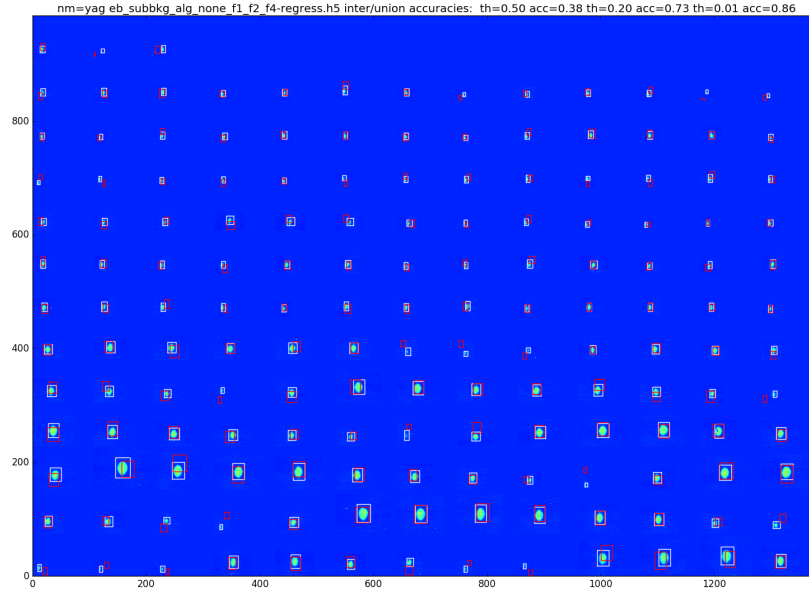
## Results

### Pre processing=None, files=1,2,4

For the vcc, accuracies, a 1% overlap is 55%



For the yag, a 1% overlap is 86%:



## Best Results

The best results have been obtained using some signal preprocessing developed by Adbullah Ahmed. The de-noising is roughly:

- vcc: threshold at 255
- opencv medianBlur
  - yag: 5pt
  - vcc: 7pt
- opencv guassainBlur
  - yag: 55 x 55
  - vcc: 15 x 15
- yag: threshold at 1.5 (where < 1.5, set to 1.5)
- lanczos reduction

After doing the de-noising, and before the reduction, we find the maximum value in the image and call it a hit if it is in the labeled box. This signal processing solution performs quite well. Over files 1,2,4 and doing the background subtraction for file 4, it does:

- 100% for the yag
- 99% for the vcc
  - one of the vcc boxes is mislabeled though
  - the other one, it is close to the box, the gaussian blur took a longer shape with some nearby noise and made it more round (we guess)

The regression pipeline does quite well on the yag, but less well on the vcc

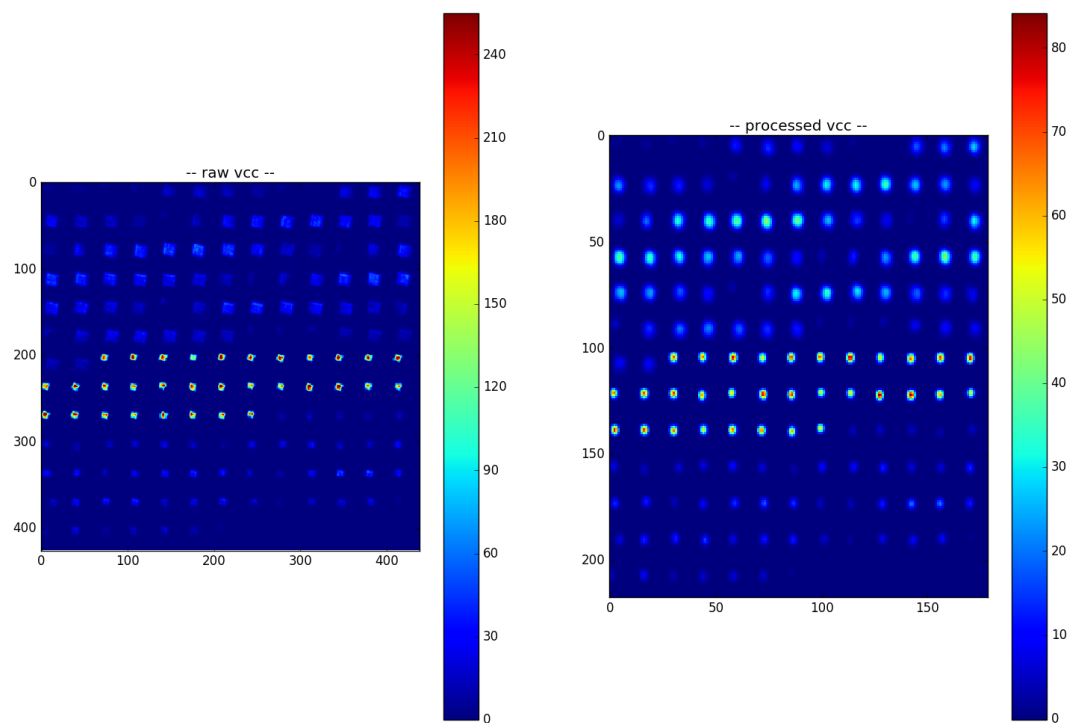
- yag: inter/union accuracies: th=0.50 acc=0.89 th=0.20 acc=0.97 th=0.01 acc=0.98
- vcc: inter/union accuracies: th=0.50 acc=0.09 th=0.20 acc=0.38 th=0.01 acc=0.66

## Preprocessing Plot

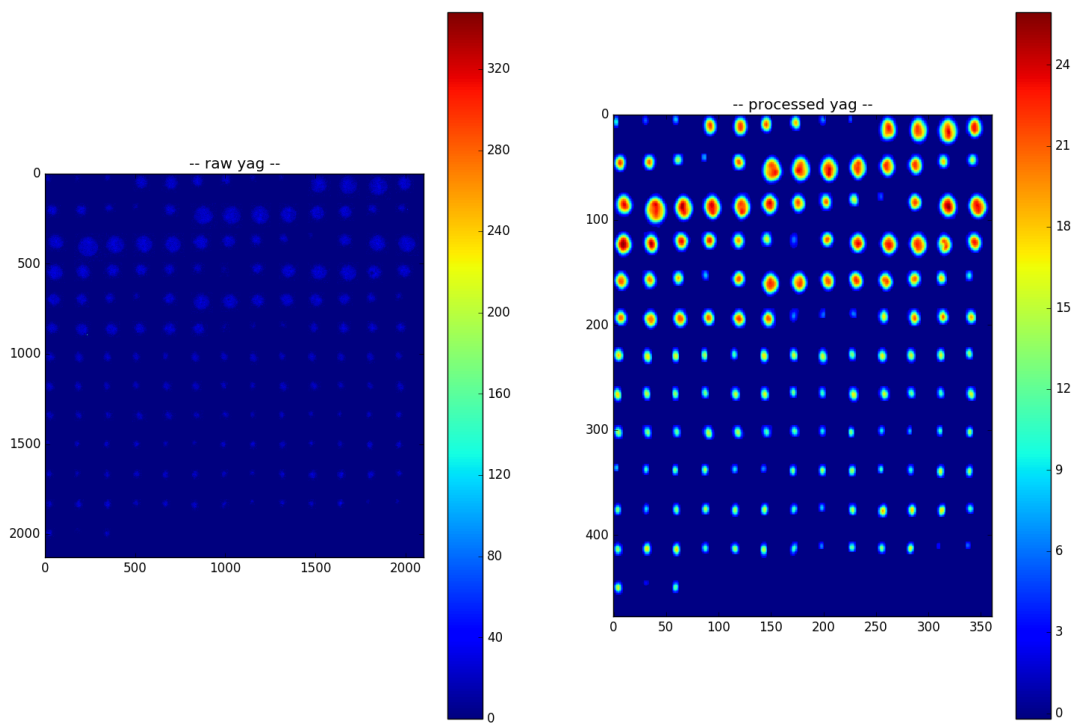
Here is a plot of the preprocessing - the main benefit from it is the de-noising, which is not as apparent in this plot



pre-processing Alg: med-gaus

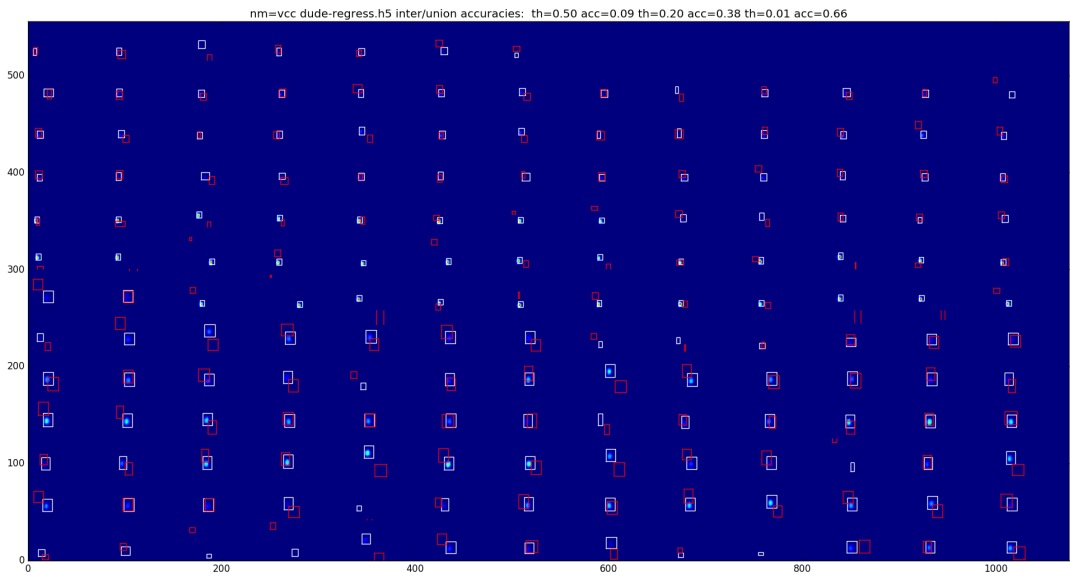


pre-processing Alg: med-gaus



Here are plots to show the regression results

vcc median+Guassain Blur



yag median+Guassain Blur

