Bubble Detection

Filip Peterek

VŠB-TUO

September 2024

Dataset

- Dataset from CAS
- Manual annotation was necessary
- Original dataset contains noise, blurry bubbles
- Only bubbles recognizable by human eye are considered
    - More than a couple pixels in diameter
    - Not too blurry

First Training Set

- Bounding boxes around bubbles
- Training set consists of 16 images containing 2766 bubbles
- Testing set consists of 2 images containing 509 bubbles
- Used to train RCNNs

Second Training Set

- Singular bubbles were extracted
- Roughly 2500 images of singular bubbles
- Roughly 2000 images of smidges, noise or clusters of bubbles
- Used to train Tsetlin Machines

RCNN Detection

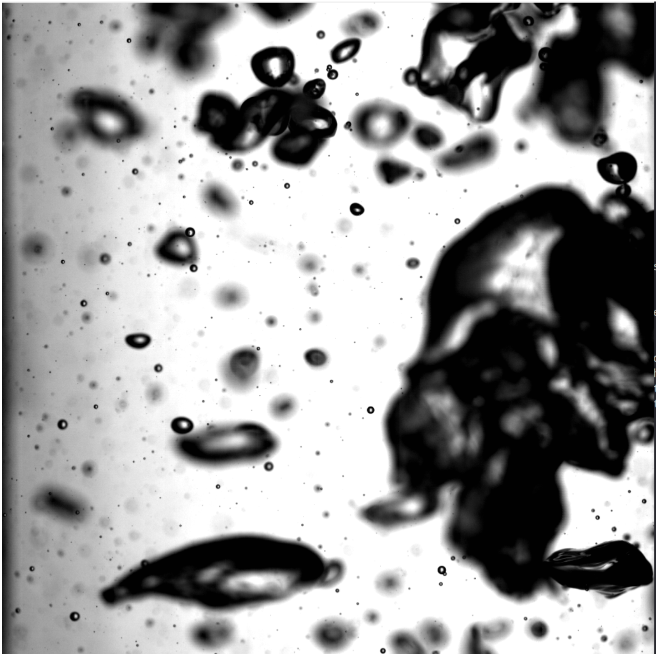
- Faster RCNNs were used for bubble detection
- Resnet-50-FPN
- Transfer learning was applied
- Has trouble detecting very small bubbles
- Sometimes misses overlapping bubbles
  - When both bubbles are very dark
  - Hardly recognizable even by human eye
- Couldn't suffice by itself
- Adjusting contrast improved detection

Segmentation and Tsetlin Machines

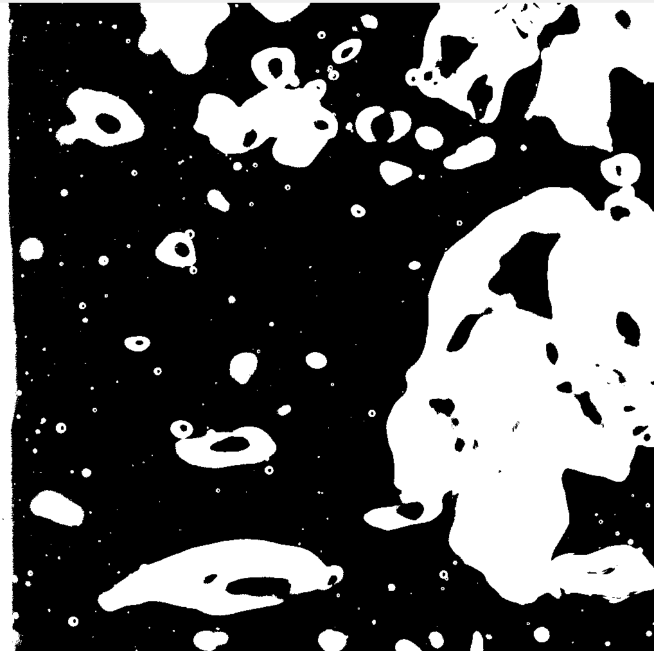
- To catch what RCNN missed, segmentation is used
- First, contrast is adjusted
- Then, thresholding is applied
- Simple flood fill algorithm selects objects in image
- Histogram of oriented gradients is computed for each object
- Tsetlin Machine then determines whether object is a singular bubble

Segmentation and Tsetlin Machines

Segmentation and Tsetlin Machines



Segmentation and Tsetlin Machines



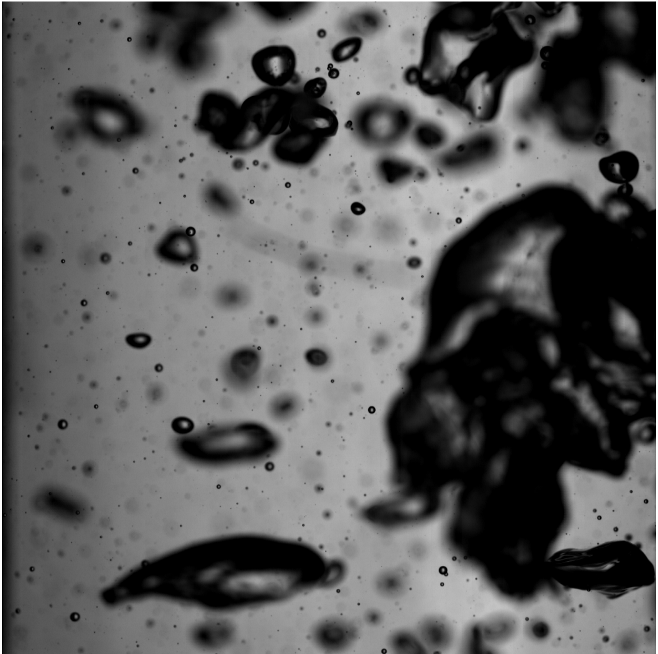
Combining Results

- RCNN and Segmentation results are merged together
- Intersection over union is computed
- If IOU for two objects passes a certain threshold, we consider them the same object
- Thus, duplicates are filtered
- Segmentation results are preferred over RCNN results
  - Prevents margins around bubbles

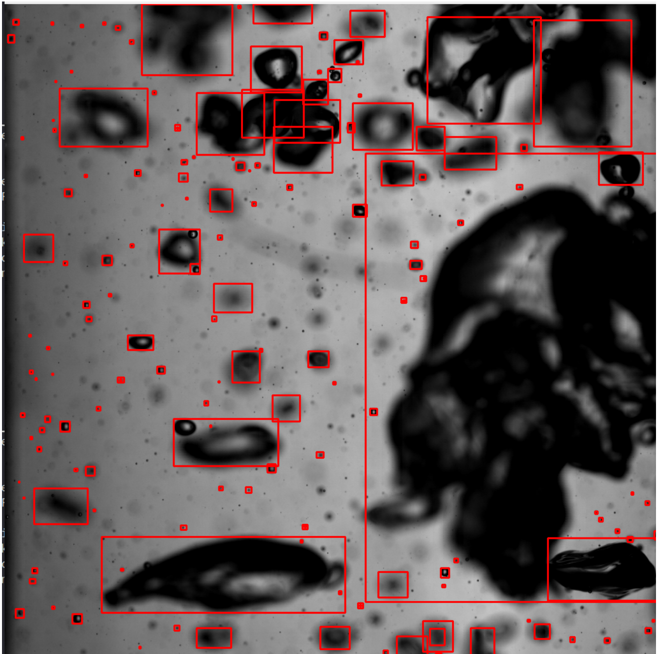
Results

- Around 76% accuracy
- Six false positives
- Detection using Resnet takes 236 ms
- Recognition using Tsetlin Machines takes 79 ms
  - Includes HOG computation
- In total, processing one image takes 1506 ms
- Mainly the result of an inefficient flood fill algorithm
  - 990 ms
- Written in Python, can by optimized

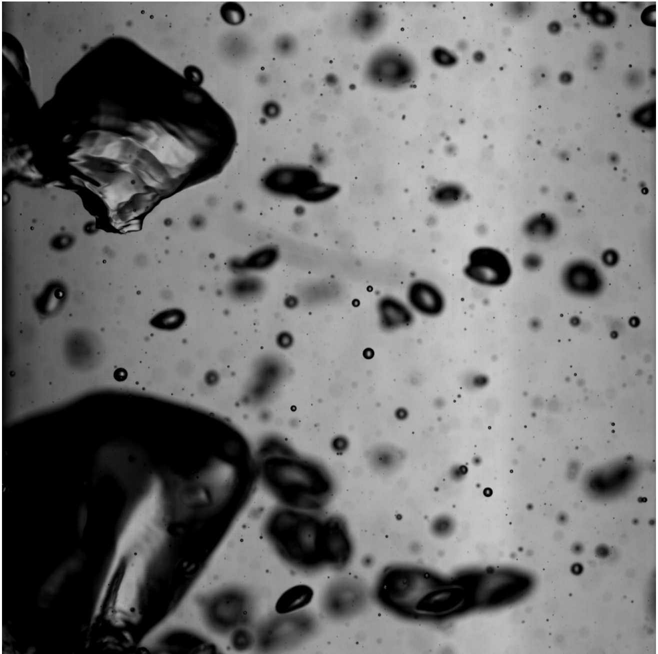
Results



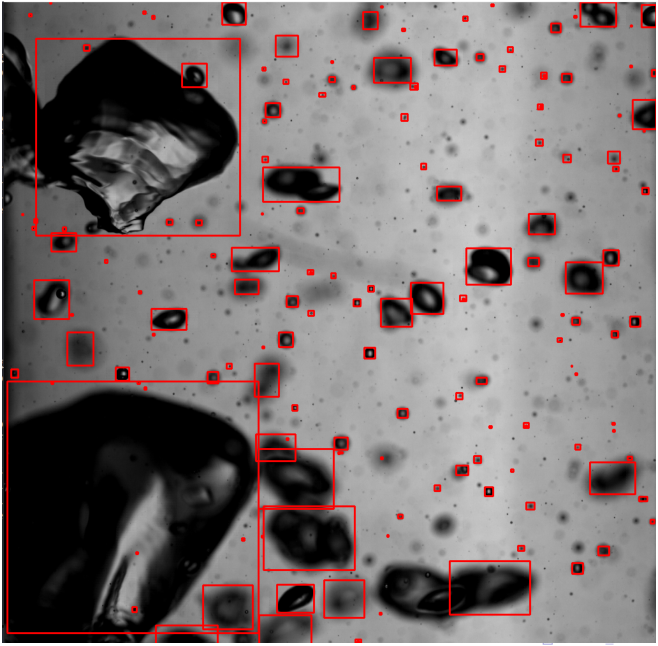
Results



Results



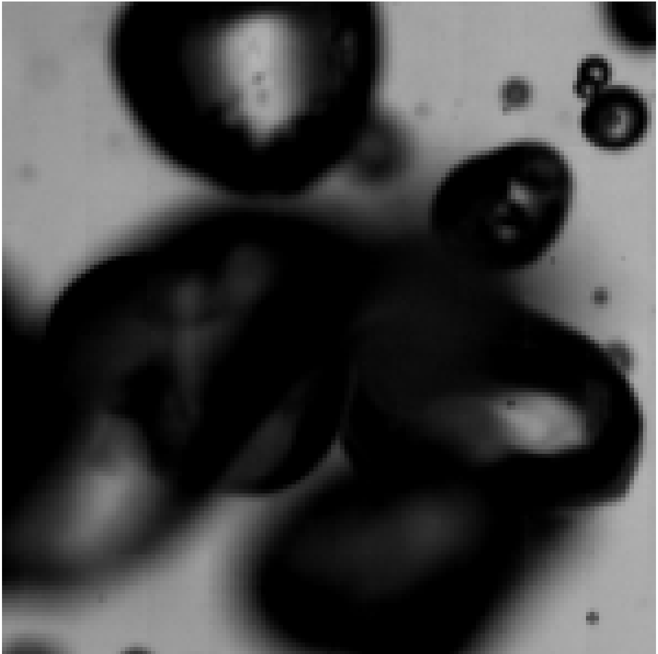
Results



Challenges

- Very dark transitions

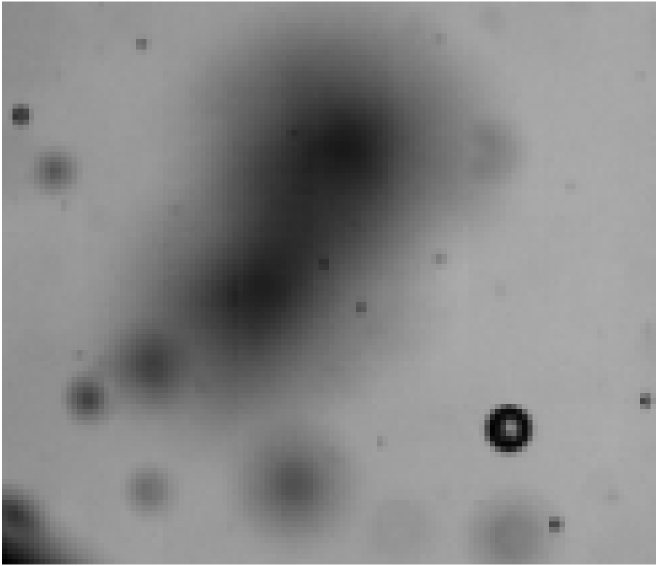
Challenges



Challenges

- Very dark transitions
- Blurry bubbles

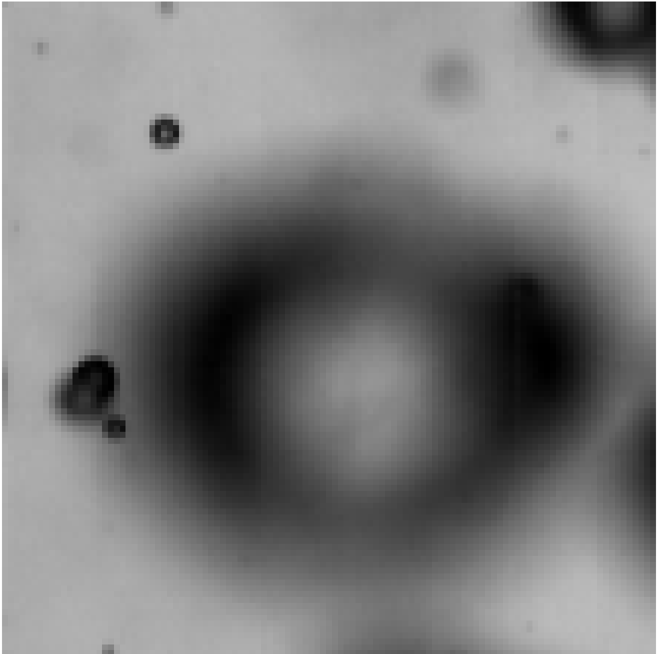
Challenges



Challenges

- Very dark transitions
- Blurry bubbles
- Bright spots where light reflects
  - Breaks thresholding

Challenges



Failed Experiments

- Local Binary Pattern
- Background subtraction
- Convolutional Tsetlin Machine