APPENDIX

SEGMENTATION

The first step to extract the object of interest from our images. This is crucially important and challenging since the ALL-IDB2 and C-NMC datasets were collected using different methods. Furthermore, the ALL-IDB2 data includes the background surrounding the WBC. Conversely, the C-NMC images have been segmented by an expert oncologist. Examples of these images are found in Figures 2 and 3.

Our image segmentation algorithm was partially inspired by Ghane et al. in the use of the CMYK color space, histogram equalization, and contrast stretching [20]. For a given image, $\mathbf{a}[\vec{x}]$, we first extract the red, green, and blue channels, respectively, by

$$\mathbf{r}[\vec{x}] = \left\{ \begin{array}{c} 1 \\ \emptyset \\ \emptyset \end{array} \right\} \mathbf{a}[\vec{x}], \tag{4}$$

$$\mathbf{g}[\vec{x}] = \left\{ \begin{array}{c} \emptyset \\ 1 \\ \emptyset \end{array} \right\} \mathbf{a}[\vec{x}], \tag{5}$$

$$\mathbf{b}[\vec{x}] = \left\{ \begin{array}{c} \emptyset \\ \emptyset \\ 1 \end{array} \right\} \mathbf{a}[\vec{x}]. \tag{6}$$

We need to then convert all of the black pixels to yellow. This will allow us to extract our object from the image easier in the CMYK color space. Let v and h be the width and height of image $\mathbf{a}[\vec{x}]$. Then $\forall i \in \{1, ..., v\}$ and $\forall j \in \{1, ..., h\}$, we convert the black pixels to yellow by

$$\mathbf{r}^*_{i,j}[\vec{x}] = \left\{ \begin{array}{ll} 255, & \mathbf{r}_{i,j}[\vec{x}] \stackrel{?}{=} \mathbf{g}_{i,j}[\vec{x}] \stackrel{?}{=} \mathbf{b}_{i,j}[\vec{x}] \stackrel{?}{=} 0 \\ \mathbf{r}_{i,j}[\vec{x}], & \text{else} \end{array} \right., \tag{7}$$

$$\mathbf{g}^*_{i,j}[\vec{x}] = \left\{ \begin{array}{ll} 255, & \mathbf{r}_{i,j}[\vec{x}] \stackrel{?}{=} \mathbf{g}_{i,j}[\vec{x}] \stackrel{?}{=} \mathbf{b}_{i,j}[\vec{x}] \stackrel{?}{=} 0 \\ \mathbf{g}_{i,j}[\vec{x}], & \text{else} \end{array} \right., \tag{8}$$

$$\mathbf{b}^*_{i,j}[\vec{x}] = \left\{ \begin{array}{ll} 0, & \mathbf{r}_{i,j}[\vec{x}] \stackrel{?}{=} \mathbf{g}_{i,j}[\vec{x}] \stackrel{?}{=} \mathbf{b}_{i,j}[\vec{x}] \stackrel{?}{=} 0 \\ \mathbf{b}_{i,j}[\vec{x}], & \text{else} \end{array} \right., \tag{9}$$

$$\mathbf{d}[\vec{x}] = \left\{ \begin{array}{c} \mathbf{r}^*[\vec{x}] \\ \mathbf{g}^*[\vec{x}] \\ \mathbf{b}^*[\vec{x}] \end{array} \right\}. \tag{10}$$

Now that we converted all the black pixels to yellow, we can convert our input image to the CMYK color space and extract the Y channel by

$$\mathbf{f}[\vec{x}] = \left(\left\{ \begin{array}{c} \emptyset \\ \emptyset \\ 1 \\ \emptyset \end{array} \right\} L_{CMYK} \mathbf{d}[\vec{x}] \right) \otimes \mathbf{g}[\vec{p}] \tag{11}$$

where L_{CMYK} converts the input image to the CMYK color space, $\left\{ \begin{array}{c} \emptyset \\ \emptyset \\ 1 \\ \emptyset \end{array} \right\}$ extracts the Y channel, $\otimes$ is the convolution operator, and $\mathbf{g}[\vec{p}]$ is the local maximum filter using a 5×5 kernel. This process converts the image into a more useful color space, where we use the helpful channel for extracting our object of interest. We also apply a local max operation to help remove some noise from the images.

We then perform histogram equalization and contrast stretching to obtain two new images. We obtain the contrast stretched image by

$$\mathbf{s}[\vec{x}] = (\mathbf{f}[\vec{x}] - \epsilon)(\frac{255}{\delta - \epsilon}), \tag{12}$$

where δ and ϵ are the 2^{nd} and 98^{th} percentile values of image $\mathbf{f}[\vec{x}]$. Next, to obtain the image with histogram equalization, we performed

$$\mathbf{e}[\vec{x}] = \mathbb{E}\mathbf{f}[\vec{p}], \tag{13}$$

where $\mathbb{E}$ is the histogram equalization operator. We then combined the two images as performed by Ghane et al. [20]. Explicitly, we combined the images by

$$\mathbf{j}[\vec{x}] = (2 \times \mathbf{s}[\vec{x}]) + \mathbf{e}[\vec{x}]. \tag{14}$$

We then performed a simple thresholding and extracted the object closest to the center of the image. This was performed by

$$\mathbf{u}[\vec{x}] = \mathcal{I}_{(v/2,h/2)} \Gamma_{(\bigwedge \mathbf{j}[\vec{x}]+0.01)<} \mathbf{j}[\vec{x}], \tag{15}$$

where Γ is the threshold operator and retains all the pixel values that are less than the the minimum value of image $\mathbf{j}[\vec{x}]$ plus 0.01, $\bigwedge$ is the minimum value operator, v and h are the first and second values from $Z\mathbf{j}[\vec{x}]$, respectively. The size operator is Z. The resulting image $\mathbf{u}[\vec{x}]$ is a binary image where the object of interest is white (or 1's) and the background is black (or 0's). This provides us with our extracted shapes as observed in Figures 2 and 3.

Shape

We collected the encircled image-histograms (EIs) using the algorithm described by Lamberti [16]. This algorithm results in a vector $\vec{c}_{EI}$ which contains the white and black pixel counts from image $\mathbf{u}[\vec{x}]$. These counts are the first two metrics, $\vec{m}_1$ and $\vec{m}_2$, respectively. The shape proportion (SP) value for a given image, i, is merely

$$\vec{m}_{3,i} = \frac{\vec{m}_{1,i}}{\vec{m}_{1,i} + \vec{m}_{2,i}}. \tag{16}$$

The SP value is essentially the proportion of white pixels after applying the SPEI algorithm [16]. This SPEI algorithm puts a shape in its minimum encompassing circle. Then the circle is placed in its minimum encompassing square. Figure 4 provides an illustrative example of the SPEI algorithm result. Lamberti, et al. showed SPs to have unique values for regular polygons

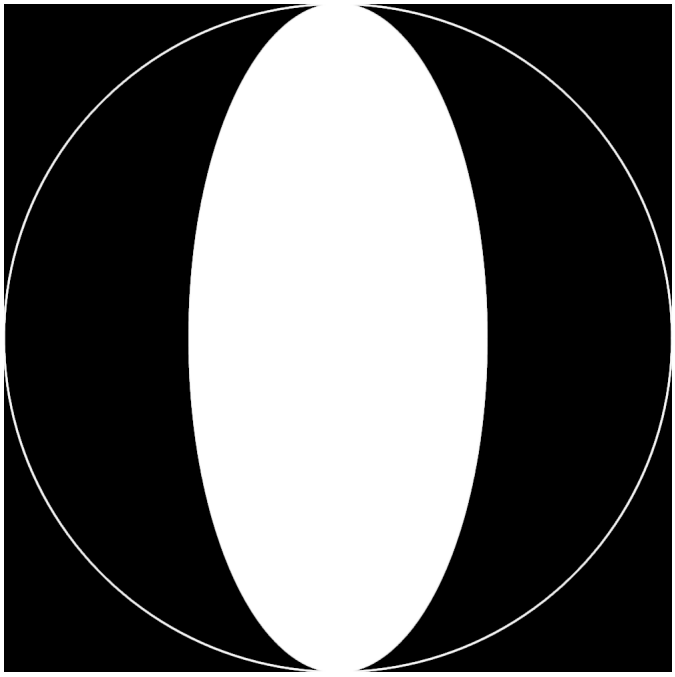

Fig. 4: Figure provides an example of the SPEI algorithm result with a reference circle. The SPEI algorithm will put the oval in the minimum encompassing circle. Then SPEI will put the region in the minimum encompassing square. The reference circle is only for presentation purposes. It is not part of the actual final image.

and circles [16]. The EIs are the white and black pixel counts after applying SPEIs [16].

We calculated eccentricity by finding the ratio of the first and second eigenvalues. To obtain the eigenvalues, we performed

$$\vec{e}_i = \mathbb{E}_{(1,2)} V \mathbf{u}_i[\vec{x}], \tag{17}$$

where V collects the covariance matrix of the shape matrix and $\mathbb{E}_{(1,2)}$ calculates the first and second eigenvalues of the resulting covariance matrix. The j^{th} eigenvalue on image i is $\vec{e}_{i,(j)}$. To obtain $\vec{m}_6$, eccentricity, we perform on a given image, i,

$$\vec{m}_{6,i} = \frac{\vec{e}_{i,(1)}}{\vec{e}_{i,(2)}}. \tag{18}$$

The first and second eigenvalues are metrics $\vec{m}_4$ and $\vec{m}_5$, respectively. It is well-known that the eigenvalues of a covariance matrix correspond to the linear combination in the data which maximizes the variance for their respective dimension [34]. For instance, the first eigenvalue is the linear combination of the data which maximizes the first eigenvalue [34]. We also know that the linear projections, or eigenvalues, are orthogonal to one another [34]. Thus, the eigenvalues are measures of the major and minor axes of our given shape. Using the ratio of the major and minor axes provides some insight to how a given shape exists as a 2D digital image [17]. A value of 1 corresponds to a shape with the same major and minor axes' lengths. A value greater than 1 corresponds to the case where the major axis' length is larger than the minor axis' length. The limit of eccentricity would correspond to the case where the major axis' length is infinitely larger than the minor axis' length.

Circularity, $\vec{m}_7$, was collected on image i by

$$\vec{m}_{7,i} = \frac{\sum \mathbf{u}_i[\vec{x}]}{4\pi \times \left(\sum \triangleleft \mathbf{u}_i[\vec{x}] - \sum \mathbf{u}_i[\vec{x}]\right)}, \tag{19}$$

where $\sum$ sums the pixel intensity values. The $\sum$ operator will compute the area of the image since we are restricted to binary images. The denominator of this metric is the perimeter of the binary image multiplied by 4π. Circularity provides a measure for how circular a shape is as a 2D digital image [17]–[19]. A value of 1 corresponds to a perfect circle [17]–[19].

The number of corners, $\vec{m}_8$, counts the number of corners on an object [35]. This metric was measures on image i by

$$\mathbf{v}_i[\vec{x}] = G_2 E_x \mathbf{u}_i[\vec{x}], \tag{20}$$

$$\mathbf{h}_i[\vec{x}] = G_2 E_y \mathbf{u}_i[\vec{x}], \tag{21}$$

$$\mathbf{A}_i[[\vec{x}]] = \begin{bmatrix} \mathbf{v}_i[\vec{x}]^2 & \mathbf{v}_i[\vec{x}]\mathbf{h}_i[\vec{x}] \\ \mathbf{v}_i[\vec{x}]\mathbf{h}_i[\vec{x}] & \mathbf{h}_i[\vec{x}]^2 \end{bmatrix}, \tag{22}$$

$$\mathbf{r}_i[\vec{x}] = \frac{det(\mathbf{A}_i[[\vec{x}]]) - k}{trace(\mathbf{A}_i[[\vec{x}]])^2}, \tag{23}$$

$$\vec{m}_{8,i} = \mathbb{C}\mathbf{r}_i[\vec{x}], \tag{24}$$

where E is the edge operator where the subscript finds the vertical or horizontal edges, x and y, respectively, G_2 is a Gaussian smoothing operator with a standard deviation of 2, $\mathbf{A}_i[[\vec{x}]]$ is an image tensor, $\mathbf{r}_i[\vec{x}]$ are the resulting corners extracted from the image, and $\mathbb{C}$ extracts the number of corners in the input image. Here we first extract the edges from the image to help first isolate the parts of the image that would contain the corners. Next, we smooth the image to remove spurious parts of the image that would not qualify as a corner. We then create a tensor and compute image $\mathbf{r}_i[\vec{x}]$ to create an image that contains all of the corners in the image. We then count the number of corners as the last operator to retrieve the desired metric.

Note that all of the variables used in our analysis have a very interpretable meaning. The EI's are the white and black pixel counts after putting the shape in the minimal encompassing circle and then the minimal encompassing square. The SP value is the proportion of white pixels divided by the summation of the EIs. The 1st and 2nd Eigenvalues are associated with the major and minor axes, respectively, of the shape. Eccentricity is the ratio of the major to minor axis. Circularity measures how circular a shape is where a value of 1 indicates a perfect circle. The number of corners estimates the number of corners the shape has.

Color

The features for color were the mean and standard deviation (SD) of the object's pixel intensity values. The object's pixels were extracted by using the segmented shape. Seven different color channels were used from two different color representations: RGB and CMYK. Explicitly, for a given color channel and image, metrics $\vec{m}_{9,i}$ to $\vec{m}_{22,i}$

$$\vec{m}_{9,i} = \mathcal{M}_{\mathbf{u}_i[\vec{x}]}\mathbf{r}_i[\vec{x}], \tag{25}$$

$$\vec{m}_{10,i} = \mathcal{T}_{\mathbf{u}_i[\vec{x}]}\mathbf{r}_i[\vec{x}], \tag{26}$$

$$\vec{m}_{11,i} = \mathcal{M}_{\mathbf{u}_i[\vec{x}]}\mathbf{g}_i[\vec{x}], \tag{27}$$

$$\vec{m}_{12,i} = \mathcal{T}_{\mathbf{u}_i[\vec{x}]}\mathbf{g}_i[\vec{x}], \tag{28}$$

$$\vec{m}_{13,i} = \mathcal{M}_{\mathbf{u}_i[\vec{x}]}\mathbf{b}_i[\vec{x}], \tag{29}$$

$$\vec{m}_{14,i} = \mathcal{T}_{\mathbf{u}_i[\vec{x}]}\mathbf{b}_i[\vec{x}], \tag{30}$$

$$\vec{m}_{15,i} = \mathcal{M}_{\mathbf{u}_i[\vec{x}]}(\begin{Bmatrix} 1 \\ \emptyset \\ \emptyset \\ \emptyset \end{Bmatrix} L_{CMYK}\mathbf{d}[\vec{x}]), \tag{31}$$

$$\vec{m}_{16,i} = \mathcal{T}_{\mathbf{u}_i[\vec{x}]}(\begin{Bmatrix} 1 \\ \emptyset \\ \emptyset \\ \emptyset \end{Bmatrix} L_{CMYK}\mathbf{d}[\vec{x}]), \tag{32}$$

$$\vec{m}_{17,i} = \mathcal{M}_{\mathbf{u}_i[\vec{x}]}(\begin{Bmatrix} \emptyset \\ 1 \\ \emptyset \\ \emptyset \end{Bmatrix} L_{CMYK}\mathbf{d}[\vec{x}]), \tag{33}$$

$$\vec{m}_{18,i} = \mathcal{T}_{\mathbf{u}_i[\vec{x}]}(\begin{Bmatrix} \emptyset \\ 1 \\ \emptyset \\ \emptyset \end{Bmatrix} L_{CMYK}\mathbf{d}[\vec{x}]), \tag{34}$$

$$\vec{m}_{19,i} = \mathcal{M}_{\mathbf{u}_i[\vec{x}]}(\begin{Bmatrix} \emptyset \\ \emptyset \\ 1 \\ \emptyset \end{Bmatrix} L_{CMYK}\mathbf{d}[\vec{x}]), \tag{35}$$

$$\vec{m}_{20,i} = \mathcal{T}_{\mathbf{u}_i[\vec{x}]}(\begin{Bmatrix} \emptyset \\ \emptyset \\ 1 \\ \emptyset \end{Bmatrix} L_{CMYK}\mathbf{d}[\vec{x}]), \tag{36}$$

$$\vec{m}_{21,i} = \mathcal{M}_{\mathbf{u}_i[\vec{x}]}(\begin{Bmatrix} \emptyset \\ \emptyset \\ \emptyset \\ 1 \end{Bmatrix} L_{CMYK}\mathbf{d}[\vec{x}]), \tag{37}$$

$$\vec{m}_{22,i} = \mathcal{T}_{\mathbf{u}_i[\vec{x}]}(\begin{Bmatrix} \emptyset \\ \emptyset \\ \emptyset \\ 1 \end{Bmatrix} L_{CMYK}\mathbf{d}[\vec{x}]), \tag{38}$$

where $\mathcal{M}$ and $\mathcal{T}$ are the mean and standard deviation operators, respectively. Both operators are only extracted on the 1's of image $\mathbf{u}[\vec{x}]$, which ensures that only the colors of object are considered for a given color channel. The mean value for a given channel indicates how much of that color is in the respective channel. Large values indicate that a particular color is very present in the image, while small correspond to the image having very little of a given color. The SD indicates how much the presence of a color changes throughout an image. A large SD indicates that the presence of the intensity of a color changes dramatically, while small a SD corresponds to a fairly uniform representation of the intensity of a color. Thus, our color features have an interpretable and explainable meaning.

Texture

The features for texture where the mean and SD of the co-occurance matrix of the masked grayscale image. This was claculated by:

$$\vec{m}_{23,i} = \mathcal{M}\mathfrak{C}_{1,256}((L_L\mathbf{a}_i[\vec{x}]) \times \mathbf{u}_i[\vec{x}]), \tag{39}$$

$$\vec{m}_{24,i} = \mathcal{T}\mathfrak{C}_{1,256}((L_L\mathbf{a}_i[\vec{x}]) \times \mathbf{u}_i[\vec{x}]), \tag{40}$$

where L_L converts the given input image to a grayscale image and $\mathfrak{C}$ calculates the co-occurrence matrix with a shift of 1 to the right and bottom for 256 intensities. We multiply the grayscaled image by the extracted shape to remove the background from the image. We then calculate the co-occurrence matrix. This image provides the counts of number of times a certain pixel intensity is next to another intensity with a single shift. This 256×256 matrix is useful for texture measures [17]. We then extracted the mean and SD from this matrix. The mean of this matrix indicates how often an average intensity is nearby another value. Large means would indicate that intensities are nearby other intensity values often which translates to a very rough image. Small means would indicate that intensities are nearby very few other values with translates to a smoother image. The SD indicates how often much the intensities change over an image, with large values indicating a large amount of change and small values indicating a small amount of change. Thus, our texture features have an interpretable and explainable meaning.