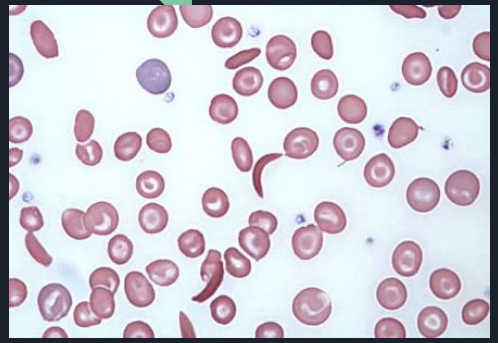
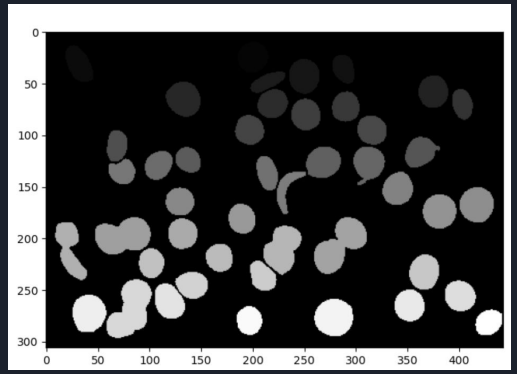
Sickle Cell Classification

By: Ori Weiss

Image Preparation






```
def image_prep(path):
    mask_size = 400
    img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
    edges = canny(img/255.)
    fill_coins = ndi.binary_fill_holes(edges)
    label_objects, nb_labels = ndi.label(fill_coins)
    sizes = np.bincount(label_objects.ravel())
    mask_sizes = sizes > mask_size
    mask_sizes[0] = 0
    img_cleaned = mask_sizes[label_objects]
    labeled_img, num_features = ndi.label(img_cleaned)
    return labeled_img, num_features
```

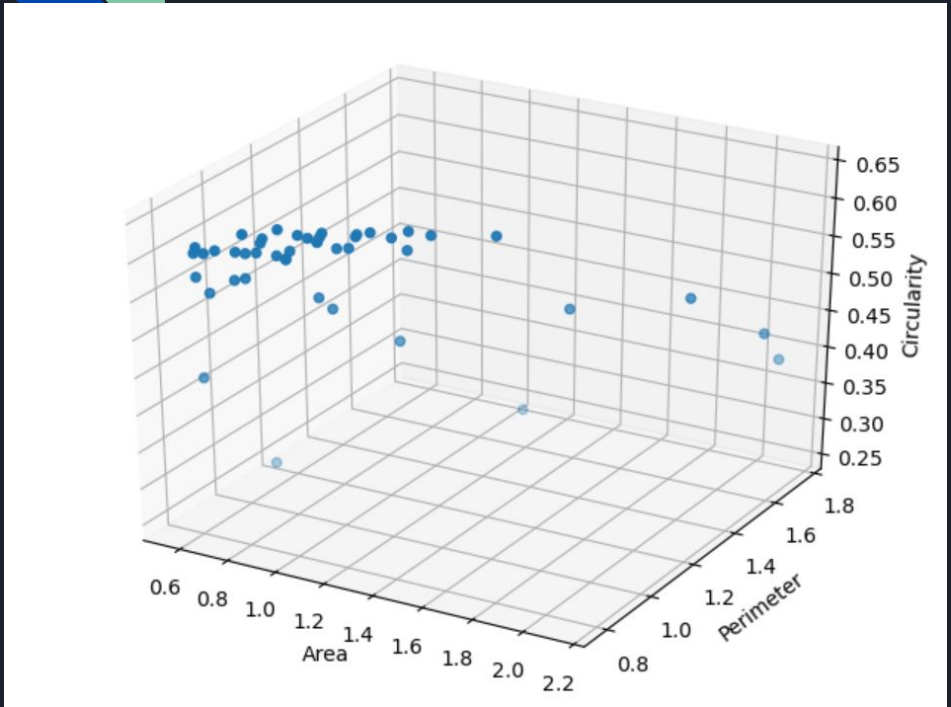
- Python libraries used; OpenCV, numpy skimage, scripy-ndimage, matplotlib
- Image Prep Steps
  - Read in image,
  - convert to 8 bit black and white,
  - Detect Edges
  - Fill in holes
  - Use mask to filter out undesirable cells
  - Label each cell
- Resulting image should become a 2D (black and white) array with pixel values corresponding to the cell's individual ID

Extract Individual Cell Features

```
def findperimeter(mat, num_features):
    perimeter = [0] * (num_features+1)

    # Traversing the matrix and finding ones to
    # calculate their contribution.
    for i in range(len(mat)):
        for j in range(0, len(mat[i])):
            if (mat[i][j] != 0):
                perimeter[mat[i][j]] += (4 - numofneighbour(mat, i, j, mat[i][j]))

    return perimeter
def extract_area_perim(img,num_features):
    area = [0] * (num_features+1)
    for i in range(len(img)):
        for j in range(len(img[i])):
            value = img[i][j]
            if(value != 0):
                area[value]+=1
    return area, findperimeter(img,num_features)

def extract_circularity(area, perimeter):
    circularity = [0] * (len(area))
    for i in range(len(area)):
        circularity[i] = 4 * (math.pi * area[i])/(math.pow(perimeter[i],2))
    return circularity

def convert_to_relative(cellAreas, cellPerims):
    averageCellSize = np.mean(cellAreas)
    averagePerimSize = np.mean(cellPerims)
    relativeArea = [0] * len(cellAreas)
    relativePerim = [0] * len(cellPerims)
    for i in range(len(cellAreas)):
        relativeArea[i] = cellAreas[i]/averageCellSize
        relativePerim[i] = cellPerims[i] / averagePerimSize
    return relativeArea, relativePerim
```

- Extract the area and perimeter of each cell by:
  - Counting all border pixels of each cell
  - Counting all pixels of the same number
- Extract each cell's sphericality or circularity by using the formula:
  - (4 * pi * AREA) / (PERIMETER ^ 2)
- Retrieve the relative perimeter and area of each cell by using the formulas:
  - RelativeArea =
    - Area[i] / AverageArea
  - RelativePerimeter =
    - Perimeter[i] / AveragePerimeter

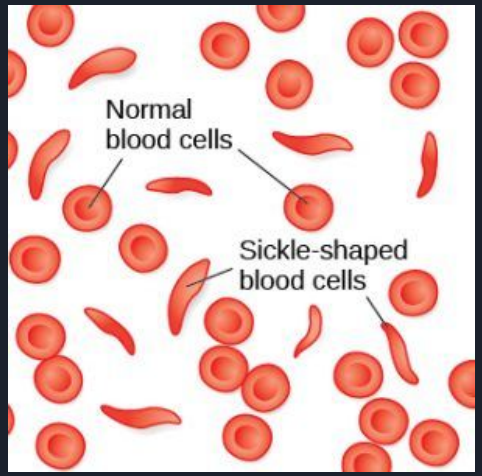
Graphing



- We can now Graph our features onto a 3D array
- This shows us how each cell's features compare to each other on a scatter plot
- We can use this information to later classify our cells

Creating Training Data

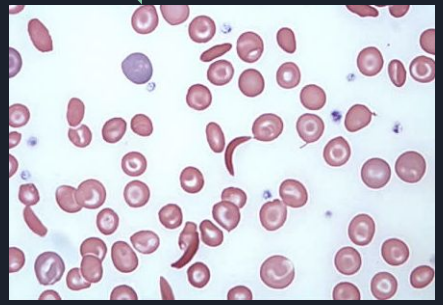
```
def getSickleData():
    return [
        [0.5775517617144933, 1.1320754716981132, 0.256951250988054 05],
        [0.5598438067562659, 0.8647798742138365, 0.426841183677819 87],
        [0.7349189934505344, 0.9518477043673013, 0.463520798439684 3],
        [0.7624956911409859, 1.097424412094065, 0.36178690908687 12],
        [0.7493967597380213, 1.097424412094065, 0.35557176327073 15],
        [0.7838676318510859, 0.9742441209406495, 0.471923747804412 04],
        [0.6494464944649447, 1.226705796038151, 0.250196090832982 17],
        [0.6621530563131719, 0.9361702127659575, 0.437992554855414 05],
        [0.6029237922610134, 0.9192918510804479, 0.381688827071657 13],
        [0.5357396169910946, 1.1168966414996095, 0.229763817741833 38],
        [0.8200803845850737, 0.9966154647227284, 0.441728099033285 4],
        [0.7186539522420995, 0.8935173131996876, 0.481579199302133 6],

    ]

def getHealthyData():
    return [
        [1.0177914720206738, 0.9837898267188373, 0.623125815588058 1],
        [0.8673756750488685, 0.9167132476243711, 0.611592026107828 5],
        [1.5326508299373818, 1.285634432643935, 0.5494520837620561],
        [1.01514097339562, 0.9837898267188373, 0.621503092109964 2],
        [0.8832786667991915, 0.9167132476243711, 0.622805325287804 1],
        [0.7931617135473611, 0.8831749580771381, 0.602545490521868 7],
        [0.8143657025477917, 0.8831749580771381, 0.618636406444249],
        [1.4955438491866282, 1.330352152040246, 0.5007114341608758],
        [0.7467779876089189, 0.8831749580771381, 0.567308912128776 9],
        [0.9382765132690587, 0.9390721073225266, 0.630455668577543 5],
        [0.9064705297684126, 0.9390721073225266, 0.609084289981694 6],
        [0.75274160951529, 0.8719955282280604, 0.5865958669424729],
        [0.6984063877016864, 0.8160983789826719, 0.621362104875894 5],
        [0.7348507437961767, 0.8272770088317495, 0.636235619582008 8],
        [1.1847728853990656, 1.073225265511459, 0.609501699719894 8],
        [1.0992943047410793, 1.0396869759642258, 0.602601712603245],
        [1.1841102607428022, 1.0844046953605366, 0.596665540648842 7],
        [0.6891296425139979, 0.8719955282280604, 0.537024385229024 4],
        [0.754066858827817, 0.9055338177752934, 0.544906636150767 3],
```



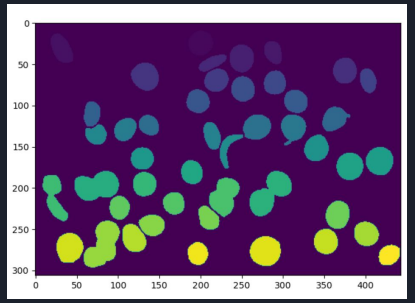
- Done by person
- Look at an image and separate out the area, perimeter, and circularity of sickle cells and healthy cells into two different groups or arrays
- Mark those groups for future use

Classification using KNN

Original Image

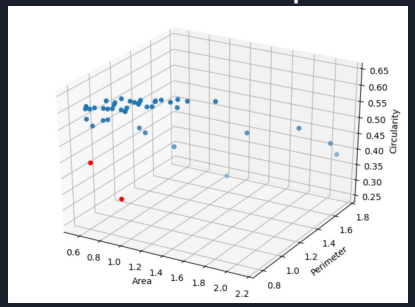


Prepped Image

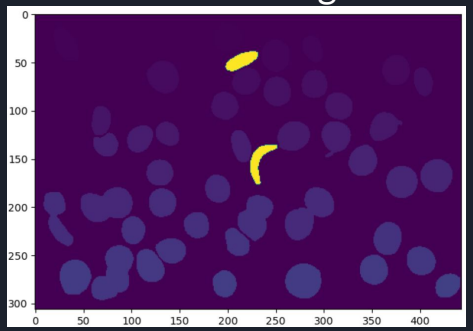


- Using our preloaded training data, we can classify new data using K-Nearest-Neighbor Algorithm
- My algorithm is set to use k = 3
- We can then see which cells are classified as sickle, highlight them on the image and graph

Classified Graph



Classified Image




```
Total Cells:  45
Sickle Cells:  2
Healthy Cells:  43
Percent Sickle:  4.444444444444445 %
Percent Healthy:  95.55555555555556 %
```

Discussion & Conclusion

- The program was able to successfully classify the majority of sickle cells in any given image
- However, if a sickle cell was overlapping a healthy cell, the algorithm was not able to classify these cells as different, it classified them as one giant healthy cell.
- These Cells are seen as outliers in the graph, with a giant area
- They can be removed in later versions of this program by removing any cell that is over two standard deviations away.
- I made the methods for this, they are in FeatureExtraction.py and are titled removeOutliers() and removeFromImg(). However utilizing these methods would have required me to redo the training data step which would have been time consuming.