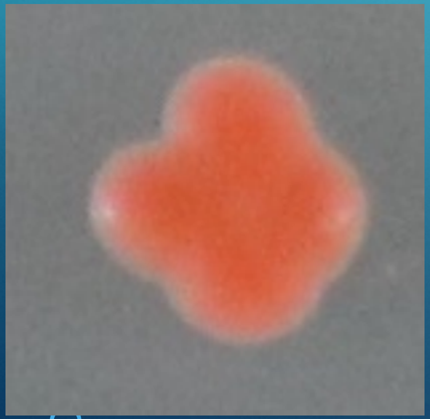
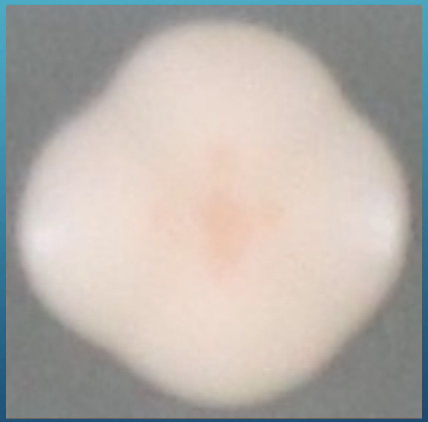
DEVELOPMENT OF A YEAST COLONY CLASSIFICATION SYSTEM USING COMPUTER VISION.

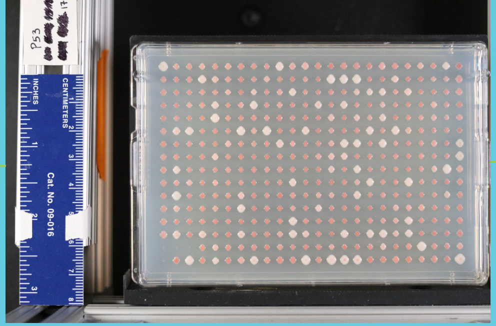
GREG GLICKERT

PROBLEM STATEMENT

High throughput phenotyping generates a large amount of data and classifying this data by hand presents many problems such as being tedious and bias.

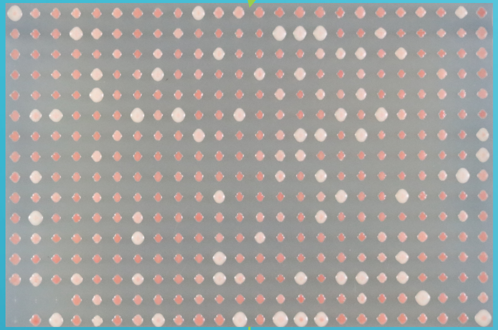


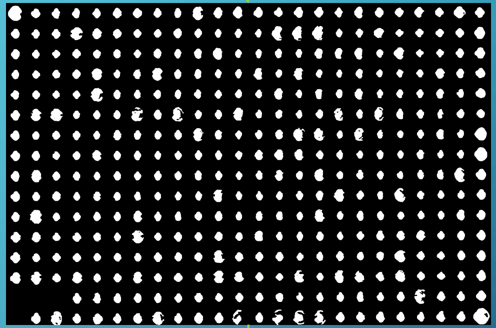




Color processing

Size processing



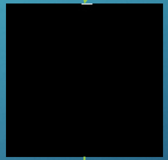


White cell

Red cell

No growth

Growth



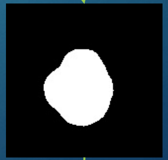


	Image processed	Cluster	Size	color	red
0	A.JPG	U1-A1	9547	23.19895857	20.13667827
1	A.JPG	U2-A1	5545	50.52499024	52.70208378
2	A.JPG	U3-A1	5620	50.23032798	52.66768898
3	A.JPG	U4-A1	5312	35.74966774	37.43292378
4	A.JPG	U1-A2	5072	48.01897639	49.47979755
5	A.JPG	U2-A2	10041	14.36643812	8.030703864
6	A.JPG	U3-A2	4118	46.79006758	47.52796333
7	A.JPG	U4-A2	4361	46.62439391	47.35590228
8	A.JPG	U1-A3	4777	51.3006584	52.35181329
9	A.JPG	U2-A3	7187	19.4842296	14.14730308
10	A.JPG	U3-A3	10531	10.61116946	0

TECHNICAL OVERVIEW

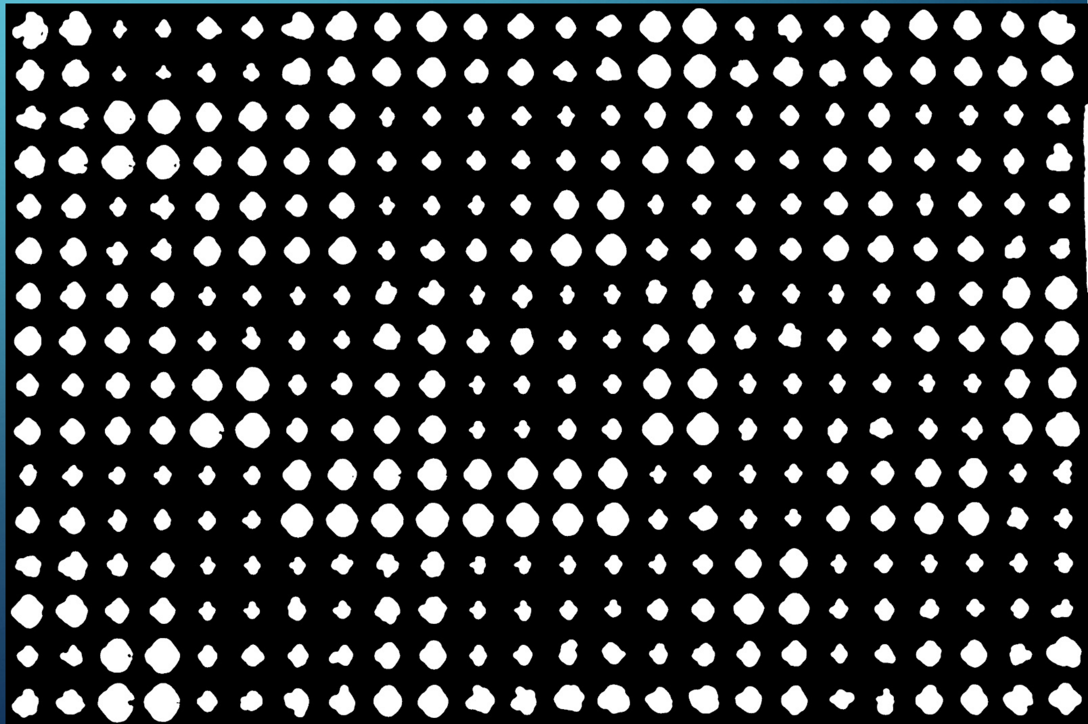
- Script is written in python 3.7
- Script runtime is around 80 seconds per image on a 2.3 GHz X86_64 core
  - Biggest contributor to the runtime is finding the plate which can be improved
- Dependencies for script are
  - Pillow, cv2, NumPy, os, pandas, matplotlib, imutils, SciPy, easygui, tqdm, xlrd, SKlearn

CROPPING THE PLATE

- 96 well, 384 or in the future 1536 well plates can be cropping into cluster of four or into individual cells
- User selects how they want the image cropped
- Uses the pillow library

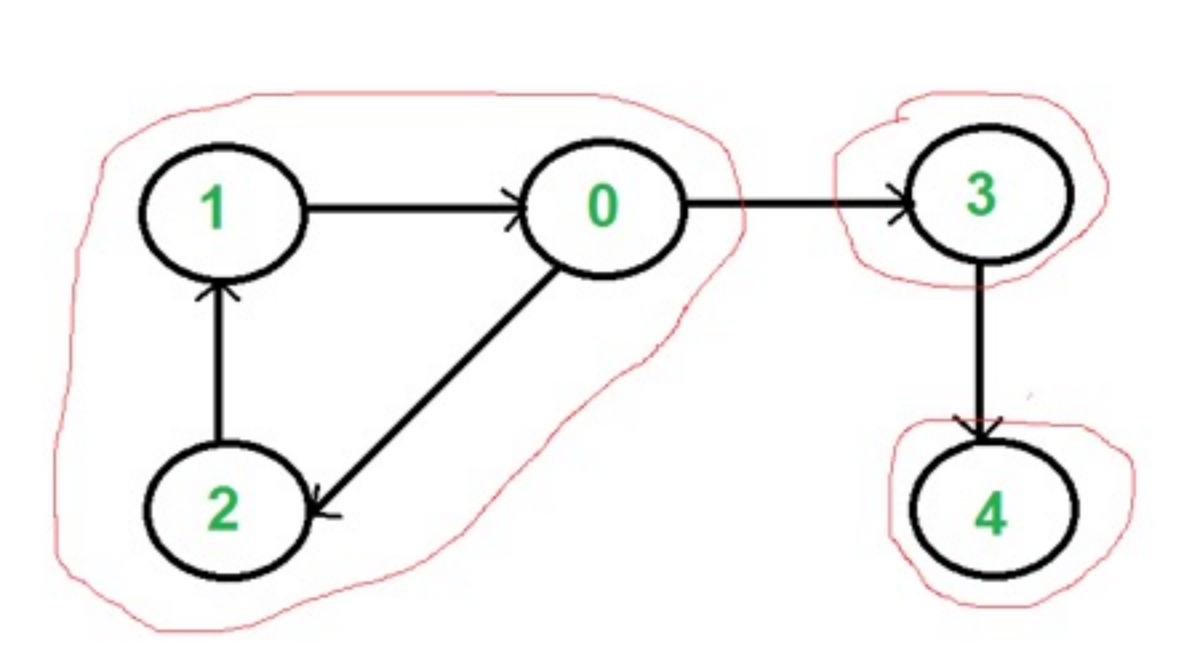
THRESHOLDING

- Takes the cropped image and turns background black and cells white
- Uses open computer vision library to threshold and remove anything that turns white that’s not a cell



HOW THE SIZE IS FOUND

- The size is found using a connected components algorithm available in the open computer vision library
- Algorithm gives the size of each cell and the location of each cell



HOW COLOR DATA IS FOUND

- Color data is split into channels then analyzed to return the color metric
- Redness determined similar but first all color but red is removed from the image

```python
# split the image into its respective RGB components
(B, G, R) = cv2.split(image.astype("float"))  # CV2 works in BGR not RGB
# compute rg = R - G
rg = np.absolute(R - G)
# compute yb = 0.5 * (R + G) - B
yb = np.absolute(0.5 * (R + G) - B)
# compute the mean and standard deviation of both `rg` and `yb`
(rbMean, rbStd) = (np.mean(rg), np.std(rg))
(ybMean, ybStd) = (np.mean(yb), np.std(yb))
# combine the mean and standard deviations
stdRoot = np.sqrt((rbStd ** 2) + (ybStd ** 2))
meanRoot = np.sqrt((rbMean ** 2) + (ybMean ** 2))
# derive the "colorfulness" metric and return it
return stdRoot + (0.3 * meanRoot)
```

WHAT HAPPENS TO THE DATA?

- The data is sent to a pandas data frame
  - Can be exported into xlsx or csv format

	Image processed	Cluster	Size	color	red
0	A.JPG	U1-A1	9547	23.19895857	20.13667827
1	A.JPG	U2-A1	5545	50.52499024	52.70208378
2	A.JPG	U3-A1	5620	50.23032798	52.66768898
3	A.JPG	U4-A1	5312	35.74966774	37.43292378
4	A.JPG	U1-A2	5072	48.01897639	49.47979755
5	A.JPG	U2-A2	10041	14.36643812	8.030703864
6	A.JPG	U3-A2	4118	46.79006758	47.52796333
7	A.JPG	U4-A2	4361	46.62439391	47.35590228
8	A.JPG	U1-A3	4777	51.3006584	52.35181329
9	A.JPG	U2-A3	7187	19.4842296	14.14730308
10	A.JPG	U3-A3	10531	10.61116946	0

	Image processed	Cluster	Q1_size	Q2_size	Q3_size	Q4_size	Avg_size
0	01.JPG	U1-A1	2938	3807	4959	4792	4124
1	01.JPG	U1-A2	2129	2965	3777	4080	3237.75
2	01.JPG	U1-A3	3484	3903	4407	4653	4111.75
3	01.JPG	U1-A4	2656	3089	3343	3285	3093.25
4	01.JPG	U1-A5	3090	3563	3728	3727	3527
5	01.JPG	U1-A6	2686	3334	3664	3753	3359.25
6	01.JPG	U1-A7	2808	3252	3527	3765	3338
7	01.JPG	U1-A8	2772	3115	3446	3715	3262
8	01.JPG	U1-A9	1407	1706	1979	2208	1825

WHAT CAN YOU DO WITH THIS DATA?

- Train a machine learning model to learn and predict which cells are red and which cells are white.
- Improvements
  - More labeled data
  - Feature engineering
  - More advanced models

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

```
Confusion Matrix for lr
[[ 14   1]
 [  2 137]]
accuracy for lr
0.9805194805194806
```

```
Confusion Matrix for svc
[[ 15   0]
 [  4 135]]
accuracy for svc
0.974025974025974
```

```
Confusion Matrix for mlp
[[ 15   0]
 [  4 135]]
accuracy for mlp
0.974025974025974
```