CENTRO UNIVERSITÁRIO FEI

NAYARA VIANA GEROLOMO
DANIEL LUIS COSTA

CONTAGEM DIFERENCIAL AUTOMÁTICA DE LEUCÓCITOS EM IMAGENS MICROSCÓPICAS

São Bernardo do Campo
2017

NAYARA VIANA GEROLOMO
DANIEL LUIS COSTA

CONTAGEM DIFERENCIAL AUTOMÁTICA DE LEUCÓCITOS EM IMAGENS MICROSCÓPICAS

Trabalho de Conclusão de Curso apresentado ao Centro Universitário FEI, como parte dos requisitos necessários para obtenção do título de Bacharel em Ciência da Computação. Orientado pelo Prof. Dr. Paulo Sérgio Silva Rodrigues.

São Bernardo do Campo
2017

RESUMO

A contagem diferencial de leucócitos é importante para o diagnóstico de várias doenças. Quando feito manualmente, esse processo é lento e apresenta variância intra e inter-pessoal de resultados, além de requerer um especialista. Este trabalho trata de um método automático de contagem diferencial de neutrófilos e linfócitos a partir de imagens microscópicas. O método proposto é dividido em três etapas principais: pré-processamento, segmentação e reconhecimento dos leucócitos. Enquanto a segmentação é baseada em limiarização, a classificação dos tipos de leucócitos, contida na etapa de reconhecimento, é feita por máquinas de vetor de suporte (SVMs, *support vector machines*). O método foi concebido, parametrizado e validado com base nos resultados de experimentos conduzidos sobre um banco de imagens proveniente de lavagens bronco-alveolares feitas em ratos de laboratório. Além dos passos que compõem o método apresentado, foram conduzidos experimentos envolvendo a utilização de limiarização por minimização da divergência *fuzzy* (LMDF) e operadores morfológicos de abertura.

Palavras-chave: Imagens microscópicas. Leucócitos. Contagem diferencial automática. Processamento de imagens. Limiarização. Divergência fuzzy. Operações Morfológicas. Máquina de vetores de suporte.

LISTA DE TABELAS

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS

ANA	*antinuclear antibody*.
ANN	*artificial neural network*.
BF *matching*	*brute force matching*.
BLOB	*Binary Large Object*.
BoW	*bag-of-visual-words*.
CAGA	*chain-like agent genetic algorithm*.
CLBP	*complete local binary pattern*.
CPM	*cell pyramid matching*.
CSS	*curvature scale space*.
DCT	*discrete cosine transform*.
DoG	*difference of gaussians*.
DSSM	*double-strategy splitting model*.
FBP	*feedfoward back-propagation*.
FCM	*fuzzy c-means clustering*.
FDP	função de distribuição de probabilidade.
FN	falso negativo.
FP	falso positivo.
GIMP	*GNU image manipulation program*.
GMM	*gaussian mixture model*.
GVF	*gradient vector flow*.
HEp-2	*human epithelial type 2*.
HNN	*Hopfiled neural network*.
HSV	*hue-saturation-value*.
IIF	imunofluorescência indireta.
JPEG	*Joint Photographic Experts Group*.
KBT	*Kleihauer-Betke test*.
LMDF	limiarização por minimização da divergência *fuzzy*.
MK	*moving k-means clustering*.
MKL	*multiple kernel learning*.
NPM	neutrófilo polimorfonuclear.
PCNN	*pulse-coupled neural network*.

RBC *red blood cell*.

RGB *red green blue*.

ROC *reciever operating characteristic*.

ROI *region of interest*.

SID *scale-invariant descriptor*.

SIFT *scale invariant feature transform*.

SRGAE *seeded region growing area extraction*.

SVM *support vector machine*.

SWLDA *step-wise linear discriminant analysis*.

TM *template matching*.

UNIFESP Universidade Federal de São Paulo.

VP verdadeiro positivo.

WBC *white blood cell*.

WDMR *wavelet decomposition and multi-scale region-growing*.

SUMÁRIO

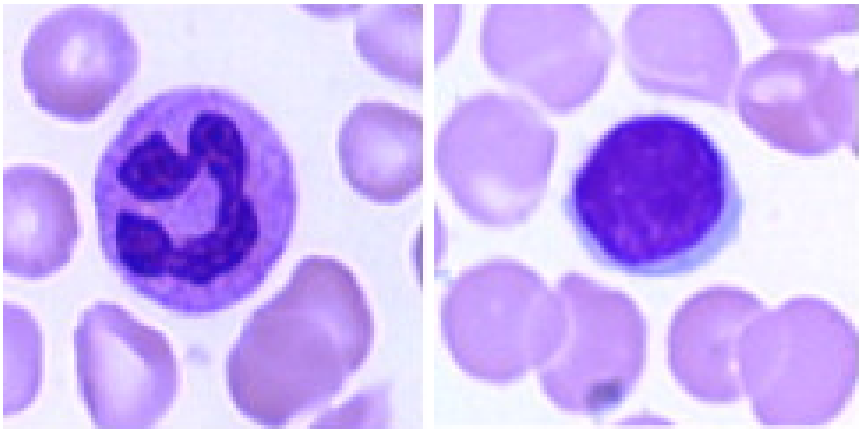
1 INTRODUÇÃO

Células sanguíneas humanas são principalmente divididas em três categorias: células vermelhas (RBCs, *red blood cells*), células brancas (WBCs, *white blood cells*) e plaquetas. As WBCs, também chamadas de leucócitos, por sua vez, fazem parte do sistema imunológico do organismo.

A contagem de WBCs no sangue de um indivíduo é importante para o diagnóstico de uma série de doenças. Um paciente soropositivo, por exemplo, tende a apresentar uma quantidade menor de WBCs no sangue (III; ZHOU, 2003). Em contrapartida, anemias e estresse, por exemplo, são condições que aumentam a contagem de WBCs no sangue (LABRY et al., 1990).

WBCs migram para tecidos durante processos inflamatórios como resposta a infecções. Com isso, a contagem de WBCs em seções de tecido é igualmente importante para o reconhecimento de uma série de condições. Adicionalmente, reconhecer o tipo de WBCs é necessário para determinar a natureza da infecção.

Existem 5 tipos de WBCs (GHOSH; BHATTACHARJEE; NASIPURI, 2016): neutrófilos, linfócitos, eosinófilos, basófilos e monócitos. Os neutrófilos (Figura 1a) são os principais responsáveis pela defesa primária contra bactérias e fungos (GHOSH; BHATTACHARJEE; NASIPURI, 2016). O aumento no número de neutrófilos, a neutrofilia, geralmente indica uma infecção bacteriana. Os linfócitos (Figura 1b), por sua vez, são responsáveis por reconhecer antígenos e fabricar os anticorpos apropriados com base em sua capacidade de memória imunológica. Uma alta concentração de linfócitos normalmente é indicativo de uma infecção viral ou, em menos casos, bacteriana (GHOSH; BHATTACHARJEE; NASIPURI, 2016).



(a) neutrófilo (b) linfócito

Figura 1 – WBCs contadas neste trabalho.

Ao processo de contar WBCs, fazendo a distinção do tipo de WBC, é dado o nome de contagem diferencial. A contagem diferencial manual (i.e. contar as células uma a uma em um microscópio) requer o trabalho exaustivo de um especialista. Além de demorado, esse processo apresenta alta intra e intervariância de resultados (SARASWAT; ARYA, 2014), fato pelo qual não é muito confiável para pesquisas, por exemplo. Assim, faz-se necessária a concepção de métodos rápidos, baratos e robustos de contagem diferencial. O interesse deste trabalho está na automatização da contagem diferencial de neutrófilos e linfócitos, que pode atingir esses objetivos.

1.1 OBJETIVO

O objetivo deste trabalho foi o de conceber e validar uma metodologia de contagem diferencial automática de neutrófilos e linfócitos, supondo que os outros três tipos de leucócitos não estejam presentes.

2 TRABALHOS RELACIONADOS

Em Ghosh, Bhattacharjee e Nasipuri (2016) propõe-se desenvolver um sistema automático de contagem diferencial das células brancas (WBCs, *white blood cells*) existentes na imagem microscópica de uma amostra de sangue utilizando lógica Fuzzy e *Random forest*. O procedimento consiste em cinco etapas: realce da região das células; segmentação das células e das regiões de *background*; restauração da borda das células; separação das regiões de interesse levando em consideração formato, tamanho, cor e textura, utilizando técnicas de lógica *fuzzy* e não-*fuzzy*; e, por fim, uma decisão final é tomada baseada na saída dos classificadores. Os resultados alcançados foram de 95,8% de precisão, 95,2% de sensibilidade e taxa de erro de 7.5%, contra 93,4%, 91,1% e 12,9% respectivamente do método manual de contagem.

No trabalho de Guo e Yu (2013) é apresentado um sistema para contagem total de células. O método proposto utiliza a técnica de histograma *dual-threshold* para a separação das células do fundo da imagem, aplicação de *floodfill* nas células e as detecta por extração de *blobs*, seguido do uso do método k-*means* para segmentar as células aderentes. O sistema apresentou rapidez, eficiência e adaptabilidade para diversos tipos de células (morfologia, tipo ou tamanho diferentes), tendo taxa de erro entre 0 e 1,33% em relação à contagem manual. Entretanto, não é possível utilizá-lo para distinguir diferentes tipos de células em uma mesma imagem.

Visando resolver a dificuldade em segmentar células sobrepostas em imagens microscópicas, Wang et al. (2016) sugere uma técnica utilizando transformada de *bottom-hat* para melhorar a qualidade da imagem; decomposição *wavelet* e crescimento de regiões multi escala (WDMR, *wavelet decomposition and multi-scale region-growing*) para localizar com exatidão as regiões de interesse (ROIs, *regions of interest*); combinação entre morfologia matemática adaptativa (utilizando um modelo separável de dupla estratégia (DSSM, *double-strategy splitting model*)) e o uso do espaço de escala de curvatura (CSS, *curvature scale space*) para dividir células sobrepostas; e extração de informações sobre formatos e texturas com base em espaços de cores para classificação dos núcleos celulares. As características ótimas são obtidas com o uso de uma máquina de vetor de suporte (SVM, *support vector machine*) em conjunto com o *chain-like agent genetic algorithm* (CAGA). A sensibilidade média da segmentação foi de 91,53% ($\pm 4{,}05\%$) e a especificidade foi de 91,64% ($\pm 4{,}07\%$). O desempenho de classificação de imagens das células comparadas pode atingir 96,19% ($\pm 0{,}31\%$) para precisão, 99,05% ($\pm 0{,}27\%$) para sensibilidade e 93,33% ($\pm 0{,}81\%$) para especificidade.

O novo método de limiarização em histogramas unimodais de Melo et al. (2015) produziu melhores resultados que as técnicas tradicionais limiarização de Otsu e de Rosin, com as quais foi comparado na segmentação de células somáticas de leite bovino. No método, a imagem de entrada utilizando o sistema de cores azul-verde-vermelho (RGB, *red green blue*) é, primeiramente, convertida para o espaço de cor Lab. Então, prossegue-se com a utilização do algoritmo k-*means* para descartar o fundo da imagem. Com a imagem resultante convertida para escala de cinzas, aplica-se o limiar calculado para gerar sua binarização. A segmentação das células é obtida pela transformada de *watershed*. Por fim, as células são classificadas pelo tamanho e então contadas. Enquanto a utilização dos algoritmos clássicos Otsu e Rosin obteve precisão de contagem de 0% e 78,9% em relação a contagem do especialista, respectivamente, a estratégia apresentada atingiu 99,7%, o que sugere sua eficácia.

No trabalho de Adagale e Pawar (2013) é proposta uma forma de segmentar e contar células sanguíneas isoladas e sobrepostas que utiliza algoritmos de redes neurais artificiais (ANNs, *artificial neural networks*). Visando obter qualidade nos resultados, foi apresentada a combinação do modelo rede neural de pulso acoplado (PCNN, *pulse-coupled neural network*) com comparação de *template* (TM, *template matching*). Inicialmente, as imagens das células sanguíneas são extraídas com um microscópio com escala de medida em 100x. Aplica-se então uma conversão de cor de RGB para níveis de cinza, segmenta-se a imagem utilizando PCNN e eliminam-se objetos indesejáveis com a utilização de um filtro de mediana. O TM é aplicado para segmentar as células sobrepostas. A partir desse passo, são contadas as células isoladas e as sobrepostas. Os resultados demonstraram que, para imagens com 75 e 52 células, o método proposto apresentou acuracidade de 98% e 93%, respectivamente. O trabalho também apresenta resultados de contagem de células utilizando o PCNN mas não TM, os quais são inferiores.

Em razão da dificuldade em separar células do *background* em imagens microscópicas, Mao-jun et al. (2008) recomenda a utilização de PCNN. A utilização desse método, quando combinado com um filtro de mediana, permitiu a remoção de ruídos, segmentação das células e, fazendo uso da sua propriedade de propagação de ondas automáticas, eliminação dos objetos verdadeiros-negativos restantes na imagem. Isso viabiliza a contagem total das células e a segmentação específica de uma célula em relação à sua vizinhança. A utilização de PCNN trouxe resultados promissores na remoção de ruído e segmentação quando aplicados a células de atributos similares (tais como tamanho e formato), porém, não manteve a mesma qualidade em relação à contagem, uma vez que não conseguiu identificar células adjacentes ou sobrepostas. Apresentou imprecisão, também, quando aplicado a células de estrutura mais complexa, como as de vegetais.

Com o objetivo de segmentar grupos de células de imagens de amostras de tecido com propósito de contagem, Kothari, Chaudry e Wang (2009) desenvolveu uma metodologia que lida explicitamente com aglomerados de células. Nessa metodologia, computa-se o valor médio do tamanho da célula analisada, o que permite a distinção entre células isoladas e aglomerados de células. Então, analisa-se a distância entre duas concavidades; se elas estiverem mais próximas do que um limiar calculado sobre o tamanho médio das células, esse agrupamento é quebrado em dois. Esse passo é repetido até que não haja mais particionamentos a serem feitos. As aglomerações restantes são segmentadas utilizando conexão de centroide e, por fim, o método de encaixe de elipse é aplicado. A abordagem atingiu taxa de acerto de 95,02% e superou outras 3 abordagens comparadas também baseadas em concavidade.

Para contar células coradas por imunocitoquímica em imagens microscópicas, Ramin et al. (2012) utiliza algoritmos genéticos. Na etapa de pré-processamento, o ruído e o fundo da imagem são removidos utilizando suavização por filtragem espacial e operadores morfológicos. Fazendo o uso do método de classificação de vizinhos mais próximo com métrica de distância euclidiana sobre a imagem no espaço de cor Lab, os núcleos são separados dos antígenos. Por fim, aplica-se o limiar de segmentação calculado por algoritmos genéticos, promovendo a contagem das células. A taxa de erro e o seu desvio padrão obtidos com o uso do método, usando a contagem manual como padrão-ouro, foram de 6,75% e 6,39%, respectivamente.

Com o intuito de diferenciar e contar WBCs e células vermelhas (RBCs, *red blood cells*) em imagens de sedimento urinário, Tangsuksant et al. (2013) indicou o uso de *feedfoward back-propagation* (FBP) de ANNs para a segmentação das imagens, a qual é feita no sistema de cores *hue-saturation-value* (HSV) e particiona a imagem em células e *background*. Para eliminação do ruído, utilizou operações morfológicas e transformada circular de Hough na detecção de WBCs e RBCs. A taxa de erro média para contagem dessas células foi de 8,35% e 5,28%, respectivamente.

Em Hamid et al. (2013) é apresentado um sistema automático para detecção e contagem de neutrófilos polimorfonucleares (NPMs) em imagens microscópicas de escarros. A qualidade do escarro, sobre a qual se embasa o método, é determinada utilizando faixas de reagentes com um analisador químico de urina, obtendo sensibilidade de 86,8% e especificidade de 75,9% na amostra. O modelo utilizado para realizar o trabalho de segmentação e contagem consiste nos seguintes passos: inicialmente, o fundo é subtraído da imagem pelo processo de aprimoramento, procedendo-se com a segmentação através de uma rede neural de Hopfiled (HNN, *Hopfiled neural network*) e do algoritmo de clusterização *fuzzy c-means clustering* (FCM). O passo seguinte consiste na conversão da imagem do formato RGB para o de escalas de cinza e da obtenção

de uma imagem binária através da aplicação da técnica *gray thresholding*. Neste processo, os ruídos da imagem são eliminados. A análise da imagem é executada para extrair características da NPM, utilizadas para critérios de seleção. Nessa análise, as células são agrupadas em termos de intensidade e área e o desvio padrão adquirido como resultado é utilizado para reconhecimento; os objetos que estão dentro do intervalo especificado são reconhecidos como células NPM. A partir deste momento, faz-se a contagem tanto de células isoladas quanto de células sobrepostas. O sistema proposto apresentou uma sensibilidade de 92,11% com especificidade de 81,82%, acuracidade de 87,32% e precisão 85,37%.

O sistema proposto por Ge et al. (2014), além da habitual contagem celular, diferencia RBCs fetais e maternais por testes de Kleihauer-Betke (KBTs, *Kleihauer-Betke tests*). Após a captura da imagem, ela é convertida para escalas de cinza e binarizada através de um método de limiarização dinâmico. As células são extraídas; se forem individuais, são contadas automaticamente. Caso contrário, aplica-se a transformada de distância. É feito o cálculo dos máximos regionais e determinado o número de agrupamentos através de uma validação de *clusters*. Para a separação das células individuais utiliza-se a técnica encaixe de elipse nos pontos de borda do agrupamento. Através de um treinamento supervisionado considerando características de tamanho, formato, gradiente e diferença de saturação são gerados vetores de recursos para lidar com a variação dessas características na imagem, podendo assim diferenciar as células maternais das fetais. O sistema automatizado contou mais de 60 mil células em 5 minutos, contra 2 mil em 15 minutos contadas manualmente e atingiu a precisão de 0,984($\pm$0,09), contra 0,926($\pm$0,15), demonstrando que é superior.

Os autores do trabalho Sarrafzadeh et al. (2015), na intenção de contar RBCs de forma eficaz e automatizada com baixo custo, apresentam um modelo baseado em transformada de circunferências para o reconhecimento das células contidas em amostras de sangue. O procedimento de contagem consiste primeiramente em remover WBCs e o *background*, assim obtendo a máscara de RBCs. Diferente de outros métodos abordados, neste a transformada de circunferência é aplicada na imagem em escalas de cinzas, sem ser necessária uma binarização prévia. O número mínimo e máximo de RBCs é estimado, seguido pela utilização de um limiar flexível para, de modo iterativo, encontrar as RBCs correspondentes aos valores. Para remover as RBCs conflitantes, faz-se o uso do índice de Jaccard. Para obter resultados com maior embasamento, o desempenho da transformada de circunferência no sistema é comparado com o desempenho da transformada circular de Hough. Ambas as abordagens são comparadas com a contagem feita por dois observadores. A transformada de circunferência obteve erro percentual absoluto médio de 8,36% em relação ao primeiro observador e 9,56% em relação ao segundo. Por sua

vez, a de Hough obteve 15% e 15,88%. A comparação entre os dois observadores resultou no erro médio de 2,54%, mostrando a imprecisão da contagem manual.

Em Rashid, Mashor e Hassan (2015), é proposto o uso do algoritmo de clusterização *moving k-means clustering* (MK), um classificador de aprendizado não supervisionado, para segmentação de imagens coloridas de amostras de sangue contendo doenças talassêmicas. Com o objetivo de obter a segmentação de RBCs normais e anormais, é aplicado um algoritmo de contraste global para o aprimoramento da imagem; então, é executado o algoritmo MK em cada um dos componentes HSV da imagem para separar as células do *background*; depois disso, um filtro de mediana promove uma suavização na imagem. Comparando o resultado do processamento das células vermelhas segmentadas, a melhor delas será processada utilizando o algoritmo extração de área por crescimento de regiões com sementes (SRGAE, *seeded region growing area extraction*) baseada na saturação do componente de cor para remover regiões indesejadas e obter as células vermelhas segmentadas. O método analisado fora usado em 60 imagens de sangue as quais consistem de alfa talassemia, beta talassemia, beta traço de talassemia e amostras de sangue normal. O método proposto apresenta, como resultado, acurácia de 94,57%. O presente estudo demonstra a utilidade dos componentes de cores HSI para segmentar imagem, como indispensável.

A fim de classificar seis classes de padrões de coloração de células humanas epiteliais de tipo 2 (HEp-2s, *human epithelial type 2*), Gragnaniello, Sansone e Verdoliva (2016) propõe o uso de um denso descritor local invariante tanto para a mudança de escala, quanto para as rotações. O descritor invariante à escala (SID, *scale-invariant descriptor*) é computado seguindo as etapas: transformação log-polar, que garante invariância em mudanças de escala ou rotação; suavização multiescala, para assegurar que a mudança de escala não distorça a imagem; computação de derivadas direcionais para capturar transições; e transformada discreta de Fourier em duas dimensões para reduzir o vetor de características extraído. Como técnica de classificação, optou-se pelo paradigma mochila de palavras visuais (BoW, *bag-of-visual-words*), o qual utiliza a frequência de palavras consideradas para treinar um classificador (em processamento de imagens, as características extraídas são consideradas palavras-chave) em conjunto de uma SVM. Das seis classes de células, o método possuiu melhor performance do que os comparados em quatro delas, atingindo uma precisão média de classe de 88,69% e taxa de classificação correta de 91,67%.

Ainda sobre classificação automática de células HEp-2s, em Sarrafzadeh et al. (2016) o sistema proposto busca pelas características mais relevantes para o processo de classificação. Após as imagens serem normalizadas, são extraídos seus atributos: informações de intensidade,

estatísticas, espectros, algumas características de textura obtidas por momentos invariantes, Haralick, *wavelet-based*, padrão binário local completo (CLBP, *complete local binary pattern*) e filtro Gabor. As propriedades mais significativas são selecionadas utilizando análise discriminante linear a cada passo (SWLDA, *step-wise linear discriminant analysis*), que serão utilizadas juntamente com o modelo de misturas gaussianas (GMM, *gaussian mixture model*) para classificar as células. Na busca por categorizar seis células, o sistema alcançou a média de precisão de classificação de 73,33%, e a célula com maior acerto chegou a 88,29%. Os resultados também apontaram que outras características devem ser exploradas.

O artigo de You et al. (2016) investiga o problema de individualização de células cerebrais, que costumam se aglomerar e ter bordas mal definidas. A abordagem proposta preprocessa a imagem com um filtro gaussiano para remoção de ruído e separa as células do *background* com o uso de um classificador *random forests* sobre as regiões de um *grid* regular. A separação das células consiste no uso de um filtro *min-max* para exacerbar extremos locais; os máximos absolutos são tomados como centróides e, a partir deles, é aplicado um algoritmo competitivo de crescimento de regiões que resulta nos contornos das células. O método foi testado sobre imagens provenientes da região hipocampal do cérebro de um macaco e seu desempenho foi comparado e superou o dos algoritmos *iCut* e *watershed*.

Em Lee e Chen (2014) é proposto um método para classificação de anormalidades em células vermelhas baseado em formato e textura. O método começa pela normalização do histograma para remover variações na iluminação e segmenta as células pela limiarização adaptativa de Otsu. Células ocluídas por outras são descartadas, uma vez que não disponibilizam completamente a informação de formato a ser utilizada na classificação. Reconhecidas duas células sobrepostas, qual delas está embaixo (e portanto será descartada) é determinado pela análise da concavidade da curva formada pela intersecção de seus contornos, a qual é encontrada pelo uso do detector Canny. A classificação então é feita por uma rede neural híbrida multi-camada, alimentada primeiramente por características de formato e depois por características de textura. O método foi aplicado sobre 200 imagens e classificou corretamente suas células com 91% de acurácia.

A segmentação proposta em Liao et al. (2016) inicia com uma limiarização de Otsu sobre a imagem em tons de cinza, onde regiões conexas com menos de 10 pixels são consideradas ruído e descartadas. Também é aplicado o algoritmo *region filling* para remover os buracos no centro das células. Os contornos dos componentes conexos são extraídos e aproximados por polígonos, dos quais pares de pontos são iterativamente selecionados pela detecção *bottleneck* e pelo teste de encaixe de elipses para separar cada célula do grupo de células sobrepostas. Um

procedimento melhorado de encaixe de elipses é então aplicado para determinar a segmentação final. O método foi testado sobre 80 imagens de células sanguíneas, sendo que 40 destas utilizam microscopia fluorescente. Foi comprovada acurácia de 92.2% e 90.2% para as imagens normais e fluorescentes, respectivamente, o que é significantemente maior que os demais métodos avaliados no artigo com os quais o método proposto foi comparado, também obtendo menor tempo computacional.

O trabalho apresentado em Piccinini et al. (2014) propõe a geração de um mosaico de imagens capturadas de diferentes posições da lâmina. Como comprovado pelo artigo, a contagem manual de células vivas ou mortas em um hemocitômetro é mais confiável se feita sobre o mosaico, ao invés de sobre múltiplas imagens desconexas ou sobre o próprio microscópio.

Em Tomari et al. (2014) é proposto um método de contagem de células vermelhas normais e anormais. A etapa de segmentação consiste na aplicação da limiarização adaptativa de Otsu sobre o canal verde, seguida pela aplicação de transformações morfológicas para remover buracos das células e ruído. O algoritmo descarta células sobrepostas e na borda da imagem. As demais células são então classificadas com o uso de uma ANN sobre as características geométricas de compactividade e valores invariantes de momento extraídas das hemácias. O método foi aplicado sobre as imagens de 4 diferentes amostras de sangue e foi obtida 83% de acurácia na classificação das hemácias. A rede neural foi construída sobre um conjunto de teste de 100 imagens e um conjunto de validação de 50 imagens.

O trabalho de Wiliem et al. (2014) visa classificar em uma de 6 classes definidas *a priori* células HEp-2s nas quais foi utilizado o protocolo de indirect immunofluorescence (IIF, imunofluorescência indireta), com o propósitos de avaliar testes de patologias de anticorpos anti-nucleares (ANAs, *antinuclear antibodies*). O algoritmo proposto trabalha com a imagem de uma única célula e sua máscara como entrada. O método de classificação usa o framework aprendizado de múltiplos *kerneis* (MKL, *multiple kernel learning*). A extração de características é feita por BoW; contudo, a imagem é dividida em regiões e a característica extraída de cada região é uma média dos histogramas de cada *patch* (pequeno retângulo) contido nessa região. Os histogramas de cada *patch* são uma medida de proximidade dele para cada uma das entradas previamente aprendidas contidas no dicionário de palavras visuais. A distância para cada termo visual, bem como a maneira como o valor para cada termo é fundido em um único histograma é o *kernel* do método e varia sobre o *framework* MKL. A abordagem de comparação por células de pirâmide longplsp (CPM, *cell pyramid matching*) proposta no artigo assume a existência de três regiões para cada célula, em dois diferentes níveis: para o primeiro, uma única região engloba toda a célula, enquanto no segundo nível a célula é dividida em uma região externa e

uma interna. O método foi avaliado sobre dois bancos de imagens: ICPRContest e SNPHEp-2. Os *kerneis* avaliados envolveram combinações de: transformada de cosseno discreto (DCT, *discrete cosine transform*) e transformada de característica invariante à escala (SIFT, *scale invariant feature transform*) para extração de características; atribuição suave, codificação esparsa e quantização de vetor para cálculo do histograma; comparação piramidal espacial, região dual e CPM (proposto pelo artigo) para estruturar as regiões. As combinações de melhor performance tiveram suas possíveis fusões sob o *framework* MKL avaliadas e o melhor resultado se deu para (DCT-atribuição suave CPM) + (DCT-quantização de vetor CPM), com 74.9% de performance média. Usando esses dois *kernels*, o método foi comparado com outros três da literatura e os superou.

O método apresentado em Faridi et al. (2016) é capaz de segmentar células de núcleo deformado em imagens de câncer de mama enquanto ignora células de núcleo saudável. Para remoção de ruído é utilizado filtro bilateral. A segmentação inicia com a correção do *gamma* e limiarização sobre ele. Um fechamento é então aplicado para preencher os buracos internos às regiões reconhecidas e eliminar as muito pequenas. É aplicado então um filtro diferença de gassianas (DoG, *difference of gaussians*) seguido de uma limiarização; as regiões do resultado são consideradas os núcleos das células a serem segmentadas. É dado início então, para cada núcleo, a um procedimento de segmentação da borda das células; o algoritmo *level set* é utilizado e inicia com uma borda gerada pela dilatação do núcleo, que encolhe iterativamente com base em mudanças na iluminação da imagem. Os experimentos resultaram em 86% de acurácia na detecção dos núcleos e 87% para a segmentação da borda.

O trabalho de Krishnan et al. (2012) visa discriminar uma mucosa oral fibrosa de uma mucosa normal com base na morfologia e textura dos núcleos das células basais, características que variam durante processos de transformação malignos. O método proposto pré-processa a imagem com a aplicação de filtros de mediana e de difusão anisotrópica. A camada basal (que contém as células a serem avaliadas) é extraída com a utilização de três procedimentos: uma etapa inicial de segmentação por *fuzzy divergence* separa o epitélio do *background*; é aplicado um fechamento para remoção de pequenas regiões no epitélio, sendo este selecionado por rotulamento de componentes conexos e descartadas as demais regiões do *foreground*. Então, o detector Canny é utilizado para encontrar a borda entre o epitélio e o *background*, sobre a qual é encaixada uma parábola. É definida uma segunda parábola, paralela à primeira mas deslocada para cima. A sobreposição das regiões compreendidas entre as duas parábolas e entre a segunda parábola e a borda detectada é a camada basal extraída. Para segmentar as células na camada basal é utilizado um método de *color deconvolution*, que exacerba o contraste. A segmentação

por divergência *fuzzy* seguida de operações morfológicas é novamente aplicada para detectar os núcleos das células, os quais servem como marcadores para o algoritmo *watershed*. A camada basal é particionada em compartimentos que podem conter uma ou uma aglomeração de células basais ou uma célula não basal; as partições são individualmente avaliadas para determinar qual é o caso de cada uma delas e a segmentação é corrigida de acordo. Uma vez divididas as células, o real contorno de cada uma delas é encontrado com o uso de *snakes* com fluxo de vetor de gradiente (GVF, *gradient vector flow*). O artigo considera 23 diferentes características a serem extraídas das células para classificação, mas descarta 5 delas por meio de uma avaliação feita com o uso de um classificador não-supervisionado. Por fim, é utilizado um classificador (classificadores supervisionados utilizam validação cruzada *k-fold*). O artigo considera o uso de SVM, redes bayesianas e GMM. Os três classificadores obtiveram, respectivamente, acurácia de 99.66%, 96.56% e 90.37%.

Para segmentar células humanas HT29 de câncer de cólon, Gharipour e Liew (2014) propõe um método para segmentação que consiste em uma combinação do convencional *fuzzy clustering* com a extração de informações espaciais locais e globais. Além do proposto, 9 outros métodos da literatura foram avaliados e superados em precisão.

O método proposto em Tareef et al. (2017) visa segmentar células cervicais, as quais tem formatos variados e comumente são sobrepostas, em alto grau. O método começa com uma classificação a nível de *superpixel* alcançado por uma oversegmentação com o algoritmo *simple linear iterative clustering*. O classificador utilizado é uma SVM que se baseia em características de formato, textura e existência de bordas para classificar cada *superpixel* como um núcleo, parte do citoplasma de uma célula ou pertencente ao *background*. A segmentação de cada célula começa com uma inicialização dos núcleos baseada em bordas, procede com uma deformação de formato baseada em codificação esparsa para englobar o citoplasma e refina o contorno da célula com o uso do modelo de evolução *level set* por distância regularizada.

Doenças como leucemia são diagnosticadas através da contagem completa de células sanguíneas. Em Reyes, Rozo e Morales (2015) foi feito um *review* sobre a segmentação de leucócitos, já que suas propriedades morfológicas e citoquímicas são de grande relevância para diagnose do desenvolvimento da doença. Pensando em velocidade e simplicidade, o processamento das imagens em tons de cinza apontou ser a melhor abordagem, onde foram utilizados limiarização multimodal e lógica *fuzzy*. Os contornos ativos implicam no uso maior da potência computacional e consideram a necessidade de heurísticas bem-estabelecidas para acelerar o desenvolvimento de um contorno adequado para realizar a segmentação.

Visando resolver o desafio de extrair valores numéricos de leucócitos em imagens microscópicas de amostras de sangue ou tecido, na perspectiva de identificar doenças, seus progressos e desenvolvimento de remédios, em Saraswat e Arya (2014) discute-se métodos recentemente desenvolvidos para identificação de leucócitos. Na etapa de pré-processamento de imagens os problemas relacionados às variações de cores, variações de iluminação e presença de artefatos e/ou ruído nas imagens microscópicas coradas, são melhores resolvidos quando métodos baseados em deconvoluções são aplicados. Na segmentação, os métodos melhor qualificados foram *multispectral imaging* e *Suport Vector Machine Clustering* (SVMC) que obtiveram 94% de precisão.

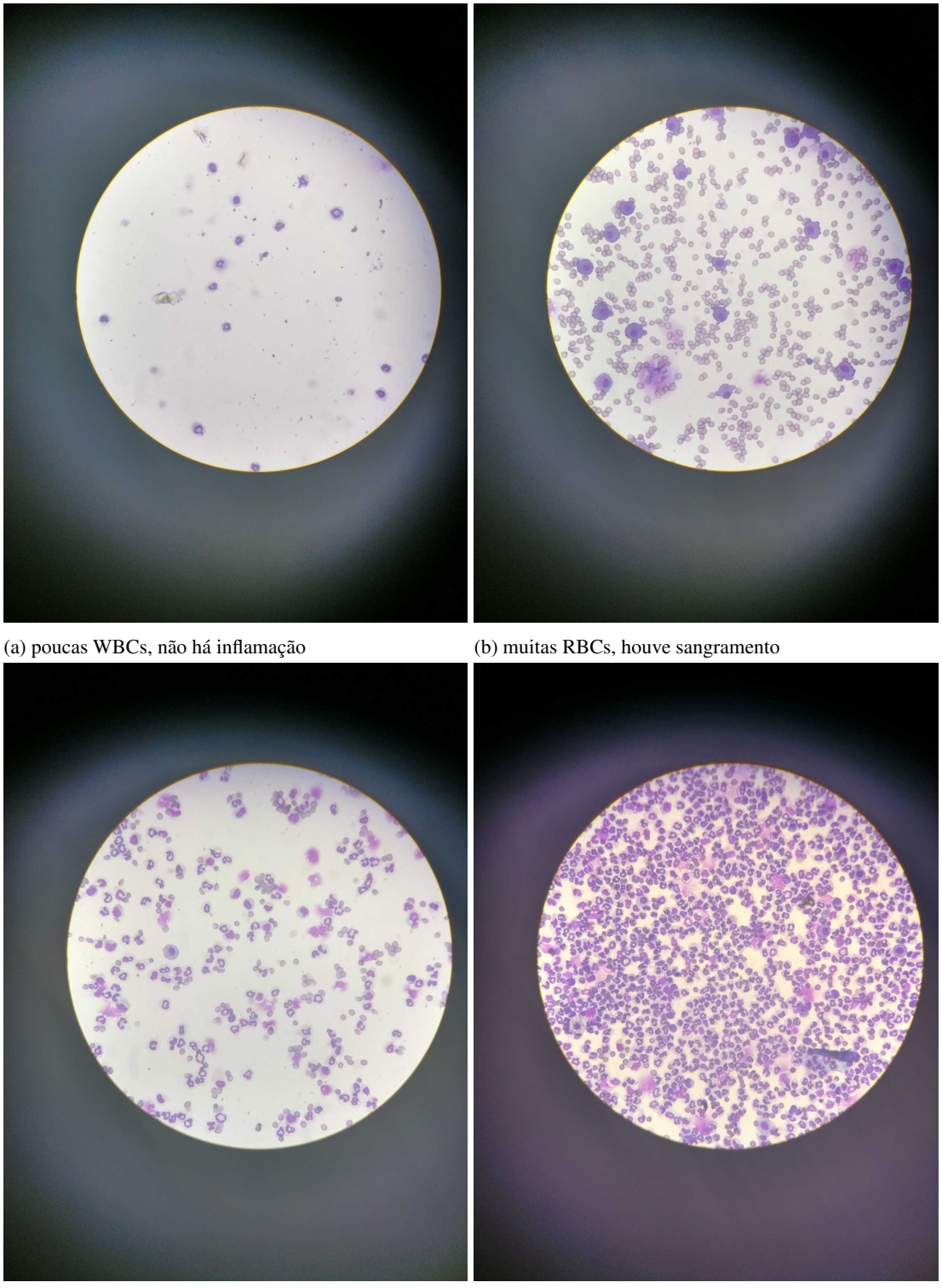
3 FUNDAMENTOS

3.1 BANCO DE IMAGENS

As imagens utilizadas neste trabalho foram cedidas por Gabriela Araujo de Azevedo, mestranda no programa de pós graduação de ciências farmacêuticas da Universidade Federal de São Paulo (UNIFESP). O termo de autorização de uso dessas imagens consta no Apêndice 6. O banco conta com 129 imagens de 3120x4160 *pixels* em formato *Joint Photographic Experts Group* (JPEG). Quatro imagens pertencentes ao banco são mostradas na Figura 2.

O processo conduzido para obtenção das imagens do banco começou com uma lavagem bronco-alveolar com solução salina em ratos para recolhimento das células. Após o recolhimento, as amostra foram centrifugada. Então, foi retirado o sobrenadante para que restassem apenas as células. Adicionou-se cristal violeta para coloração da amostra.

Sobre uma lâmina, as células foram então colocadas no microscópio com ampliação de 400 vezes. As fotografias foram retiradas a partir de um aparelho celular encaixado em um suporte localizado em um dos oculares do microscópio.



(a) poucas WBCs, não há inflamação

(b) muitas RBCs, houve sangramento

(c) processo inflamatório

(d) processo inflamatório avançado

Figura 2 – Parte do banco de imagens utilizado neste trabalho.

3.1.1 Anotações

Para validar o método automático de contagem empregado (Capítulo 4), parametrizá-lo e treinar o modelo de classificação foram necessários alguns tipos de anotações manuais. Para reduzir a anotação da base, os realizadores deste trabalho foram treinados pela especialista Gabriela Araujo de Azevedo, quem nos cedeu as imagens desta base. Além disso, o documento do Apêndice 6 foi confeccionado por ela, para referência. Os detalhes sobre cada tipo de anotação conduzida estão dispostos nas próximas seções.

3.1.1.1 Anotação da Lâmina

Refere-se ao posicionamento e tamanho da lâmina, posicionada manualmente. Essa anotação foi executada sobre 30 imagens com o auxílio da ferramenta de seleção elíptica do programa de manipulação de imagens do GNU (GIMP, *GNU image manipulation program*) (GIMP, 2017). Apesar da lâmina ser considerada a priori como tendo formato circular no método empregado para contagem (Capítulo 4), isso não foi imposto neste processo de anotação.

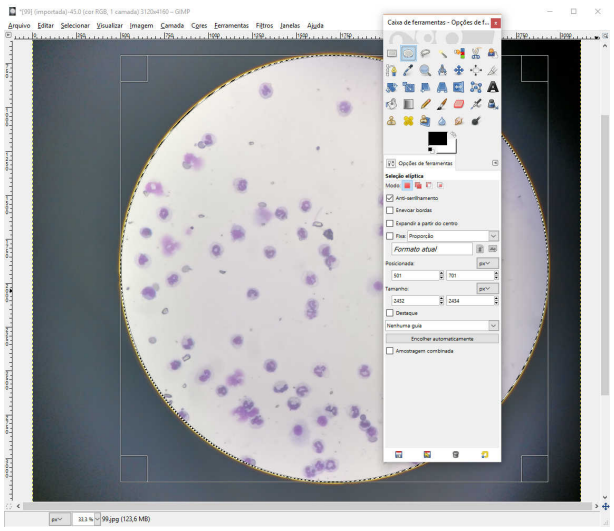


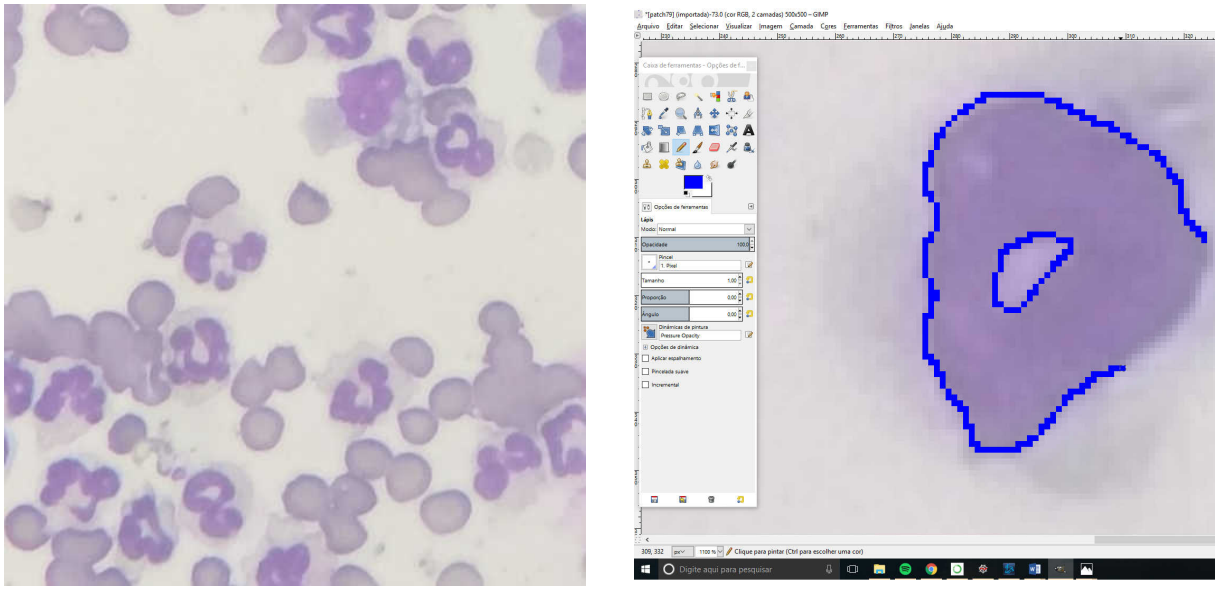
Figura 3 – Processo de anotação da lâmina.

3.1.1.2 Anotação dos Contornos

Refere-se aos contornos das células, desenhados manualmente. Para tal, foi utilizada a ferramenta lápis do GIMP (GIMP, 2017), como ilustrado na Figura 4b. Como esse processo é dispendioso, ele foi executado não sobre imagens inteiras, mas sobre *patchs* de tamanho

500x500 *pixels* (Figura 4a) posicionados aleatoriamente dentro da lâmina. Foram anotados os contornos dos leucócitos em 30 imagens aleatórias.

Das regiões referentes às células que não tocavam a borda do *patch* (e portanto teriam seu formato comprometido), mediu-se uma área média de 1725 *pixels*, com 428 *pixels* de desvio padrão.



(a) exemplo de patch

(b) processo de anotação

Figura 4 – Anotação dos contornos dos leucócitos.

3.1.1.3 Anotação dos Rótulos

A rotulação manual das regiões, provenientes da sub-etapa de extração de regiões, 4.4, consiste na anotação da classe a qual ela pertence: caso ela represente uma célula, é rotulada como "neutrófilo"ou "linfócito". Caso contrário, é designada para uma classe denominada "outros".

Além disso, foram discernidas regiões onde ocorrem um dos dois cenários: um componente conexo engloba a região referente a duas ou mais células; há mais de um componente conexo no interior de uma célula, ou seja, a região referente a ela é desconexa.

Regiões que delineiam mal o formato de uma célula, seja por encobrir apenas uma porção dela ou extrapolar as bordas do núcleo, contanto que cubram apenas uma célula e sejam a única região extraída sobre essa célula, não recebem tratamento especial: não é feita separação de regiões boas ou ruins, apenas são discernidas aquelas para as quais a relação região-célula é de um para um.

Para executar esse processo de anotação, um *script* foi escrito. Essa anotação foi conduzida sobre 350 regiões aleatórias, também de imagens aleatórias da base.

3.1.1.4 Anotação da Contagem

Foram retiradas 10 imagens da base e contadas manualmente para serem utilizadas como parâmetro de comparação à contagem feita de forma automática. A figura 5 exemplifica duas das imagens anotadas.

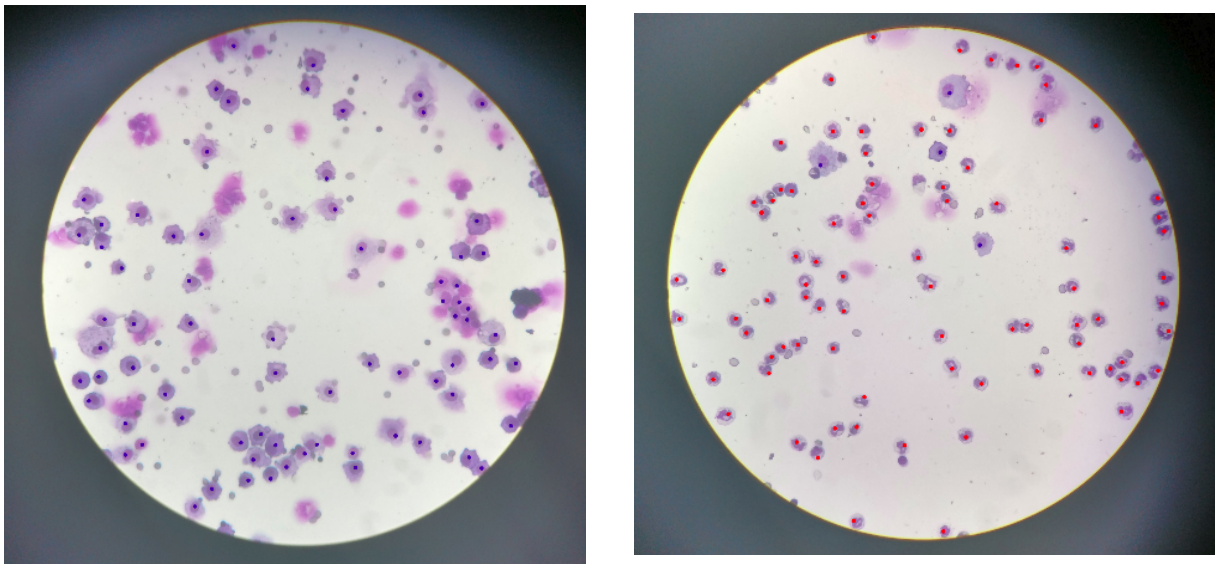


Figura 5 – Anotação das células encontradas nas imagens, os representados em vermelho referem-se a neutrófilos e em azul a Linfócitos.

3.2 PRÉ-PROCESSAMENTO DE IMAGENS

O objetivo de pré-processar uma imagem, antes que qualquer análise seja feita, é o de suprimir degradações ou detalhes indesejáveis e o de realçar a informação relevante que ela contém.

Uma degradação comum com a qual esta etapa lida é a presença de ruídos na imagem, variações aleatórias de intensidade geralmente causadas por ruído eletrônico. Ruídos podem assumir diferentes distribuições de probabilidade, conforme a sua natureza. Com base em tais distribuições, são escolhidos métodos apropriados para a sua remoção. Exemplos de tais métodos são filtros locais suavizadores, como os descritos nas próximas seções.

3.2.1 Filtro de Média

O filtro de média, sendo um filtro de suavização, é capaz de minimizar os efeitos de ruído em imagens. Como detalhado em Pedrini e Schwartz (2008, Cap. 4), a aplicação desse filtro faz com que cada *pixel* seja substituído pelo valor médio de seus vizinhos. Levando em consideração uma máscara $n \times n$, o pixel central da janela, será substituído pela média dos $n \times n$ vizinhos contidos na janela. Sendo h_1 e h_2 representação de máscaras utilizadas no filtro de média, h_1 sendo aplicado a uma janela 3×3, e h_2 a uma janela 5×5:

$$h_1 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{1}$$

$$h_2 = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2}$$

É importante ressaltar que quanto maior o grau n da máscara utilizada, maior será a influência dos vizinhos no *pixel* central. Outra consideração a ser observada é que imagens com distribuição homogênea de tons de cinza, não sofrerão grandes alterações. A Figura 6 mostra uma imagem após a aplicação do filtro de média.

(a) imagem com ruído (b) imagem após aplicação do filtro de média

Figura 6 – Demonstração da aplicação do filtro de média.
Fonte: Edinburgh (1997)

3.2.2 Filtro de Mediana

O filtro de mediana consiste em um filtro não-linear que, como descrito por Pedrini e Schwartz (2008, Cap. 4), substitui a intensidade de cada *pixel* pela mediana das intensidades na vizinhança do *pixel*. Assim como o filtro de média, o filtro de mediana é aplicado em janelas $n \times n$. Supondo a janela representada pela tabela 1.

280	265	189
198	80	191
220	165	199

Tabela 1 – Exemplo de janela de uma imagem.

Ordenando-se crescentemente os valores, a sequência obtida é: $80\ 165\ 189\ 191\ 198\ 199\ 220\ 265\ 280$ e o valor médio é 198, portanto a janela após a aplicação do filtro de mediana, é representada pela tabela 2.

280	265	189
198	198	191
220	165	199

Tabela 2 – Resultado da aplicação do filtro de mediana 3×3 sobre a janela representada na Tabela 1.

Apesar de efetivo para remoção de ruídos, o filtro de mediana é complexo e custoso, pois necessita a ordenação dos *pixels* de cada uma das janelas da imagem. A figura 7 ilustra a aplicação do mesmo.

3.2.3 Filtro Gaussiano

Os filtros Gaussianos, bem como os descritos anteriormente, são utilizados para suavização de imagens. Os coeficientes da máscara dessa técnica, que podem ser encontrados em Pedrini e Schwartz (2008, Cap. 4), são derivados a partir de uma função Gaussiana bidimensional, a função discreta de média zero e desvio padrão σ, é definida como:

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(\frac{-(x^2+y^2)}{2\sigma^2}\right) \tag{3}$$

e sua representação gráfica é apresentada pela figura 8.

(a) imagem com ruído (b) imagem após aplicação do filtro de mediana

Figura 7 – Demonstração da aplicação do filtro de mediana.
Fonte: Edinburgh (1997)

A suavização é realizada substituindo-se o *pixel* central pela média ponderada dos *pixels* vizinhos, garantindo o mesmo grau de suavização em todas as direções. A figura 9 exemplifica a aplicação desse filtro para suavização de imagem.

3.3 SEGMENTAÇÃO

Um processo de segmentação refere-se ao particionamento de uma imagem em regiões (GONZALEZ; WOODS, 2002). O objetivo é encontrar as regiões de interesse (ROIs, *regions of interest*) para a resolução de um determinado problema.

3.3.1 Limiarização

Limiarização é um processo de segmentação de imagens onde o espectro de intensidades é repartido em $n + 1$ partições contíguas, essas separadas por n limiares. A um pixel, então, é atribuído um valor referente à partição que contêm sua intensidade original (GONZALEZ; WOODS, 2002).

No caso em que $n = 1$, o resultado do processo é uma binarização. Nesse caso, cada pixel é distinguido como superior ou inferior a um único limiar L. O valor atribuído a cada pixel p na imagem binária B, portanto, pode ser descrito por

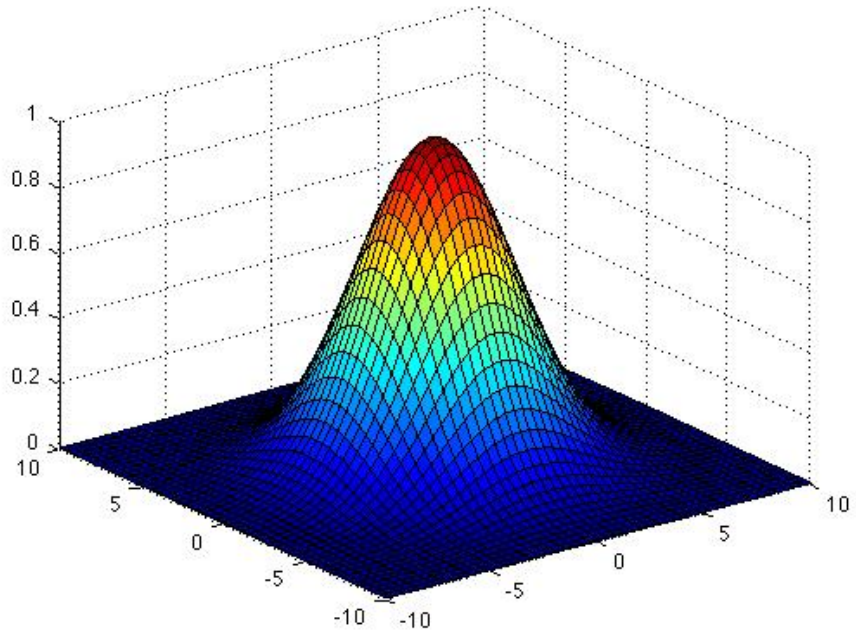


Figura 8 – Demonstração gráfica da função Gaussiana bidimensional.
Fonte: Dorigo (2011)

$$B_p = \begin{cases} 1 \;\; se \;\; I_p > \quad L \\ 0 \;\; se \;\; I_p <= L \end{cases}, \;\; \forall p$$

sendo I a imagem original de entrada e 0 e 1 os valores correspondentes às duas partições do espectro de intensidades.

Para a resolução de um problema de segmentação usando limiarização, valores apropriados de limiares precisam ser encontrados. Para tal, uma técnica utilizada é a de equilíbrio do histograma de intensidades. Nesse caso, é procurado um conjunto de limiares que otimize alguma métrica aplicada sobre o particionamento desse histograma, resultante do corte pelos limiares.

3.3.1.1 Método de Otsu

O método de Otsu (1979) é um método de binarização de imagens de único canal. O método assume que a imagem tenha duas classes de *pixels* e um histograma bi-modal. Efetivamente, ele procura, de maneira exaustiva, um limiar que separe essas duas classes de maneira a minimizar suas intra-variâncias. Isso equivale a maximizar a inter-variância entre elas (OTSU, 1979).

Figura 9 – Exemplo de uma imagem anterior e posterior à aplicação do filtro Gaussiano.

Fonte: Klemen (2014)

As intra-variâncias dos *foreground* (f) e *background* (b) da imagem, caso separados por um limiar L, são definidas por

$$\sigma_b^2 = \frac{\sum_{i=I_{min}}^{L}((i - \overline{I_b})^2 \times H_i)}{|b|} \ , \tag{4}$$

$$\sigma_f^2 = \frac{\sum_{i=L+1}^{I_{max}}((i - \overline{I_b})^2 \times H_i)}{|f|} \ , \tag{5}$$

onde H é o histograma de intensidades, $\overline{I_r}$ e $|r|$ são, respectivamente, a intensidade média e a quantidade de *pixels* de uma região r e I_{min} e I_{max} são as intensidades mínima e máxima da imagem I.

Por sua vez, a intra-variância conjunta é função a ser minimizada pelo método de Otsu. Essa função é dada por uma soma das intra-variâncias σ_b^2 e σ_f^2, ponderada pelas probabilidades de um *pixel* pertencer a cada classe. Ou seja,

$$\sigma_C^2(L) = \frac{|b|}{|I|}\sigma_b^2 + \frac{|f|}{|I|}\sigma_f^2 \ , \tag{6}$$

sendo $|I|$ a quantidade de *pixels* na imagem.

3.3.1.2 Minimização da Divergência Fuzzy

A teoria dos conjuntos *fuzzy*, diferente da teoria dos conjuntos convencional, permite que um elemento pertença parcialmente a um conjunto. Considerando uma imagem segmentada,

pode ser calculado para cada *pixel* um valor de pertinência $\mu_r(p)$ à região r ao qual ele foi designado. A métrica utilizada para determinar esse valor pode ser derivada de distribuições de probabilidade parametrizadas pela média de intensidades $\overline{I_r}$ dos *pixels* na região r. Para a segmentação de leucócitos por limiarização por minimização da divergência *fuzzy* (LMDF), a distribuição mais apropriada é a de Cauchy (GHOSH et al., 2010). A sua função de distribuição de probabilidade (FDP) é definida por

$$f(x; x_0,\gamma) = \frac{1}{\pi\gamma}\left[\frac{\gamma^2}{\gamma^2 + (x - x_0)^2}\right], \tag{7}$$

onde x_0 é o parâmetro de localização e γ o parâmetro de escala .

Sendo p um pixel pertencente a uma região r e $\mu_r(p)$ seu valor de pertinência a essa região, é definido em Ghosh et al. (2010) que

$$\mu_r(p) = c.f(I_p; \overline{I_r},1) = \frac{c}{\pi}\left[\frac{1}{1 + (I_p - \overline{I_r})^2}\right] \tag{8}$$

onde I_p é a intensidade do *pixel* p, $\overline{I_r}$ a média de intensidade dentre todos os *pixels* pertencentes à região r e c um fator de normalização definido por $c = 1/(I_{max} - I_{min})$. Adicionalmente, define-se que $\mu_I(p) = \mu_r(p) \mid (p \in r,\, r \subset I)$ é a pertinência de um *pixel* p em uma imagem segmentada I, qualquer que seja a região r a que ele pertence.

A segmentação por LMDF (CHAIRA; RAY, 2003), por sua vez, é um método de segmentação por limiarização. No caso de um único limiar, é executada uma busca linear para determinar o limiar que segmenta a imagem de maneira a minimizar a divergência *fuzzy* para com uma imagem idealmente segmentada I, ou seja, com $\mu_I(p) = 1, \forall p \in I$.

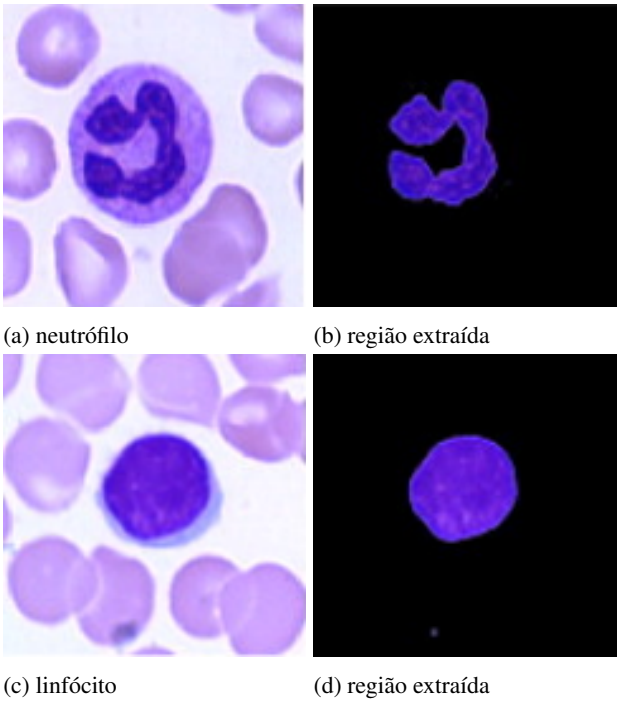
A divergência *fuzzy* entre duas imagens segmentadas A e B, baseada na entropia *fuzzy*, é definida por

$$\begin{aligned} DF(A,B) = \sum_{p \in A,B} \Big[& 2 - \Big(1 + \mu_A(p) - \mu_B(p)\Big) \cdot e^{\mu_B(p) - \mu_A(p)} \\ & - \Big(1 + \mu_B(p) - \mu_A(p)\Big) \cdot e^{\mu_A(p) - \mu_B(p)} \Big] \end{aligned} \tag{9}$$

Por fim, a divergência *fuzzy* de uma imagem A para com uma imagem idealmente segmentada I, função a ser minimizada na escolha do limiar, é definida por

$$DF_I(A) = DF(A,I) = \sum_p \Big[2 - \Big(\mu_A(p)\Big) \cdot e^{1-\mu_A(p)} - \Big(2 - \mu_A(p)\Big) \cdot e^{\mu_A(p)-1}\Big] \tag{10}$$

Um exemplo do resultado da aplicação deste método na segmentação de neutrófilos e linfócitos é exibido na Figura 10.



(a) neutrófilo (b) região extraída

(c) linfócito (d) região extraída

Figura 10 – Segmentação por limiarização por divergência *fuzzy* com função de pertinência derivada da distribuição de Cauchy. O limiar utilizado na binarização é a metade do limiar ótimo.
Fonte: Ghosh et al. (2010)

3.3.2 Operações Morfológicas

A morfologia matemática aplicada a imagens baseia-se na teoria dos conjuntos. Na sua definição para imagens binárias apresentada em Gonzalez e Woods (2002, Cap. 9), os conjuntos são membros do espaço $\mathbb{Z}^2$ e representam regiões conexas na imagem. O conjunto de *pixels* brancos (ou pretos, dependendo da convenção) em uma imagem, por exemplo, é uma descrição morfológica completa da mesma; os *pixels* restantes são o complemento da imagem.

Além das operações básicas sobre conjuntos de união e intersecção, são definidas outras operações mais complexas e com propósitos de aplicação específicos. As primeiras delas são a dilatação e a erosão, que são chamadas de operações elementares e formam base para os demais operadores. Para essas operações, é utilizado um elemento estruturante, um conjunto de *pixels* conhecido que translada pela imagem para determinar se cada *pixel* pertence ou não ao conjunto de resposta da operação. O elemento estruturante contém um pixel pré-definido,

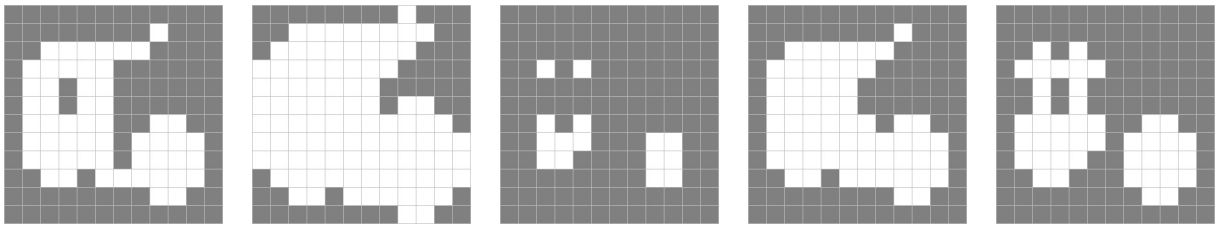
geralmente central, denominado âncora. Ao ser transladado, o elemento estruturante avalia o pixel sobre o qual a âncora está posicionada.

3.3.2.1 Dilatação

Sendo A e B conjuntos em $\mathbb{Z}^2$, a dilatação de A por B é definida por

$$A \bigoplus B = \{p|(\widehat{B})_p \bigcap A \neq \emptyset\} \tag{11}$$

Em prática, A é o conjunto de *pixels* no *foreground* da imagem e B o elemento estruturante, enquanto $(\widehat{B})_p$ o denota quando deslocado de maneira a posicionar sua âncora sobre o *pixel* p. Com isso, são pertencentes ao conjunto de resposta os *pixels* sobre os quais o elemento estrutural intersecta com A, o que amplia os objetos pertencentes na imagem. A Figura 11b mostra o resultado da dilatação do elemento estruturante formado por uma cruz de 5 *pixels* com âncora no pixel central sobre a figura 11a. Como pode ser observado, a dilatação é capaz de preencher frestas entre objetos e buracos internos, comportamento desejado em alguns cenários.



(a) imagem original (b) dilatação (c) erosão (d) abertura (e) fechamento

Figura 11 – Exemplos da aplicação de quatro operações morfológicas, utilizando como elemento estruturante uma cruz de 5 *pixels* com a âncora no *pixel* central.

3.3.2.2 Erosão

Similarmente à dilatação, a erosão de A por B é definida por

$$A \bigodot B = \{p|(\widehat{B})_p \subseteq A\} \tag{12}$$

Segundo essa definição, um *pixel* só pertence ao conjunto de saída se o elemento estruturante posicionado sobre ele cobrir apenas o *foreground*. Como isso não ocorre próximo às bordas, a erosão afina os objetos. Esse comportamento pode levar à remoção completa de objetos pequenos e de seções finas protuberantes de objetos maiores, como pode ser observado

na Figura 11c, o que é interessante se esses forem considerados irrelevantes ou resultados de erros em operadores anteriores.

É importante apontar que a erosão não desfaz a dilatação. Na realidade, elas são operações duais. Sendo A^c o complemento de A, a inversão de *foreground* e *background*, vale

$$A \bigoplus B = (A^c \bigodot B)^c \tag{13}$$

3.3.2.3 Abertura e Fechamento

As operações de dilatação e erosão, como explicado, podem ser aplicadas com o propósito de preencher frestas entre objetos ou buracos internos e de remover objetos pequenos ou seções finas protuberantes de objetos maiores. Contudo, a aplicação isolada desses operadores altera o formato dos objeto. Para que eles retornem à forma original, ou ao menos se aproximem dela, é comum a aplicação posterior da outra operação elementar; dilatação seguida de erosão ou erosão seguida de dilatação. A esses dois processos são definidos outros dois operadores, sob os nomes de fechamento e abertura, respectivamente. As figuras 11d e 11e mostram a aplicação dessas operações. Assim como a dilatação e a erosão, esses operadores também são duais. Adicionalmente, após aberta ou fechada, aplicações adicionais do mesmo operador à imagem não produzem qualquer efeito; a abertura e o fechamento são idempotentes.

3.3.3 Extração de BLOBs

A extração de objetos binários grandes (BLOBs, *binary large o bjects*) consiste na separação de áreas da imagem cujos *pixels* são similares de acordo com algum critério pré-estabelecido. Em sua forma mais simples, quando aplicado sobre imagens binárias, um detector de *blobs* faz o trabalho de rotular componentes conexos do *foreground* e do *background*. O significado prático dessa afirmação depende do tipo de conectividade utilizada, ou seja, a definição de quais *pixels* do entorno de um *pixel* p são seus vizinhos. Geralmente são utilizadas as 4-conectividade ou 8-conectividade, ilustradas na Figura 12 junto com os resultados da extração. A separação de componentes conexos pode ser realizada da mesma maneira como feito sobre um grafo genérico, por meio da repetição de qualquer processo de expansão ou busca. Nesse caso, a conectividade entre os *pixels* é representada pelas arestas do grafo.

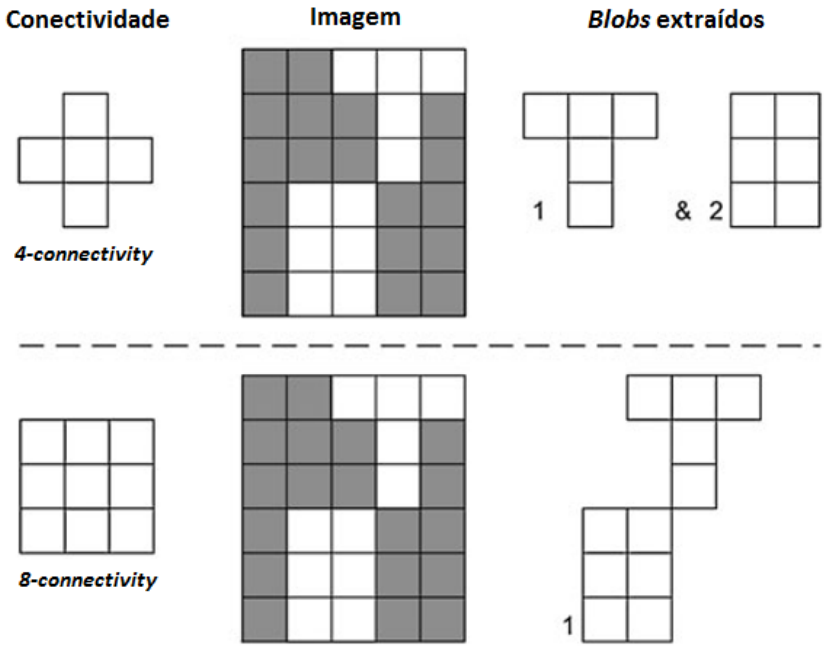


Figura 12 – Ilustração de diferentes tipos de conectividade. Com a 4-conectividade, só são vizinhos de um *pixel* aqueles adjacentes horizontal ou verticalmente a ele, enquanto que no caso da 8-conectividade os adjacentes diagonalmente também fazem parte da vizinhança.
Fonte: Whatwhenhow (2017)

3.3.3.1 Seguimento de Borda

O método de seguimento de borda (SUZUKI; ABE, 1985) é utilizado para encontrar os contornos dos componentes conexos de uma imagem binária. Esses componentes são denotados 1-componentes e 0-componentes, com base no valor atribuído aos seus *pixels*. Além disso, há uma distinção entre contornos externos e internos. Supondo que, por convenção, o *foreground* de uma imagem seja composto pelos *pixels* de valor 1, então os 1-componentes são delineados pelos contornos externos e os 0-componentes, pelos internos. Na prática, os contornos externos delineiam os artefatos encontrados na imagem, enquanto os internos delineiam os buracos desses objetos.

O processo efetuado para computar esses contornos é apresentado informalmente no Algoritmo 1. Basicamente, a imagem é varrida na sequência ilustrada na Figura 13 (linha 6). Uma mudança no valor do *pixel* sendo examinado significa que uma borda foi encontrada. Quando isso acontece (linha 7), o algoritmo checa se aquele *pixel* está contido em um contorno que já foi descoberto anteriormente (linha 8). Se esse não for o caso, ele instancia um novo contorno (linha 9) e o percorre (*segue a sua borda*) conforme ilustrado na Figura 14, incluindo nele todos os *pixels* por que passar.

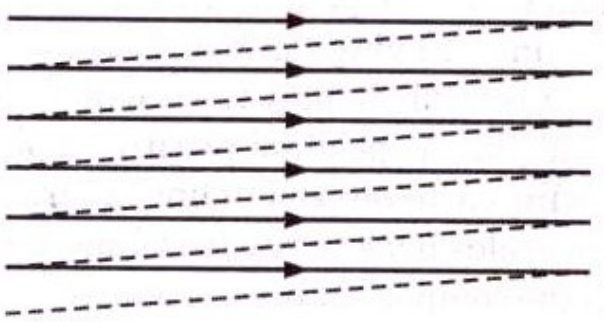
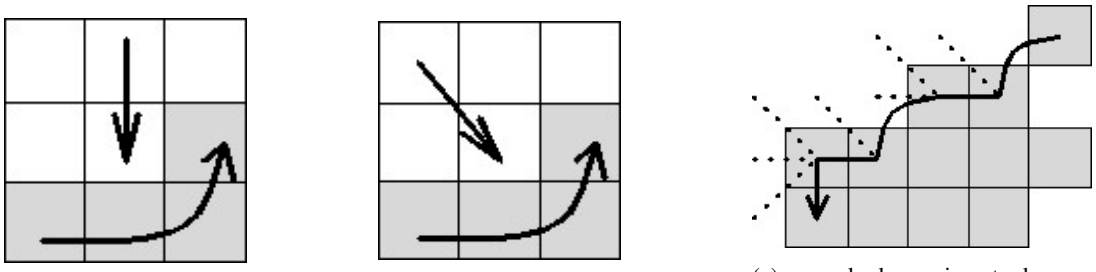


Figura 13 – Ordem de varredura dos *pixels* efetuada pelo método de seguimento de borda. A ordem apresentada é a mesma que a de um escaneamento padrão em uma imagem rasterizada.
Fonte: Digital... (2003)



(a) ordem de checagem (b) ordem de checagem (c) exemplo de seguimento de borda

Figura 14 – Ilustração do método de seguimento de borda usando 8-conectividade (Figura 12). Para cada *pixel* percorrido e com base na direção de onde o seguimento veio, checa o valor dos *pixels* vizinhos conforme a ordem exibida nas Figuras a e b. Dos *pixels* checados, o algoritmo segue para o primeiro deles com valor igual ao do *pixel* atual. Se nenhum deles satisfazer essa condição, ele volta para o *pixel* anterior.
Fonte: Digital... (2003)

Além de encontrar o contorno dos componentes conexos (C), o método de seguimento de borda também retorna uma hierarquia de englobamentos de contornos F (Figura 15). Essa hierarquia indica quando um contorno está dentro de outro. Para computá-la, o algoritmo mantém os contornos por quem passa em uma pilha E. Ao encontrar um contorno, seu rótulo é comparado com o topo da pilha (linha 17). Se forem iguais, isso significa que a varredura da imagem (linha 6) está saindo da região delimitada por aquele contorno, e então ele é desempilhado (linha 18). Caso contrário, ela está entrando e o contorno é empilhado (linhas 11 e 21). Sabendo-se a qualquer momento qual contorno delimita imediatamente a região na qual a varredura está posicionada, esse contorno é definido como predecessor de qualquer novo contorno instanciado, o que fica registrado na hierarquia de englobamentos (linha 10).

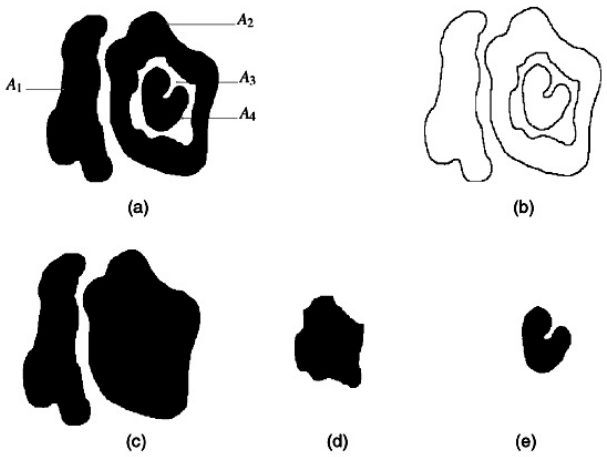


Figura 15 – Sobre os componentes conexos da Figura a, são apresentados os contornos encontrados pelo método de seguimento de borda (Figura b). As Figuras c a e apresentam as regiões delineadas pelos contornos dos primeiro (A_1 e A_2), segundo (A_3) e terceiro (A_4) níveis hierárquicos.
Fonte: Hasan e Karam (2000)

Algoritmo 1 – Seguimento de Borda

Entrada: a imagem binária I
Saída: os contornos dos componentes conexos, rotulados em C para cada $pixel$; a hierarquia de englobamento dos contornos expressada em termos de predecessão, indicada em F para cada contorno
início
  $E \leftarrow$ nova pilha vazia;
  empilha($E, -1$);
  para *cada pixel* p *em* I*, exceto o primeiro, percorridos de acordo com a Fig. 13* **faça**
    se $I_p \neq I_{p\,anterior}$ **então**
      se C_p *não foi atribuído* **então**
        $c \leftarrow$ novoRotuloContorno();
        $F_c \leftarrow$ topo(E);
        empilha(E, c);
        para *cada pixel* p' *na borda partindo de* p*, percorridos de acordo com a Fig. 14* **faça**
          $C_{p'} \leftarrow c$;
        fim
      fim
      senão
        se $C_p = topo(E)$ **então**
          desempilha(E);
        fim
        senão
          empilha(E, c);
        fim
      fim
    fim
  fim
fim

3.3.4 Distância de Jaccard

A distância de Jaccard (JACCARD, 1901) é uma métrica que mensura a dissimilaridade de dois conjuntos. Sendo $IJ(A,B)$ o índice de Jaccard entre dois conjuntos A e B,

$$IJ(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

A distância de Jaccard $DJ(A,B)$ é o seu complemento para um, ou seja, $DJ(A,B) = 1 - IJ(A,B)$.

No processamento de imagens, ela é utilizada para medir a similaridade de duas regiões, essas compreendidas como conjuntos de *pixels*. Com isso, a distância de Jaccard pode ser utilizada como métrica para avaliação da segmentação de imagens médicas (TAHA; HANBURY, 2015). Nesse caso, o índice de Jaccard pode ser definido em função do número de *pixels* verdadeiro positivos (VPs), falso positivos (FPs) e falso negativos (FNs) em uma segmentação A comparada com o padrão ouro B.

$$IJ(A,B) = \frac{VP}{VP + FP + FN}$$

3.4 BRUTE FORCE MATCHING

O *matching* por força bruta (BF *matching*, *brute force matching*) é, de maneira genérica, uma forma de encontrar elementos correspondentes em dois conjuntos A e B. Na prática, ele seleciona um elemento $a \in A$ e o compara com todos os elementos $b \in B$ usando alguma métrica de distância d fornecida, seu *kernel*. O menor valor encontrado, então, refere-se ao elemento de B que mais se aproxima de a:

$$C_{d,B}(a) = c \in B \mid d(a,c) = \min_{b \in B} d(a,b) \tag{14}$$

Esse processo é repetido para todo o conjunto A, o que resulta em um mapeamento completo $C_{d,B}$ de elementos correspondentes. Se relevante, podem ser estabelecidos critérios de desempate.

Como consequência desse processo, é gerado um conjunto de distâncias mínimas

$$M_d(A,B) = \{d\Big(a, C_{d,B}(a)\Big) \mid a \in A\} \tag{15}$$

Por fim, as tendências centrais de $M_d(A,B)$ são possibilidades de métrica de similaridade entre A e B. Dessa maneira, o BF *matching* pode ser utilizado para avaliar a segmentação de uma imagem. Nessa caso, são feitas comparações entre as regiões da segmentação e as regiões presentes no padrão ouro. Neste trabalho, o *kernel* do BF *matching* e a tendência central de $M_d(A,B)$ utilizados são, respectivamente, a distância de Jaccard (Seção 3.3.4) e a média aritmética.

3.5 EXTRAÇÃO DE CARACTERÍSTICAS

Como abordado em Pedrini e Schwartz (2008), a análise de imagens pode ser feita pela análise de características das suas regiões. Segundo Saraswat e Arya (2014), essas características podem ser divididas em duas categorias: geométricas, que se referem ao contorno da região e ao seu formato, e de textura, que avaliam os valores dos *pixels* internos à região.

Algumas características de ambos os tipos são definidas nas próximas seções.

Neste trabalho, as regiões são representadas pelo formato de cadeia: é utilizada uma lista contendo os *pixels* pertencentes ao contorno sendo descrito. Não há aproximação de segmentos, mesmo quando uma sequência de *pixels* forma uma linha reta: são sempre armazenados todos os *pixels* no contorno.

Neste trabalho, as regiões manipuladas são sempre conexas (garantido pelo método de extração descrito na Seção 4.4). Desta maneira, um único contorno externo é suficiente para englobar os *pixels* pertencentes à região. Contudo, a região pode conter buracos. Portanto, uma região r é representada por um contorno externo $C_{r,0}$ e n contornos internos, $C_{r,i}, 1 <= i <= n$, que delimitam os buracos internos à região.

3.5.1 Geométricas

3.5.1.1 Área

Número de *pixels* contidos na região:

$$area(r) = area(C_{r,0}) - \sum_{i=1}^{n} area(C_{r,i}), \tag{16}$$

sendo que a área de um contorno C, $area(C)$, é definida pelo número de *pixels* no seu interior, ou seja,

$$area(C) = \sum_{p \in regiao(C)} 1, \tag{17}$$

onde $regiao(C)$ é a região delimitada pelo contorno C.

3.5.1.2 Perímetro

Comprimento total dos contornos da região, internos e externos:

$$perimetro(r) = comprimento(C_{r,0}) + \sum_{i=1}^{n} comprimento(C_{r,i}), \tag{18}$$

Para calculá-lo, é somada a distância entre cada par de *pixels* adjacentes. Cada par contribui para um somatório com uma parcela que depende da sua posição relativa um ao outro: se estiverem em diagonal, somam $\sqrt{2}$, caso contrário, somam 1. Formalmente,

$$comprimento(C) = \left(\sum_{i=1}^{|C|-1} distancia(C_i, C_{i+1}) \right) + distancia(C_1, C_{|C|}) \tag{19}$$

$$distancia(p, p') = \begin{cases} 1 \;\; se \;\; p_x = p'_x \vee p_y = p'_y \\ \sqrt{2} \;\; senão \end{cases}, \tag{20}$$

sendo C_i o i-ésimo pixel em C, pela ordem da lista do formato de cadeia, $|C|$ a quantidade de *pixels* em C e p_x e p_y as coordenadas do *pixel* p.

3.5.1.3 Área do Retângulo Mínimo Rotacionado

Consiste em calcular a área do menor retângulo rotacionado que englobe todo o contorno do objeto. O retângulo mínimo rotacionado é representado pela figura 16.

3.5.1.4 Retangularidade

Cálculo da razão entre a área do menor retângulo rotacionado e a área da região:

$$retangularidade(r) = \frac{area_{retangulo}}{area_{regiao}} \tag{21}$$

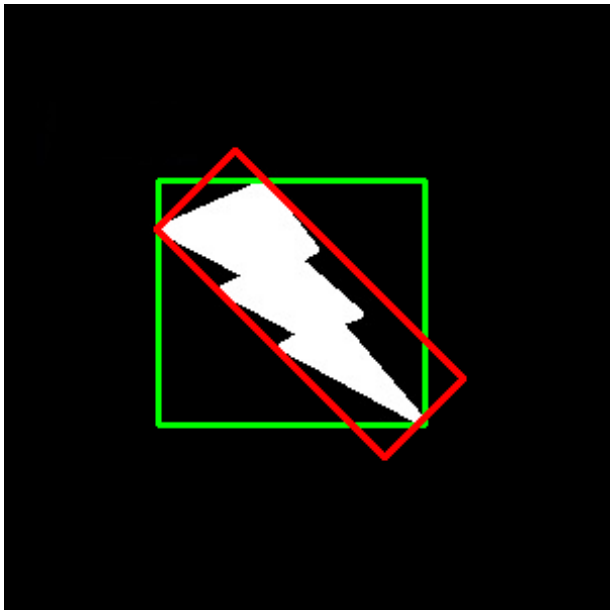


Figura 16 – Retângulo mínimo rotacionado em vermelho.
Fonte: OpenCV... (2017)

3.5.1.5 Área círculo Mínimo

Consiste em calcular a área do menor círculo que englobe todo o contorno do objeto. O círculo mínimo é representado pela figura 17.

3.5.1.6 Circularidade

Cálculo da razão entre a área do menor círculo que englobe a região e a área da região:

$$circularidade(r) = \frac{area_{circulo}}{area_{circulo}} \tag{22}$$

3.5.1.7 Compacidade

Relação obtida entre o perímetro e a área de um objeto, onde o menor valor é encontrado em um círculo, com compacidade de 4π. Calcula-se:

$$Circularidade = \frac{P^2}{A} \tag{23}$$

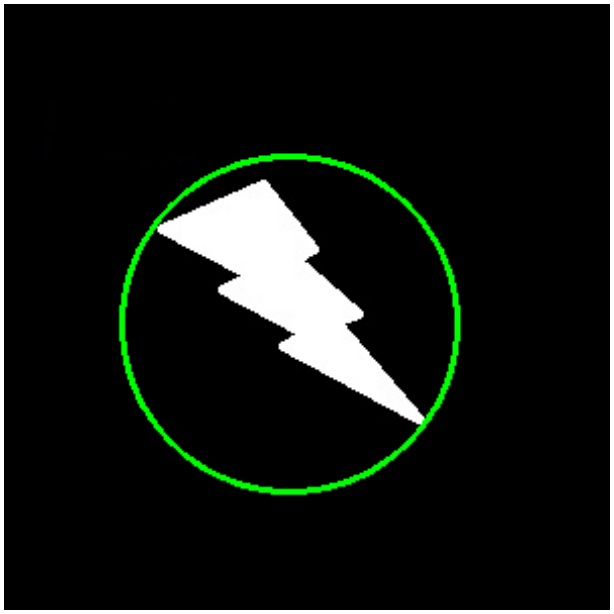


Figura 17 – Círculo mínimo representado em verde.
Fonte: OpenCV... (2017)

3.5.1.8 Área do Convex Hull

Objetiva calcular a área do menor polígono que englobe um determinado conjunto de pontos. O polígono convex hull pode ser observado na figura 18.

3.5.1.9 Solidez

Relação entre a área da região, e a área de seu fecho convexo:

$$Solidez = \frac{A}{A_{fechoconvexo}} \tag{24}$$

3.5.2 Textura

3.5.3 Média

A média é calculada pela somatória do valor de cada pixel de uma região, dividida pela quantidade de pixels na região.

$$Média = \frac{\sum_{i=0}^{Qpixels-1} pixel[i]}{Qpixels} \tag{25}$$

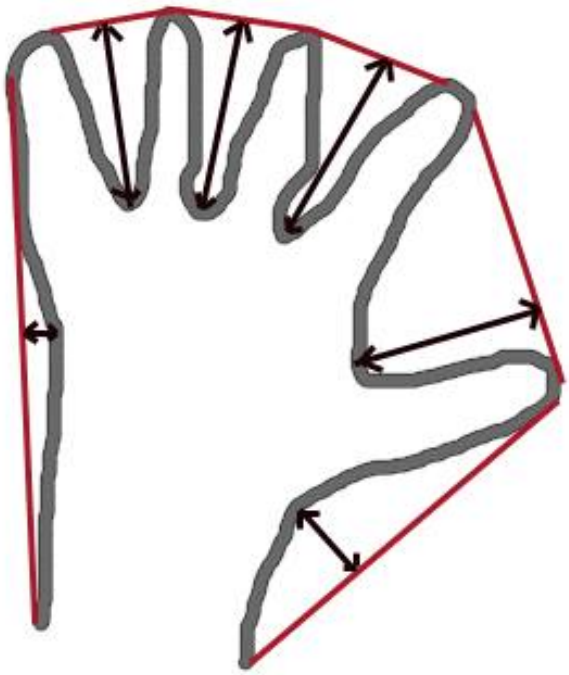


Figura 18 – Convex Hull do objeto está representado em vermelho.
Fonte: OpenCV... (2017)

3.5.3.1 Variância

A variância, medida de heterogeneidade, é apresentada por:

$$f_{var_i} = \sum_{i=0}^{H_g} \sum_{j=0}^{H_g} (i - \mu_i)^2 p_{i,j} \tag{26}$$

$$f_{var_j} = \sum_{i=0}^{H_g} \sum_{j=0}^{H_g} (j - \mu_j)^2 p_{i,j} \tag{27}$$

onde:

$$\mu_i = \sum_{i,j=0}^{H_g} i P_{i,j} \tag{28}$$

$$\mu_j = \sum_{i,j=0}^{H_g} j P_{i,j} \tag{29}$$

3.5.3.2 Desvio Padrão

O desvio padrão é calculado pelo resultado positivo da raiz quadrada da variância, ou seja:

$$DesvioPadrão = \sqrt{f_{var}} \tag{30}$$

3.6 CLASSIFICAÇÃO

Aprender é a capacidade de um sistema de melhorar sua performance futura com base em observações. No aprendizado supervisionado, o sistema recebe um conjunto de dados para treinamento $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$ e gera uma função de hipótese h que se aproxima de uma função desconhecida $f(\mathbf{x}_i) = y_i$, $i = 1, 2, ..., n$ (RUSSELL; NORVIG, 2003). Em um processo estocástico, f (e portanto h) pode ser uma distribuição de probabilidade ao invés de uma função.

A hipótese h pode, então, ser utilizada para predizer o valor de y para um valor de **x** provido, mesmo para dados novos (i.e. não pertencentes ao conjunto de treinamento). À capacidade de prever corretamente sobre dados novos é dado o nome de generalização, que é comprovada por um processo de validação.

Um sistema de aprendizado supervisionado é um classificador se y é o rótulo de uma classe, em oposição a sistemas de regressão, onde y é um número. A hipótese h de um classificador é escolhida de um espaço de hipótese, todas as funções que podem ser alcançadas pelo modelo sendo utilizado. O modelo máquina de vetor de suporte (SVM, *support vector machine*), por exemplo, é um espaço de hipótese.

3.6.1 Máquina de Vetores de Suporte

O aprendizado estatístico não é apenas uma ferramenta para análise teórica, mas também uma ferramenta para criação de algoritmos práticos para estimar funções multidimensionais. Como detalhado no estudo de Vapnik (1999), essa abordagem resolve problemas práticos como reconhecimento de padrões, estimativa de regressão e estimativa de densidade. As amostras são representadas pelos termos dos seus atributos ou das suas características, e tem-se como objetivo encontrar a equação da reta que, dado um conjunto de objetos **x** = $[\ x_1\ x_2\ \ldots\ x_n\ ]$, classifique os elementos corretamente quando representados em um plano cartesiano de características.

Dentre as técnicas de aprendizado estatístico, está a SVM. Essa técnica é composta por um aprendizado supervisionado que, como referido em Lorena e Carvalho (2007), dado um conjunto de amostras (x_i, y_i), onde x_i representa um elemento e y_i representa sua classe, consiga resultar em um classificador capaz de, com dados de classe desconhecida, predizer de forma precisa suas respectivas classes. Ou seja, os novos objetos são rotulados a partir de características encontradas no conjunto de amostras já classificadas por um observador. O grupo de amostras já especificado, é chamado de treinamento.

Nascimento et al. (2009) explica o algoritmo da SVM como: dadas N amostras de treinamento (x_i,y_i), com $i = 1, 2, ..., N$ e $x_i \in R$ é uma representação vetorial de um conjunto e $y_i \in \{-1,1\}$ sua classe, o objetivo é encontrar o hiperplano que divide os elementos x_i entre as classes -1 e 1, o risco empírico dos treinamentos é calculado pela fórmula

$$\varepsilon_\psi(\varsigma) = \frac{1}{2N}\sum_{i=1}^{N}|y_i - f(x_i, \varsigma)| \tag{31}$$

sendo que $\varepsilon_\psi(\varsigma)$ é fixo para um ς arbitrário e um conjunto de treinamento $\{x_i,y_i\}$.

Como exemplificado na 19, a separação ótima entre as classes feita pela SVM, é obtida pelo hiperplano L, sendo L_1 e L_2 os hiperplanos paralelos orientados pelo ponto mais próximo a L de cada classe, a maximização de D, distância entre os hiperplanos L_1 e L_2, garante uma taxa de erro menor à classificação por essa técnica.

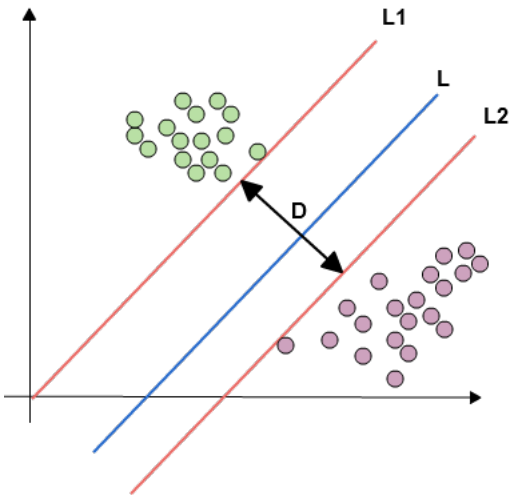


Figura 19 – Representação gráfica do hiperplano L de uma SVM dividindo linearmente duas classes e dos hiperplanos L1 e L2 traçados a partir dos pontos mais próximos.

3.6.1.1 Padronização

Considerando que cada uma das características que serão inseridas na SVM possuem diferentes amplitudes, deve-se utilizar um método de padronização dos mesmos. O método geral mais comum é obtido com a razão entre a diferença da amostra e da média, pelo desvio padrão, como mostrado pela fórmula:

$$x' = \frac{x - \overline{x}}{\sigma} \tag{32}$$

Sendo $\overline{x}$ a média dos valores de uma característica na base e σ o desvio padrão da mesma característica na base.

3.6.2 Análise por Curvas de ROC

Um gráfico de característica operacional de receptor (ROC, *reciever operating characteristic*) é uma técnica para visualizar, organizar e selecionar classificadores com base em sua performance (FAWCETT, 2006). Considerando a matriz de confusão mostrada na Figura 20 para classificadores discretos binários (i.e. que distinguem entre duas classes, neste caso verdadeiro e falso), o espaço de ROC é dimensionado pelas taxas de VPs e FPs, definidos por

$$taxa\ vp = \frac{VP}{VP + FP}$$

$$taxa\ fp = \frac{FN}{VN + FN}$$

		Saída da Hipótese: V	Saída da Hipótese: F
Classe Real	V	Verdadeiros Positivos (**VP**)	Falso Negativo (**FN**)
	F	Falso Positivo (**FP**)	Verdadeiro Negativo (**VN**)

Figura 20 – Matriz de confusão, também chamada de matriz de erro.

Classificadores discretos (que tem como saída o rótulo de uma classe) podem ser mapeados para pontos no espaço de ROC (Figura 21). Os pontos que recaem sobre a linha central representam classificadores virtualmente randômicos. Quanto mais próximo do canto superior esquerdo, melhor a acurácia de um classificador.

Classificadores que, por outro lado, tem como saída um valor escalar, podem ser reduzidos a classificadores binários com a introdução de um limiar. Variar esse limiar gera vários pontos correspondentes no espaço de ROC, que formam uma curva. A área sob a curva de ROC pode ser utilizada para medição de performance de classificadores não discretos (FAWCETT, 2006).

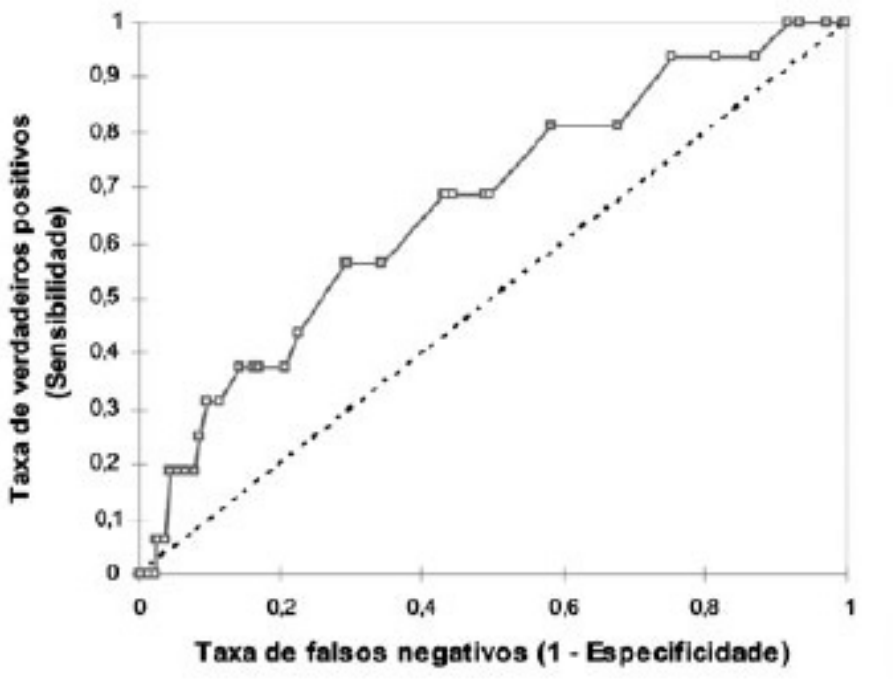


Figura 21 – Espaço de ROC com uma curva aproximada por múltiplos pontos, referentes ao desempenho do classificador binarizado por limiares diferentes.
Fonte: Invasiva (2012)

3.6.3 Validação Cruzada K-Fold

A validação cruzada é uma técnica para avaliar a capacidade de generalização de um classificador, estimando a sua precisão para conjuntos de dados novos. A validação cruzada por k-*fold* (KOHAVI, 1995) particiona o conjunto de dados em k subconjuntos de similar dimensão. O modelo é, então, treinado com $k-1$ desses subconjuntos e validado pelo subconjunto restante, produzindo uma medida de acurácia a partir de alguma métrica. Esse processo é repetido k vezes, alternando o subconjunto de validação. Por fim, a média das acurácias de cada iteração é tomada como uma medida mais confiável (e genérica) de performance.

4 METODOLOGIA

O método automático utilizado neste trabalho se delimita a resolver o problema de, provida uma imagem microscópica, efetuar uma contagem diferencial automática de linfócitos e neutrófilos. A abordagem utilizada é dividida em três etapas principais, cada uma das quais contendo duas sub-etapas, listadas na Figura 22.

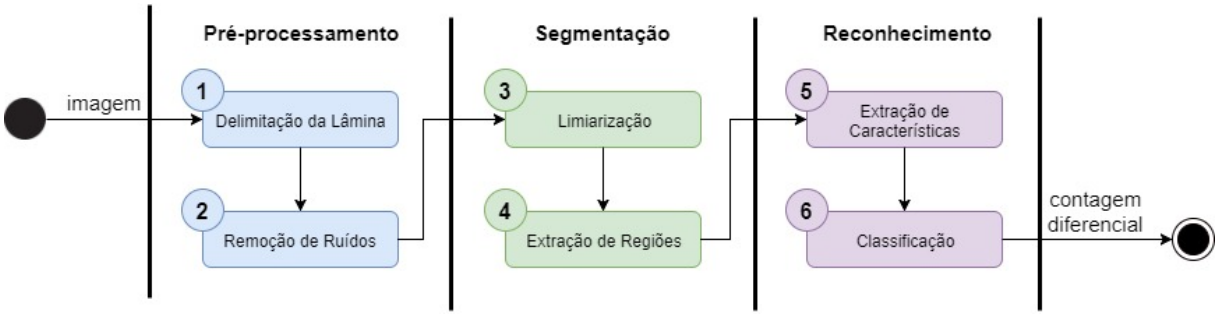


Figura 22 – Etapas do método automático de contagem diferencial de leucócitos empregado.

Durante a etapa de pré-processamento, objetiva-se adequar a imagem ao procedimento que se seguirá. Isso se dá pela delimitação da região circular do visor do microscópio (sub-etapa 1), única parte da imagem que contém informações de interesse para a resolução do problema (as células), seguida pelo realce dessas informações através da remoção de ruídos (sub-etapa 2).

Na etapa seguinte, a de segmentação, são buscadas as células contidas na imagem. A abordagem escolhida neste trabalho é a de detectar os seus núcleos e proceder com a contagem apenas sobre eles, ignorando os ectoplasmas. Para tal, a imagem é primeiro limiarizada (sub-etapa 3). Então, seus componentes conexos são encontrados (sub-etapa 4) e enviados como regiões de interesse para a próxima etapa.

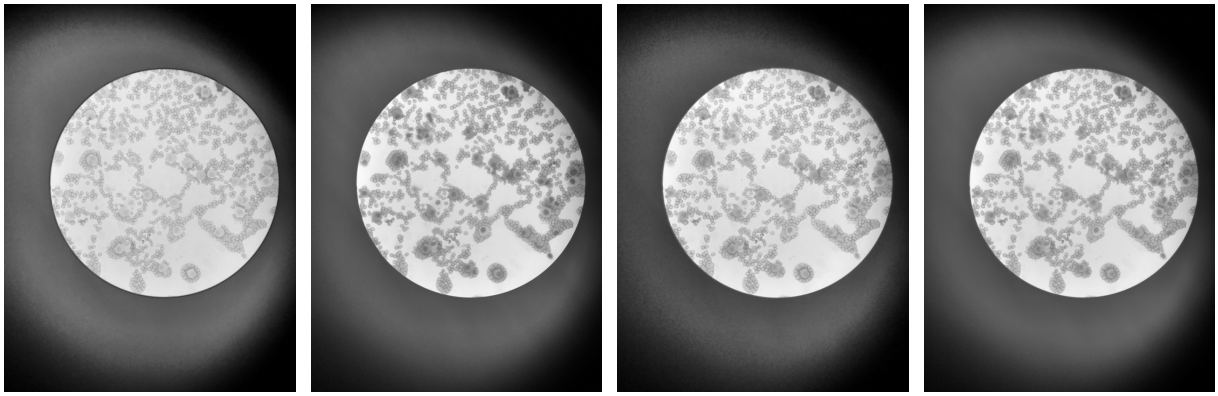
Cada objeto segmentado pode ser um linfócito, um neutrófilo ou algum outro artefato (principalmente uma hemácia ou uma mancha do tingimento). Cabe à terceira etapa reconhecer as duas células sendo contadas, o que é feito pela análise das características da região. Uma vez extraídas (sub-etapa 5), as características são utilizadas como entrada por um modelo de classificação (sub-etapa 6) que distingue linfócitos e neutrófilos de outros artefatos e incrementa a contagem diferencial quando há reconhecimento.

Cada uma das 6 sub-etapas mostradas na Figura 22 é descrita em mais detalhes nas seções seguintes. Os códigos produzidos estão inclusos nos apêndices.

4.1 DELIMITAÇÃO DA LÂMINA (SUB-ETAPA 1)

Como mostrado na Figura 2 presente na descrição da base (Seção 3.1), as imagens de entrada do método contém uma região circular, referente ao visor do microscópio, onde encontram-se as informações de interesse à contagem de leucócitos; as próprias células. Esta sub-etapa consiste na localização dessa região, doravante chamada de lâmina. Todos os processamentos posteriores são executados apenas sobre a lâmina.

Para atingir esse objetivo, é inicialmente empregada uma binarização pelo método de Otsu (Seção 3.3.1.1). O balanceamento de histograma é feito sobre o canal RGB azul, onde as intensidades das células são mais próximas das intensidades do fundo da lâmina e mais distantes das intensidades do exterior da lâmina. Isso contribui para que o método de Otsu designe células e porções do fundo da lâmina à mesma classe, ao invés de atribuir *pixels* do interior das células, principalmente no núcleo, onde são mais escuras, ao *background*. Essa conclusão pode ser intuitivamente observada na Figura 23 e também tem respaldo nos resultados do experimento 1 (Seção 5.1).

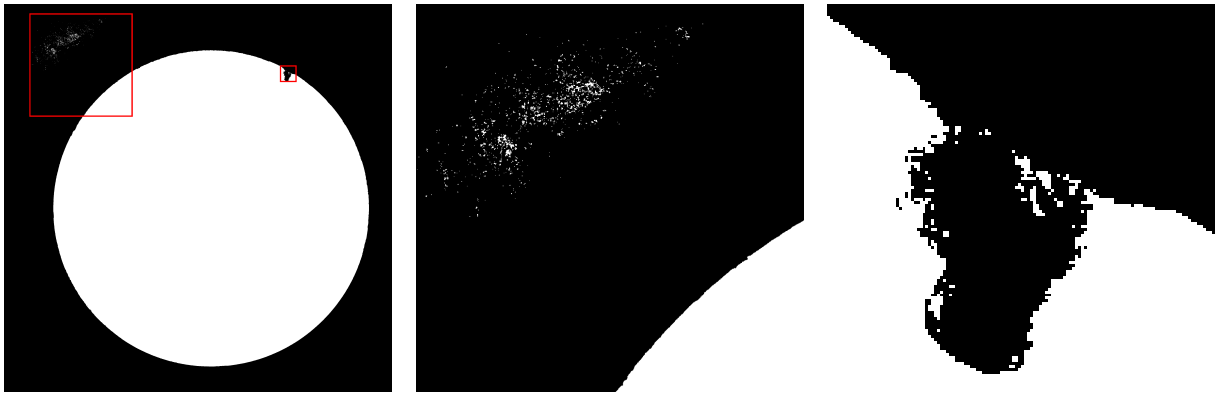


(a) azul (b) verde (c) vermelho (d) cinza

Figura 23 – Comparação visual dos canais RGB (e da sua combinação para tons de cinza). Para visualização, as primeiras três imagens tiveram os valores de intensidade do canal representado replicados nos outros canais.

A mascara resultante, como ilustrado na Figura 24, pode conter regiões espúrias além da lâmina. Com o propósito de removê-las, são ignorados todos os componentes conexos do *foreground* (conforme definido na Seção 3.3.3 e com 8-conectividade), exceto o de maior área, que assume-se ser a lâmina.

Sobre o componente conexo de maior área, por fim, é encaixado um circulo, que é o formato esperado do objeto sendo reconhecido (a lâmina). Para tal, o centro do círculo iguala



(a) mascara obtida (b) regiões espúrias no fundo preto (c) regiões espúrias na borda da lâmina

Figura 24 – Regiões espúrias presentes na máscara obtida da binarização pelo método de Otsu para delimitação da lâmina.

o centro de massa da região e seu raio é calculado a partir da fórmula $r = \sqrt{A/\pi}$, onde A é a área da região.

A validez da introdução dos passos de remoção de regiões espúrias e encaixe de círculo anteriormente descritos foram assertadas empiricamente no experimento 1 (Seção 5.1).

4.2 REMOÇÃO DE RUÍDOS (SUB-ETAPA 2)

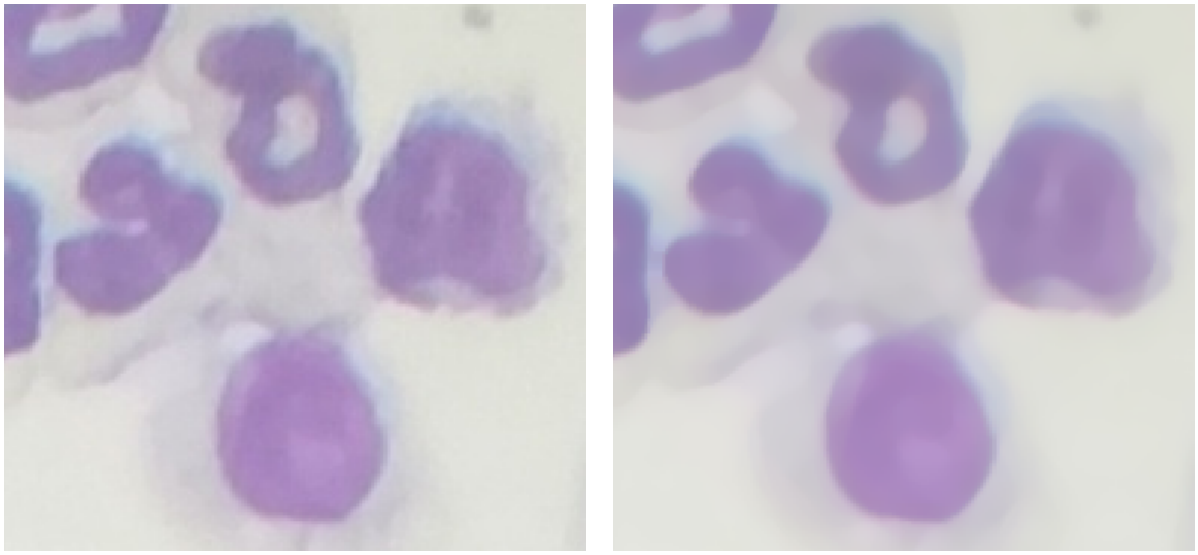
As imagens submetidas a este método são provenientes de câmeras de *smartphones* comuns (Seção 3.1). Portanto, a presença de ruídos na imagem e a consequente necessidade de um pré-processamento para removê-los (Seção 3.2) é esperada.

A sub-etapa de remoção de ruídos consiste na aplicação de um filtro de mediana (Seção 3.2.2) com *kernel* de tamanho 9x9. Essa escolha baseou-se nos resultados do experimento 2 (Seção 5.2), que condizem com a literatura (SARASWAT; ARYA, 2014). A Figura 25 exemplifica a aplicação desse filtro.

4.3 LIMIARIZAÇÃO (SUB-ETAPA 3)

Esta sub-etapa objetiva separar a área da imagem pertencente a leucócitos do seu complemento. Mais especificamente, são procurados os núcleos dos leucócitos.

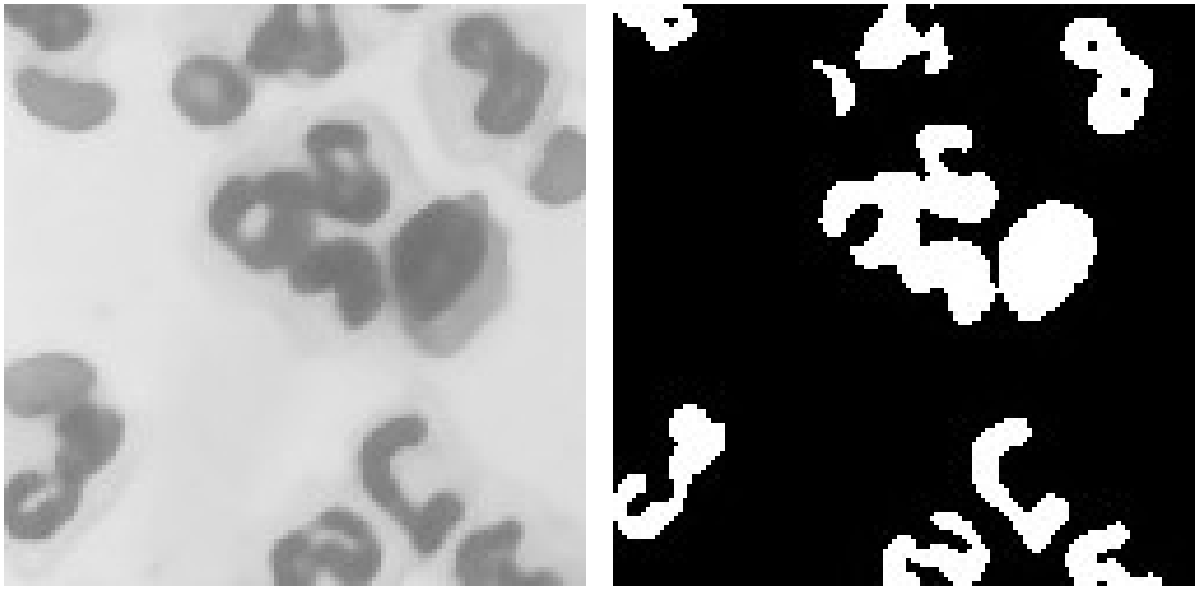
Para tal, um processo de limiarização é aplicado sobre o histograma do canal RGB verde da imagem. Esse canal tem mais diferença de intensidade entre as células e o fundo da lâmina, como pode ser observado na Figura 23. Além disso, os resultados do experimento 3 (Seção 5.3) suportam o uso desse canal, e também há precedentes na literatura (TOMARI et al., 2014).



(a) imagem original (b) imagem filtrada

Figura 25 – Demonstração da aplicação do filtro de mediana com *kernel* tamanho 9x9, filtro empregado na sub-etapa de remoção de ruídos.

É utilizado um limiar fixo, na intensidade 146, como determinado pelo experimento 3 (Seção 5.3). A Figura 26 exemplifica esta sub-etapa.



(a) canal verde (b) imagem binária resultante da limiarização

Figura 26 – Demonstração da limiarização de histograma no limiar de valor 146 sobre o canal RGB verde de uma imagem.

4.4 EXTRAÇÃO DE REGIÕES (SUB-ETAPA 4)

Distinguido o *foreground* do *background* na sub-etapa de limiarização (Seção 4.3), é necessário separar as regiões referentes a cada célula ou outro artefato presente na lâmina. Para tal, esta etapa emprega uma extração de BLOBs (Seção 3.3.3) para encontrar os componentes conexos do *foreground*. O método utilizado é o de seguimento de borda (Seção 3.3.3.1). A Figura 27 ilustra o resultado desse processo.

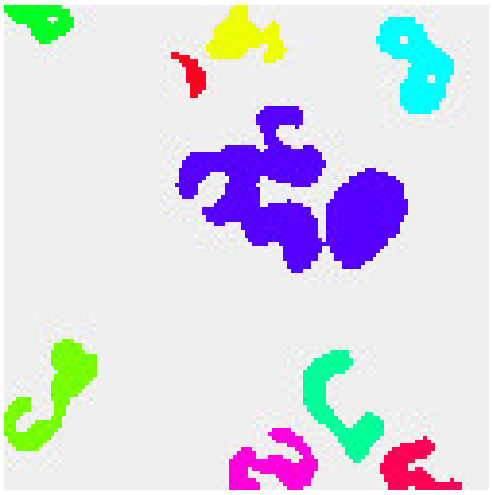


Figura 27 – Ilustração da extração de regiões conexas. Cada cor rotula uma região conexa do *foreground* (*pixels* brancos) da imagem da Figura 26b.

A utilização de um operador morfológico de abertura para separar regiões de duas ou mais células conexas entre si, ocasionadas por células aglomeradas na lâmina, foi considerada empiricamente no experimento 5 (Seção 5.5). Os resultados, contudo, mostram que a introdução desse passo não beneficiaria o método.

4.5 EXTRAÇÃO DE CARACTERÍSTICAS (SUB-ETAPA 5)

Esta sub-etapa recebe uma região e extrai um conjunto de características referentes ao seu formato (geométricas) e *pixels* internos (de textura). As características extraídas, que foram selecionadas da literatura (SARASWAT; ARYA, 2014), são listadas abaixo. Essas características são descritas formalmente na Seção 3.5.

Características referentes ao formato da região:

a) Área da região

b) Área do envoltório convexo da região

c) Solidez: razão entre os dois anteriores

d) Perímetro

e) Compacidade

f) Circularidade: razão entre a área da região e a do menor círculo que a sobrescreve

g) Retangularidade: razão entre a área da região e a do menor retângulo que a sobrescreve

h) Existência de Buraco: existência de contornos internos

Características referentes à intensidade dos *pixels* pertencentes à região:

a) Média

b) Variância

c) Desvio Padrão

Além disso, esta sub-etapa normaliza o valor dessas características pelo método de padronização (Seção 3.6.1.1). Dessa maneira, o modelo de classificação não aprende as magnitudes das características, apenas seus valores relativos.

4.6 CLASSIFICAÇÃO (SUB-ETAPA 6)

Esta sub-etapa utiliza de modelos de classificação para distinguir regiões referentes a células de interesse, diferenciando neutrófilos e linfócitos, e regiões que representam outras células na lâmina, outros artefatos ou ruído. Para tal, o vetor de características extraído na sub-etapa anterior (Seção 4.5) é alimentado a duas SVMs, cada das quais responsáveis por reconhecer um dos dois tipos de leucócitos de interesse. O experimento 6 (Seção 5.6) valida esta sub-etapa. Além disso, a utilização de SVMs na classificação de células também pode ser encontrada nos trabalhos de Gragnaniello, Sansone e Verdoliva (2016), Ponomarev et al. (2014) e Krishnan et al. (2012).

Este trabalho utiliza SVMs lineares probabilísticas: sua saída não é categórica, mas sim um valor contínuo que representa a probabilidade de uma região pertencer à classe sendo avaliada, dadas as características observadas e o conjunto de treinamento anteriormente estudado. O modelo de classificação desta etapa utiliza dessa característica para realizar uma classificação entre três classes (neutrófilos, linfócitos e "outros") utilizando apenas máquinas binárias: caso

umas ou ambas as máquinas retornem probabilidade superior a 50%, a classe de maior probabilidade é escolhida. Caso contrário, ambas estão informando que há mais de 50% de chance da região pertencer à classe "outros", e portanto essa classe é atribuída à região.

Os resultados desta etapa, aplicada sobre todas as regiões extraídas de uma imagem na etapa anterior (Seção 4.5) conclui o processo de contagem diferencial de leucócitos proposto neste trabalho.

5 EXPERIMENTOS E RESULTADOS

Durante o decorrer deste trabalho, foram conduzidos 6 experimentos com o propósito de confeccionar, parametrizar e validar o método empregado na contagem diferencial automática de leucócitos (Capítulo 4). Os resultados obtidos formam base para justificar os passos de processamento empregados no método. Cada um dos experimentos realizados é descrito nas seções a seguir. Os códigos utilizados estão inclusos nos apêndices.

5.1 EXPERIMENTO 1 - DELIMITAÇÃO DA LÂMINA

Este experimento abordou a sub-etapa de delimitação da lâmina (Seção 4.1). Para ela, foram variados:

a) os três canais RGB ou a escala de tons cinzas, de onde é extraído o histograma sobre o qual a limiarização pelo método de Otsu é aplicada;

b) a presença ou a ausência do passo de remoção de regiões espúrias;

c) a presença ou a ausência do passo de encaixe de círculo.

Para cada combinação dessas opções, a sub-etapa de delimitação da lâmina foi aplicada sobre as 30 imagens com a lâmina manualmente anotada (Seção 3.1.1.1). A métrica de performance utilizada foi a distância de Jaccard (Seção 3.3.4) para com esse padrão ouro.

Os resultados obtidos podem ser observados na Tabela 3. Cada valor exibido é a média das distâncias de Jaccard provenientes das 30 imagens. Como pode ser observado, a melhor performance é obtida com a aplicação da sub-etapa de delimitação da lâmina sobre o canal azul e com os passos de remoção de regiões espúrias e encaixe de círculo presentes.

5.2 EXPERIMENTO 2 - REMOÇÃO DE RUÍDOS

Este experimento abordou a remoção de ruídos efetuada na sub-etapa 2 (Seção 4.2). Para executar essa tarefa, foram considerados os usos de:

a) Filtros de média, de mediana e gaussiano.

b) *Kernels* de tamanho 3, 5, 7 e 9 para filtragem.

Tabela 3 – Resultados experimentais da delimitação da lâmina pelo método de Otsu.

Canal	Remoção de regiões espúrias	Encaixe de círculo	Distância de Jaccard média
Azul	Sim	Sim	0.00536315
		Não	0.00655822
	Não	Não*	0.00673332
Verde	Sim	Sim	0.01116086
		Não	0.01742746
	Não	Não*	0.05477111
Vermelho	Sim	Sim	0.01353275
		Não	0.01696285
	Não	Não*	0.02164782
Cinza	Sim	Sim	0.01048615
		Não	0.01502503
	Não	Não*	0.02886523

*O passo de encaixe de círculo assume que exista apenas um componente conexo e portanto precisa da presença do passo de remoção de regiões espúrias para ser viável.

Para cada combinação dessas opções, a sub-etapa de remoção de ruídos foi aplicada sobre os *patchs* das 30 imagens com os contornos das células manualmente anotados (Seção 3.1.1.4).

Ruídos consistem em diferenças bruscas na intensidade em um único pixel, usualmente com intensidade mais pertinente à região em que ele não está inserido. Por conta disso, ruídos resultam em regiões espúrias quando passam por uma limiarização de histograma, o que prejudica qualquer segmentação dessa vertente. A qualidade da melhor segmentação alcançável por um processo de limiarização de histograma para uma imagem, portanto, é correlacionada com a presença de ruídos nela.

Com base nessa afirmação, este experimento aplicou cada limiar possível sobre cada histograma de uma imagem (três canais RGB e escala de tons de cinza) e comparou a imagem binária resultante com o seu padrão ouro de segmentação. A métrica utilizada foi a distância de Jaccard (Seção 3.3.4), cujo valor mínimo configura a melhor qualidade de segmentação referente a cada imagem. A média desses valores entre as 30 imagens é, então, assumida como indicadora de qualidade da remoção de ruídos executada pelo filtro.

Os resultados são exibidos na Figura 28. É possível observar a superioridade do uso do filtro de mediana.

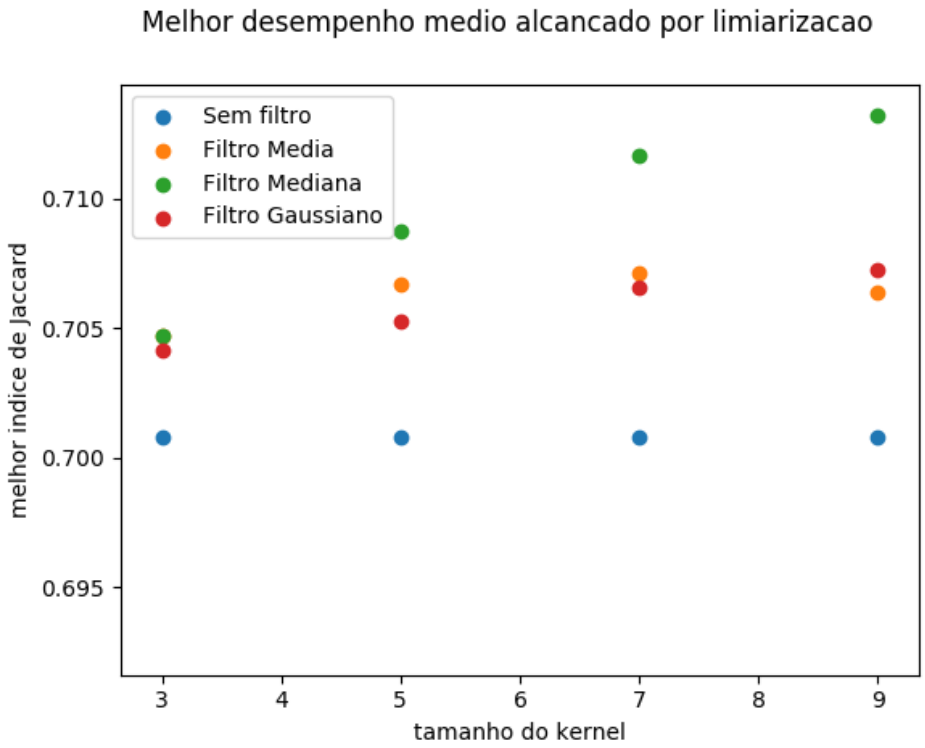


Figura 28 – Resultados experimentais do uso de diversos filtros removedores de ruído. O eixo vertical compreende a média de, para cada imagem, o melhor índice de Jaccard de alguma imagem binária alcançável por ela caso submetida a um processo de limiarização.

5.3 EXPERIMENTO 3 - LIMIARIZAÇÃO

Este experimento diz respeito à sub-etapa de limiarização (Seção 4.3) e atuou sobre imagens já filtradas para remoção de ruídos por um filtro de mediana de tamanho 9x9, como determinado ótimo no experimento 2 (Seção 5.2). O objetivo foi determinar qual histograma da imagem (um dos três canais RGB ou escala de tons de cinza) é mais apropriado para limiarização.

Para isso, este experimento submeteu as 30 imagens com a segmentação manualmente anotada (Seção 3.1.1.4) a uma limiarização, variando:

a) O histograma de aplicação, três canais RGB ou escala de tons de cinza;

b) O valor do limiar, considerando todos os possíveis.

Depois de limiarizadas, as imagens foram comparadas com o padrão ouro da segmentação, usando como métrica a distância de Jaccard (Seção 3.3.4). A Figura 29 exibe o índice de Jaccard médio entre as 30 imagens, para cada valor de limiar. O gráfico mostra que o canal RGB verde é mais apropriado para a segmentação de leucócitos por limiarização.

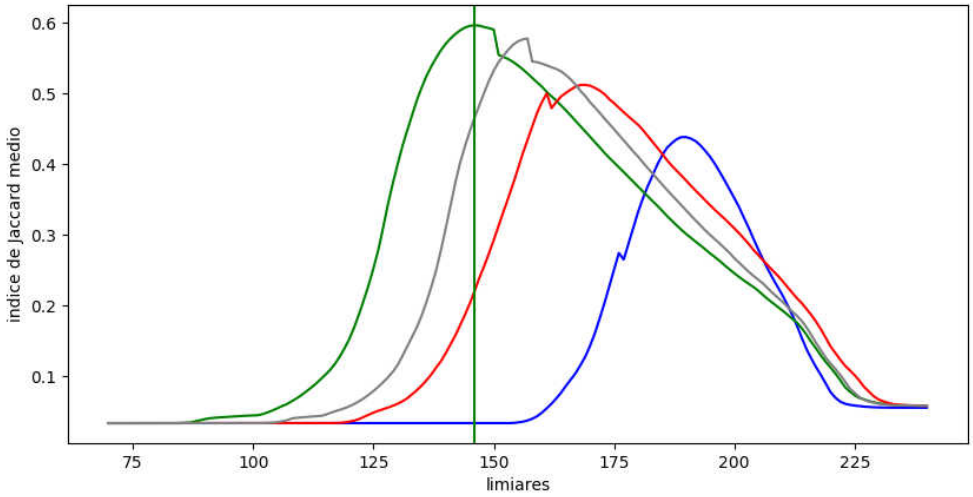


Figura 29 – Resultados experimentais da limiarização sobre cada histograma: os três canais RGB e a escala de tons de cinza. O eixo vertical compreende a média dos índices de Jaccard para com o padrão ouro obtidos em cada imagem limiarizada. O máximo está localizado no limiar 146, para o canal verde.

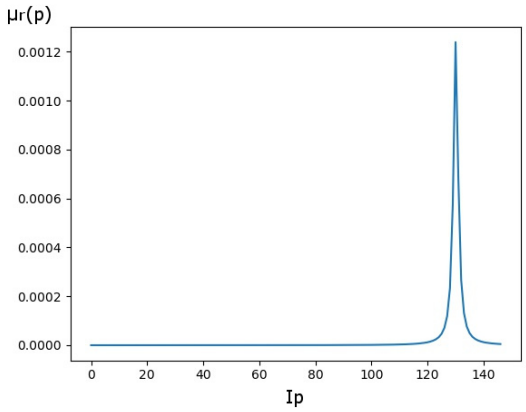
5.4 EXPERIMENTO 4 - DIVERGÊNCIA FUZZY

Este experimento considera o uso da LMDF (Seção 3.3.1.2) para determinar um limiar a ser utilizado sobre cada imagem na etapa de limiarização (Seção 4.3). Essa abordagem tem base no artigo de Ghosh et al. (2010).

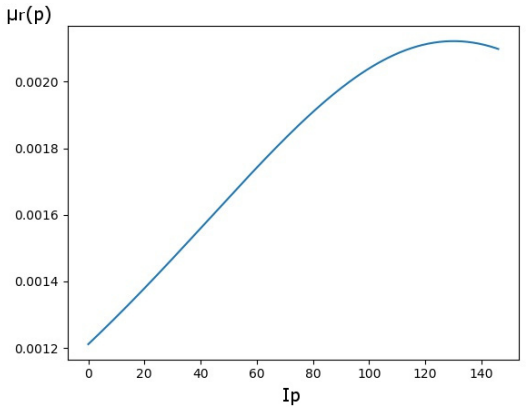
Ghosh et al. (2010) estipula o *kernel* da LMDF, a função de pertinência de um *pixel* p à região r, como apresentado na Equação 33. É utilizada a distribuição de Cauchy com parâmetro de escala $\gamma = 1$, multiplicada por um fator de normalização $c = 1/(I_{max} - I_{min}) = 1/(255 - 0)$, sobre a imagem em tons de cinza.

$$\mu_r(p) = c.f(I_p; \overline{I_r}, 1) = \frac{c}{\pi}\left[\frac{1}{1 + (I_p - \overline{I_r})^2}\right] \tag{33}$$

O gráfico da Figura 30a exibe um exemplo dessa função de pertinência. A função consiste em um pico ingrime em torno da média e pertinência zero para os demais valores de intensidade. No exemplo, $\mu_r(p)$ já converge para zero a partir da intensidade 120, apenas 10 valores de intensidade distante da média da região. É intuitivo que uma transição mais gradual pode ser de interesse. Isso, por sua vez, pode ser alcançado com uma modificação no parâmetro de escala γ (Figura 30b).



(a) $c = 1/255, \gamma = 1$



(b) $c = 1, \gamma = 150$

Figura 30

Com isso em mente, foi considerado empiricamente o uso de parâmetros diferentes dos definidos em Ghosh et al. (2010). Foram variados:

a) O fator de normalização c;

b) O parâmetro de escala γ;

c) O histograma de aplicação: três canais RGB ou escala de tons de cinza.

A função de pertinência, portanto, assume a forma mais genérica

$$\mu_r(p; c,\gamma) = \frac{c}{\pi\gamma}\left[\frac{\gamma^2}{\gamma^2 + (I_p - \overline{I_r})^2}\right] \tag{34}$$

Para este experimento, foram utilizados os *patchs* das 30 imagens com os contornos das células anotados (Seção 3.1.1.4), já submetidos às sub-etapas 1 e 2 do método (Seções 4.1 e 4.2). Os resultados de performance da aplicação da LMDF na sub-etapa de limiarização, variando esses parâmetros, são apresentados na Figura 31.

Como pode ser observado, os melhores resultados foram obtidos com a aplicação da LMDF sobre o canal RGB verde (como decidido ótimo no experimento 3, Seção 5.3) e sobre a escala de tons cinza (como feito em Ghosh et al. (2010)). A modificação dos parâmetros c e γ melhorou a performance do método. Contudo, essa melhora não foi suficiente. Foi obtida melhor qualidade de segmentação com o uso de um limiar fixo, como descrito no experimento 3 (Seção 5.3).

Uma segunda comparação entre os parâmetros definidos em Ghosh et al. (2010), o uso de uma função de pertinência de transição mais gradual com a alteração do parâmetro de escala γ e os resultados ótimos obtidos exaustivamente no experimento 3 é apresentada na Figura 32.

5.5 EXPERIMENTO 5 - OPERADOR MORFOLÓGICO DE ABERTURA

Este experimento considerou a introdução de um passo na sub-etapa de extração de regiões que aplicasse um operador mor,fológico de abertura (Seção 3.3.2.3). O propósito foi o de desconectar duas ou mais células que, por estarem aglomeradas na imagem, produziram um único componente conexo na sub-etapa de limiarização (Seção 4.3). Um exemplo desse cenário pode ser observado na Figura 33a.

A respeito do *kernel* do operador morfológico de abertura, foram variados:

a) O formato: de cruz, retangular ou elíptico.

b) O tamanho: 3x3, 5x5 ou 7x7 *pixels*.

Este experimento foi aplicado sobre as 30 imagens com o contorno das células manualmente anotado (Seção 3.1.1.1). Inicialmente, as imagens foram submetidas às sub-etapas 1, 2 e 3 do método (Seções 4.1 a 4.3). Depois, os contornos dos componentes conexos foram encontrados pelo método de seguimento de borda (Seção 3.3.3.1).

A respeito da hierarquia de contornos construída pelo método de seguimento de borda (Figura 33b), é esperado que os contornos pertencentes ao primeiro nível hierárquico sejam os contornos externos das células e que os contornos do segundo nível hierárquico delineiem seus buracos internos.

Como o propósito da aplicação do operador de abertura é o de separar células, e não o de quebrar porções finas da célula entre o contorno externo e um buraco, possivelmente aumentando o grau de desconexidade, apenas os componentes conexos definidos pelos contornos do primeiro nível hierárquico foram submetidos ao operador (Figuras 33c a 33e). Dessa forma, a erosão não é aplicada a partir de buracos internos às células e são evitadas ocorrências de quebra da região, como é o caso da imagem apresentada na Figura 33f.

Aplicada essa operação, os componentes conexos foram re-computados.

A métrica de avaliação, por este experimento lidar com a conexidade das regiões, difere da utilizada nos experimentos anteriores. Não mais basta que os *pixels* pertencentes a célula, de acordo com o padrão ouro, estejam também presentes no *foreground* da segmentação predita. É necessário, também, que esses *pixels* sejam conexos entre si (i.e. componham um componente conexo). Além disso, é desejado que eles sejam conexos com a menor quantidade possível de *pixels* não pertencentes à região definida no padrão ouro.

Para refletir essas duas restrições sobre a avaliação de uma segmentação, foi utilizado o BF *matching* (Seção 3.4). As comparações nele efetuadas ocorreram entre as regiões de cada

uma das células anotadas no padrão ouro e cada um dos componentes conexos presentes na segmentação sendo avaliada.

Para cada célula, foi atribuída a menor distância de Jaccard (Seção 3.3.4) encontrada, referente ao componente conexo que melhor a representa. Esse processo é ilustrado na Figura 34. A qualidade da segmentação, por fim, é representada pela média das distâncias mínimas encontradas para cada célula. Dessa maneira, a métrica de avaliação é mais rigorosa no que se refere à conexidade das regiões do que se a aplicação da distância de Jaccard fosse feita sobre a imagem toda.

Apesar dos argumentos apresentados, os resultados deste experimento (Tabela 4) mostram que a utilização do operador morfológico de abertura é prejudicial para o método. Isso significa que, independente do *kernel* utilizado, ocorreram mais casos em que o operador de abertura aumenta o grau de desconexidade da região referente a uma célula do que casos em que duas células anteriormente conectadas tornam-se desconexas.

Tabela 4 – Resultados experimentais da aplicação da abertura morfológica sobre a imagem binária limiarizada.

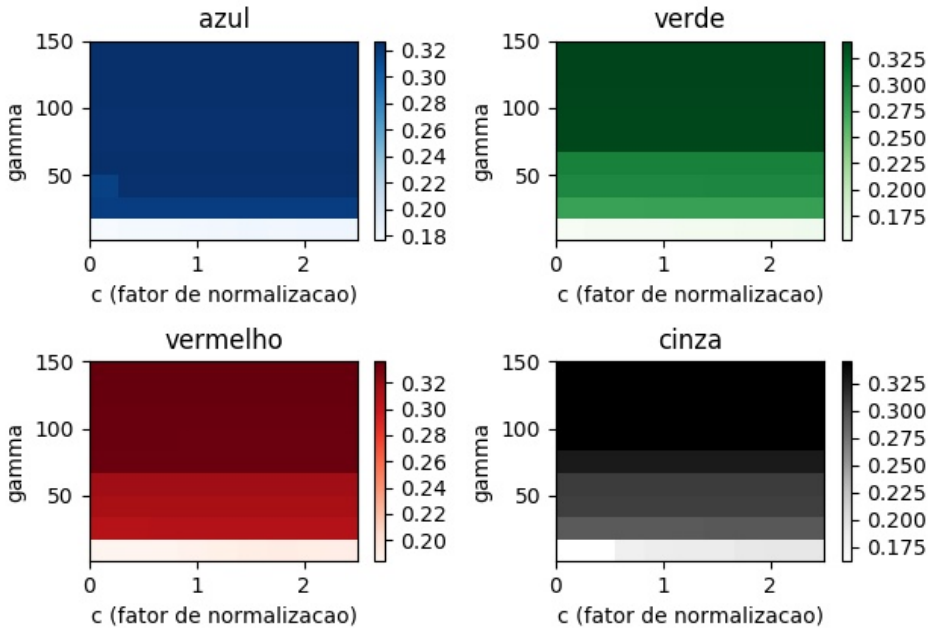
Kernel da abertura morfológica		Índice de Jaccard entre componentes conexos mais similares médio
Formato	Tamanho	
Sem abertura		0.62430109
Cruz	3x3	0.62335596
	5x5	0.62218226
	7x7	0.62028554
Retângulo	3x3	0.62311182
	5x5	0.62056393
	7x7	0.6129102
Elipse	3x3	0.62335596
	5x5	0.62280741
	7x7	0.62151954

5.6 EXPERIMENTO 6

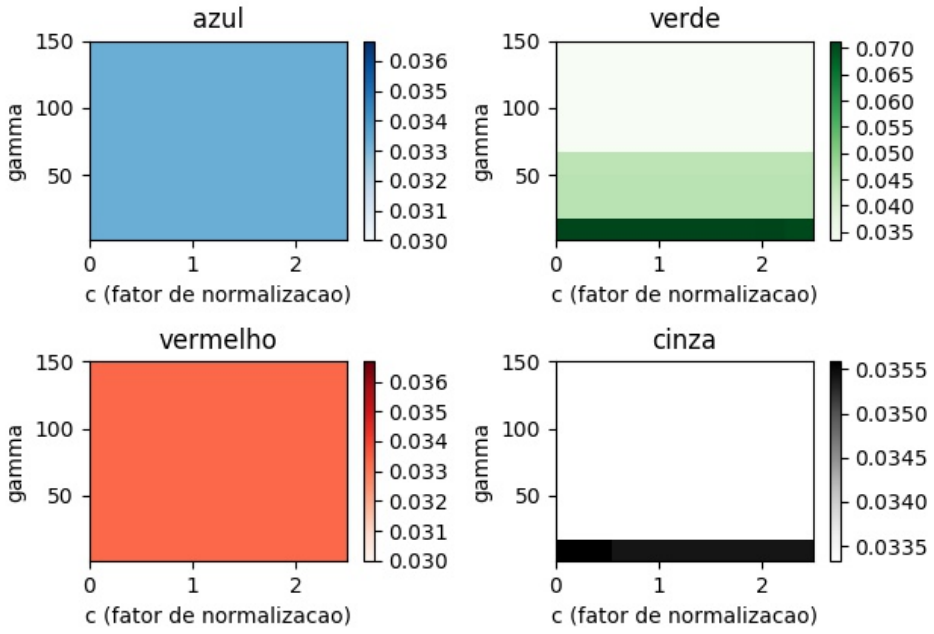
Este experimento asserta a capacidade da etapa de reconhecimento (Seções 4.5 e 4.6) de discernir os objetos de interesse, neutrófilos e linfócitos, providas regiões que os representem.

Para tal, foi utilizado o esquema de validação K-Fold (Seção 3.6.3) para particionar, em 4 iterações, as regiões de rótulo anotado (Seção 3.1.1.3) em um conjunto de treinamento e um de teste. Os resultados do classificador foram comparados com as anotações e analisados por curvas de ROC (Seção 3.6.2), que podem ser observadas na Figura 35.

Este experimento considerou apenas as regiões conexas com uma relação um para um com uma célula, conforme descrito na Seção 3.1.1.3.



(a) com o limiar que minimiza a divergência *fuzzy*



(b) com a metade do limiar que minimiza a divergência *fuzzy*

Figura 31 – Performance da limiarização por minimização da divergência *fuzzy* para com uma imagem idealmente segmentada, variando os parâmetros c (fator de normalização) e γ da Equação 34. O processo foi aplicado sobre os três canais RGB e sobre a escala de tons de cinza. É apresentado o índice de Jaccard (Seção 3.3.4) médio da segmentação obtida nesse processo comparada com o padrão ouro. A Figura a apresenta os resultados da limiarização sobre o limiar que minimiza a divergência *fuzzy*, enquanto na Figura b é utilizado metade do valor do limiar, como sugerido em Ghosh et al. (2010).

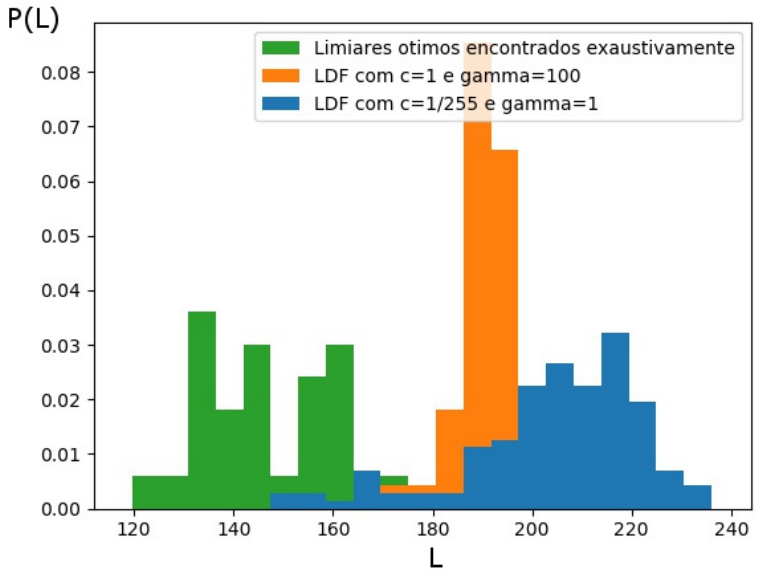
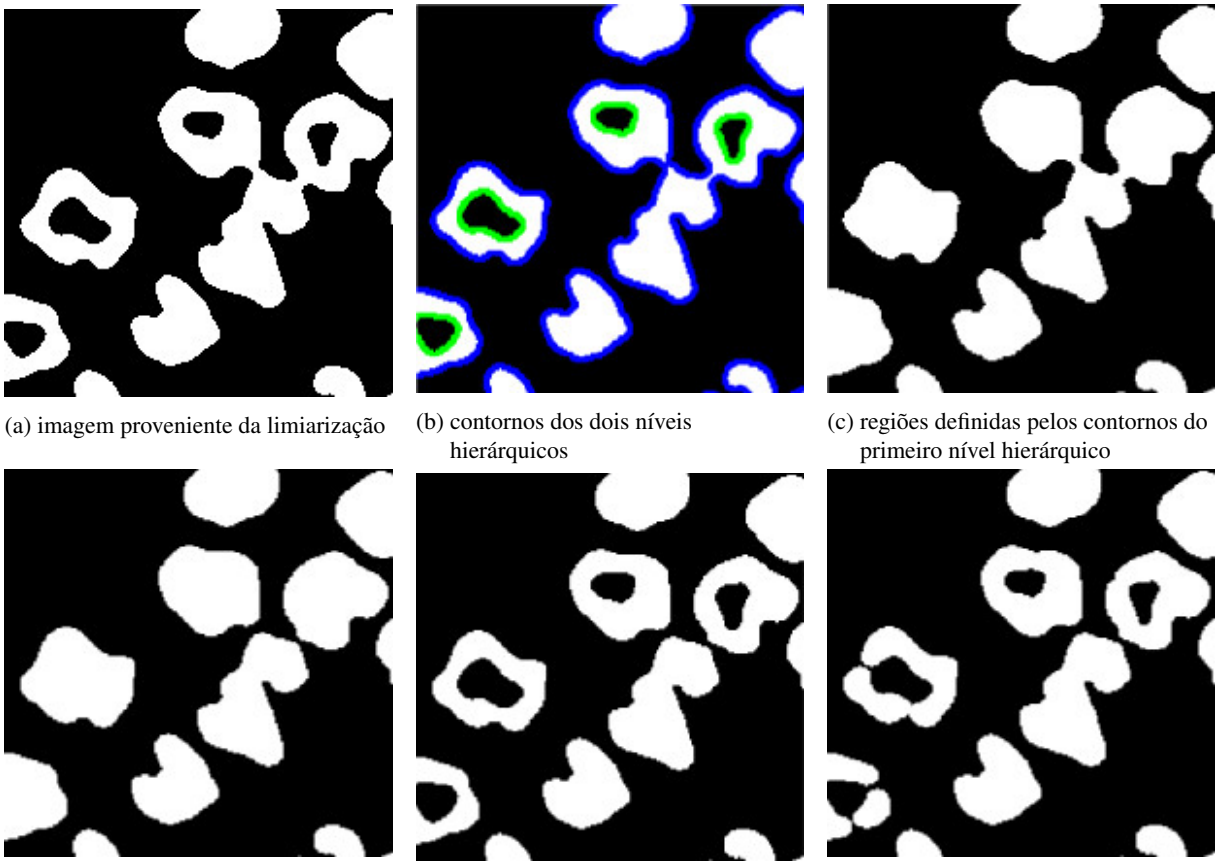


Figura 32 – Distribuição de probabilidade dos limiares ótimos encontrados exaustivamente e dos limiares provenientes da LMDF (Seção 3.3.1.2). São apresentados os resultados de aplicação da LMDF com parâmetros (c, γ) iguais a $(1/255, 1)$, como definido em (GHOSH et al., 2010), e iguais a $(1, 100)$.



(a) imagem proveniente da limiarização

(b) contornos dos dois níveis hierárquicos

(c) regiões definidas pelos contornos do primeiro nível hierárquico

(d) abertura aplicada sobre a imagem da Figura c

(e) buracos reintroduzidos

(f) abertura aplicada diretamente sobre a imagem da Figura a

Figura 33 – Aplicação do operador morfológico de abertura sobre a imagem binária (Figura a) proveniente da sub-etapa de limiarização (Seção 4.3). Os contornos das regiões (Figura b) são encontrados pelo método de seguimento de borda (Seção 3.3.3.1). Sobre as regiões definidas pelos contornos do primeiro nível hierárquico (Figura c), o operador é aplicado (Figura d). Então, os buracos definidos pelos contornos do segundo nível hierárquico são reintroduzidos (Figura e). Em comparação, a Figura f ilustra a aplicação direta do operador.

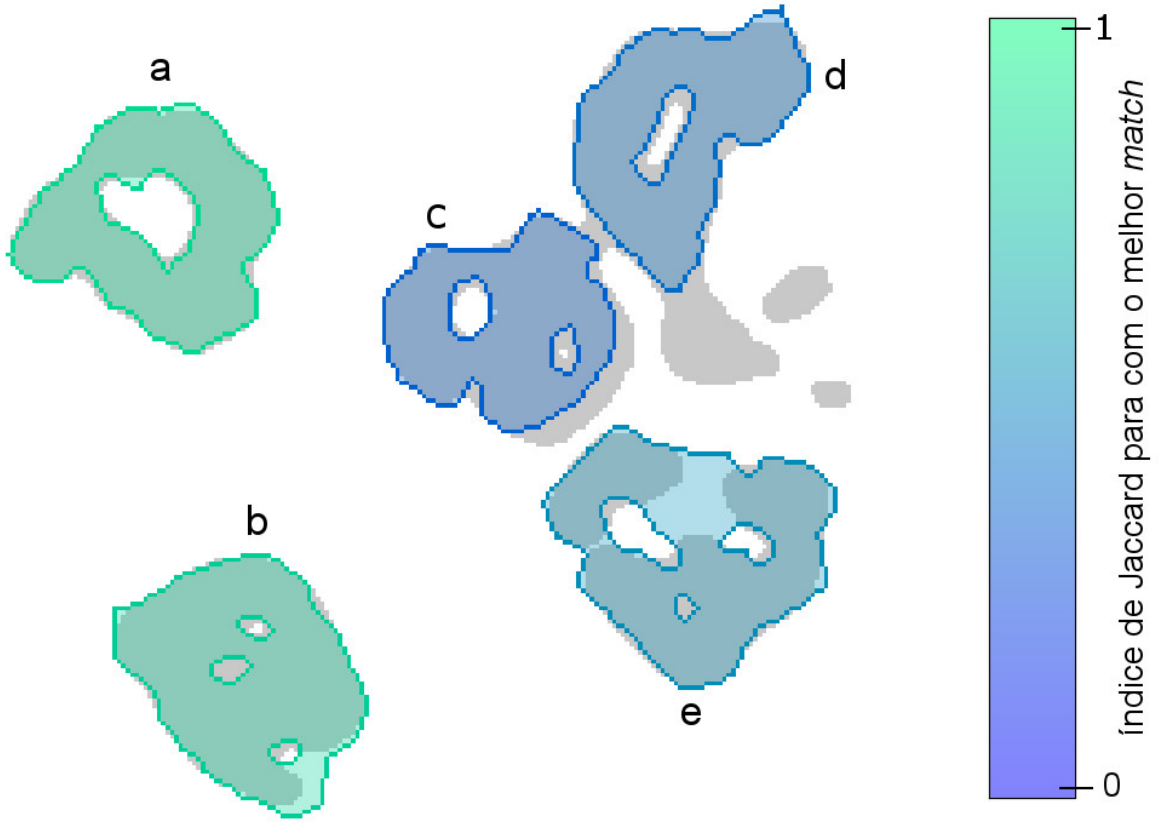
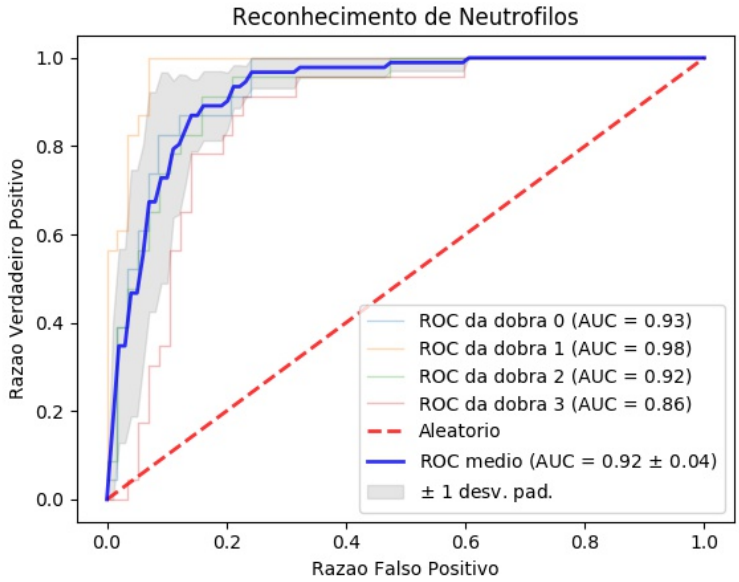
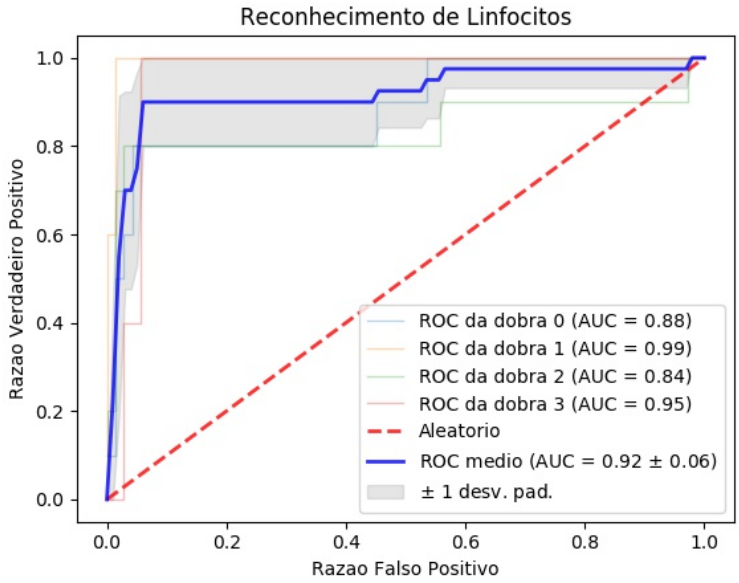


Figura 34 – Exemplo da avaliação de uma tentativa de segmentação com o método BF *matching* (Seção 3.4), usando a distância de Jaccard (Seção 3.3.4) como *kernel*. O *foreground* da tentativa de segmentação está apresentado em cinza. A coloração das células indica seu nível de correspondência para com o componente conexo do *foreground* mais próximo, de acordo com a distância de Jaccard. Como pode ser observado, as células a e b tem boa correspondência para com as suas respectivas regiões. As regiões predestinadas a representar as células c e d, contudo, estão conectadas. Com isso, cada uma dessas células foi pareada com a região que envolve ambas, atingindo baixa correspondência. A região pertinente à célula e, por sua vez, é desconexa. Essa célula, portanto, foi pareada apenas com o maior dos dois componentes conexos, o que penalizou o seu índice de Jaccard.



(a) acurácia no reconhecimento de neutrófilos



(b) acurácia no reconhecimento de linfócitos

Figura 35 – Curvas ROC do reconhecimento de neutrófilos e linfócitos.

6 DISCUSSÃO

Este trabalho originou-se no primeiro semestre de 2017. Desde então, seu objetivo manteve-se o mesmo. Contudo, o método inicialmente proposto, confeccionado unicamente com base na literatura, sofreu modificações.

Originalmente, com base no trabalho de segmentação de leucócitos de Ghosh et al. (2010), a sub-etapa de limiarização (Seção 4.3) seria realizada por LMDF (Seção 3.3.1.2). Contudo, foram obtidos, por meio do experimento 4 (Seção 5.4), resultados inferiores aos esperados, considerando-se o precedente levantado por Ghosh et al. (2010). Levantam-se duas hipóteses para justificar esse acontecimento, referentes a duas diferenças entre este trabalho e o de Ghosh et al. (2010):

a) O pré processamento da imagem: Ghosh et al. (2010) aplica uma normalização por transferência de cores (REINHARD et al., 2001), enquanto este trabalho usa apenas o filtro de mediana (Seção 3.2.2) para remoção de ruídos.

b) A tintura de coloração: Ghosh et al. (2010) utiliza a tintura de Leishman, enquanto neste trabalho foi utilizada a hematoxilina-e-eosina.

Para determinar quais dessas hipóteses é responsável pela diferença na qualidade dos resultados, faz-se necessária investigação futura.

Como tentativa de melhorar o resultado da segmentação obtido anteriormente, foi considerado o uso de um operador morfológico de abertura para resolver um problema frequente na segmentação: células aglomeradas produzindo um único componente conexo. Essa possibilidade foi estudada no experimento 5.5. A introdução dessa sub-etapa, contudo, mostrou-se prejudicial para o método.

Com base nesses dois cenários, pode-se dizer que este trabalho não obteve bons resultados referentes à segmentação, mas provocou indagação e levantou questionamentos referentes a sua deficiência nessa etapa, que podem vir a alimentar trabalhos futuros.

Para que o trabalho pudesse prosseguir para a etapa seguinte, de reconhecimento (Seções 4.4 e 4.6), a etapa de limiarização foi parametrizada por um limiar fixo (Seção 4.3), computado exaustivamente para otimizar a proximidade da segmentação com o padrão ouro das imagens da própria base. Também, para suprir a deficiência da segmentação referente à conexidade das regiões, cenários em que duas regiões são extraídas do interior da mesma célula ou uma única região engloba duas ou mais células foram manualmente isolados. O propósito dessas

manipulações é o de que a etapa de reconhecimento possa ser estudada e validade para uso futuro, quando as deficiências do processo de segmentação forem supridas por mais estudos.

Aparte dos pontos anteriormente mencionados, as etapas de pré-processamento e reconhecimento obtiveram bons resultados (experimentos 1, 2 e 6, Capítulo 5), coerentes com a literatura.

REFERÊNCIAS

ADAGALE, S. S.; PAWAR, S. S. Image segmentation using pcnn and template matching for blood cell counting. In: **2013 IEEE International Conference on Computational Intelligence and Computing Research**. [S.l.: s.n.], 2013. p. 1–5.

CHAIRA, T.; RAY, A. Segmentation using fuzzy divergence. **Pattern Recognition Letters**, v. 24, n. 12, p. 1837 – 1844, 2003. ISSN 0167-8655.

DIGITAL Image Processing. Segmentation: Edge-based segmentation. 2003. [Online; accessado em 08 de novembro de 2017]. Disponível em: <http://user.engineering.uiowa.edu/~dip/lecture/Segmentation2.html>.

DORIGO, T. **The Puzzling Weighted Average**. 2011. [Online; accessado em 24 de maio de 2017]. Disponível em: <http://www.science20.com/quantum_diaries_survivor/puzzling_weighted_average-84844>.

EDINBURGH, S. of Informatics of the University of. **Spatial domain methods**. 1997. [Online; accessado em 24 de maio de 2017]. Disponível em: <http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT5/node3.html>.

FARIDI, P. et al. An automatic system for cell nuclei pleomorphism segmentation in histopathological images of breast cancer. In: **2016 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)**. [S.l.: s.n.], 2016. p. 1–5.

FAWCETT, T. An introduction to roc analysis. **Pattern Recogn. Lett.**, Elsevier Science Inc., New York, NY, USA, v. 27, n. 8, p. 861–874, jun. 2006. ISSN 0167-8655.

GE, J. et al. A system for automated counting of fetal and maternal red blood cells in clinical kb test. In: **2014 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2014. p. 1706–1711. ISSN 1050-4729.

GHARIPOUR, A.; LIEW, A. W. C. Fuzzy clustering using local and global region information for cell image segmentation. In: **2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)**. [S.l.: s.n.], 2014. p. 216–222. ISSN 1098-7584.

GHOSH, M. et al. Automated leukocyte recognition using fuzzy divergence. **Micron**, v. 41, n. 7, p. 840 – 846, 2010. ISSN 0968-4328.

GHOSH, P.; BHATTACHARJEE, D.; NASIPURI, M. Blood smear analyzer for white blood cell counting: A hybrid microscopic image analyzing technique. **Applied Soft Computing**, v. 46, p. 629 – 638, 2016. ISSN 1568-4946.

GIMP. **GNU Image Manipulation Program**. 2017. [Online; accessado em 30 de outubro de

2017]. Disponível em: <https://www.gimp.org/>.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing (2nd Edition)**. [S.l.]: Pearson, 2002.

GRAGNANIELLO, D.; SANSONE, C.; VERDOLIVA, L. Cell image classification by a scale and rotation invariant dense local descriptor. **Pattern Recognition Letters**, v. 82, Part 1, p. 72 – 78, 2016. ISSN 0167-8655. Pattern recognition Techniques for Indirect Immunofluorescence Images Analysis.

GUO, X.; YU, F. A method of automatic cell counting based on microscopic image. In: **2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics**. [S.l.: s.n.], 2013. v. 1, p. 293–296.

HAMID, R. et al. Automated detection and counting of pus cells on sputum images. In: **2013 International Conference on Electronics, Computer and Computation (ICECCO)**. [S.l.: s.n.], 2013. p. 5–8.

HASAN, Y. M.; KARAM, L. J. Morphological reversible contour representation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE Computer Society, Los Alamitos, CA, USA, v. 22, p. 227–240, 2000. ISSN 0162-8828.

III, T. H. B.; ZHOU, Y. Hiv-associated guillain–barré syndrome. **Journal of the Neurological Sciences**, v. 208, n. 1–2, p. 39 – 42, 2003. ISSN 0022-510X.

INVASIVA, R. B. de C. **Impacto do escore SYNTAX na estratificação de risco após intervenção coronária percutânea em pacientes não-selecionados**. 2012. [Online; accessado em 21 de maio de 2017]. Disponível em: <http://oldarchive.rbci.org.br/detalhe_artigo.asp?id=605>.

JACCARD, P. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. **Bulletin del la Société Vaudoise des Sciences Naturelles**, v. 37, p. 547–579, 1901.

KLEMEN. **gaussian filter**. 2014. [Online; accessado em 24 de maio de 2017]. Disponível em: <https://forums.ni.com/t5/Machine-Vision/gaussian-filter/td-p/2441104/page/3>.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: **Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995. (IJCAI'95), p. 1137–1143. ISBN 1-55860-363-8.

KOTHARI, S.; CHAUDRY, Q.; WANG, M. D. Automated cell counting and cluster segmentation using concavity detection and ellipse fitting techniques. In: **2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro**. [S.l.: s.n.], 2009. p. 795–798. ISSN 1945-7928.

KRISHNAN, M. M. R. et al. Hybrid segmentation, characterization and classification of basal cell nuclei from histopathological images of normal oral mucosa and oral submucous fibrosis. **Expert Systems with Applications**, v. 39, n. 1, p. 1062 – 1077, 2012. ISSN 0957-4174.

LABRY, L. O. D. et al. White blood cell count as a predictor of mortality: Results over 18 years from the normative aging study. **Journal of Clinical Epidemiology**, v. 43, n. 2, p. 153 – 157, 1990. ISSN 0895-4356.

LEE, H.; CHEN, Y.-P. P. Cell morphology based classification for red cells in blood smear images. **Pattern Recognition Letters**, v. 49, p. 155 – 161, 2014. ISSN 0167-8655.

LIAO, M. et al. Automatic segmentation for cell images based on bottleneck detection and ellipse fitting. **Neurocomputing**, v. 173, Part 3, p. 615 – 622, 2016. ISSN 0925-2312.

LORENA, A. C.; CARVALHO, A. C. P. L. F. de. Uma introdução às support vector machines. In: **Revista de Informática Teórica e Aplicada**. [S.l.: s.n.], 2007. v. 14, n. 2.

MAO-JUN, S. et al. A new method for blood cell image segmentation and counting based on pcnn and autowave. In: **2008 3rd International Symposium on Communications, Control and Signal Processing**. [S.l.: s.n.], 2008. p. 6–9.

MELO, G. de et al. A robust segmentation method for counting bovine milk somatic cells in microscope slide images. **Computers and Electronics in Agriculture**, v. 115, p. 142 – 149, 2015. ISSN 0168-1699.

NASCIMENTO, R. F. F. et al. O algoritmo support vector machines (svm): avaliação da separação ótima de classes em imagens ccd-cbers-2. In: **Anais XIV Simpósio Brasileiro de Sensoriamento Remoto**. [S.l.: s.n.], 2009. p. 2079–2086.

OPENCV Contour Features. 2017. [Online; accessado em 30 de novembro de 2017]. Disponível em: <https://docs.opencv.org/3.3.1/dd/d49/tutorial_py_contour_features.html>.

OTSU, N. A threshold selection method from gray-level histograms. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 9, n. 1, p. 62–66, Jan 1979. ISSN 0018-9472.

PEDRINI, H.; SCHWARTZ, W. R. **Análise de Imagens Digitais - Princípios, algoritmos e aplicações**. [S.l.]: Thomson, 2008.

PICCININI, F. et al. Improving reliability of live/dead cell counting through automated image mosaicing. **Computer Methods and Programs in Biomedicine**, v. 117, n. 3, p. 448 – 463, 2014. ISSN 0169-2607.

PONOMAREV, G. V. et al. {ANA} hep-2 cells image classification using number, size, shape and localization of targeted cell regions. **Pattern Recognition**, v. 47, n. 7, p. 2360 – 2366, 2014. ISSN 0031-3203.

RAMIN, M. et al. Counting the number of cells in immunocytochemical images using genetic algorithm. In: **2012 12th International Conference on Hybrid Intelligent Systems (HIS)**. [S.l.: s.n.], 2012. p. 185–190.

RASHID, N. Z. N.; MASHOR, M. Y.; HASSAN, R. Unsupervised color image segmentation of red blood cell for thalassemia disease. In: **2015 2nd International Conference on Biomedical Engineering (ICoBE)**. [S.l.: s.n.], 2015. p. 1–6.

REINHARD, E. et al. Color transfer between images. **IEEE Computer Graphics and Applications**, v. 21, n. 5, p. 34–41, Sep 2001. ISSN 0272-1716.

REYES, L. E. H.; ROZO, L. X. B.; MORALES, F. A. R. Automatic leukocyte image segmentation: A review. In: **2015 20th Symposium on Signal Processing, Images and Computer Vision (STSIVA)**. [S.l.: s.n.], 2015. p. 1–9.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 2. ed. [S.l.]: Pearson Education, 2003. ISBN 0137903952.

SARASWAT, M.; ARYA, K. Automated microscopic image analysis for leukocytes identification: A survey. **Micron**, v. 65, p. 20 – 33, 2014. ISSN 0968-4328.

SARRAFZADEH, O. et al. Circlet based framework for red blood cells segmentation and counting. In: **2015 IEEE Workshop on Signal Processing Systems (SiPS)**. [S.l.: s.n.], 2015. p. 1–6.

____. Analyzing features by {SWLDA} for the classification of hep-2 cell images using {GMM}. **Pattern Recognition Letters**, v. 82, Part 1, p. 44 – 55, 2016. ISSN 0167-8655. Pattern recognition Techniques for Indirect Immunofluorescence Images Analysis.

SUZUKI, S.; ABE, K. Topological structural analysis of digitized binary images by border following. **Computer Vision, Graphics, and Image Processing**, v. 30, n. 1, p. 32–46, 1985. Disponível em: <http://dblp.uni-trier.de/db/journals/cvgip/cvgip30.html#SuzukiA85>.

TAHA, A. A.; HANBURY, A. Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool. **BMC Medical Imaging**, v. 15, n. 1, p. 29, Aug 2015. ISSN 1471-2342. Disponível em: <https://doi.org/10.1186/s12880-015-0068-x>.

TANGSUKSANT, W. et al. Development algorithm to count blood cells in urine sediment using ann and hough transform. In: **The 6th 2013 Biomedical Engineering International Conference**. [S.l.: s.n.], 2013. p. 1–4.

TAREEF, A. et al. Automatic segmentation of overlapping cervical smear cells based on local distinctive features and guided shape deformation. **Neurocomputing**, v. 221, p. 94 – 107, 2017. ISSN 0925-2312.

TOMARI, R. et al. Computer aided system for red blood cell classification in blood smear

image. **Procedia Computer Science**, v. 42, p. 206 – 213, 2014. ISSN 1877-0509.

VAPNIK, V. N. An overview of statistical learning theory. **IEEE Transactions on Neural Networks**, v. 10, n. 5, p. 988–999, Sep 1999. ISSN 1045-9227.

WANG, P. et al. Automatic cell nuclei segmentation and classification of breast cancer histopathology images. **Signal Processing**, v. 122, p. 1 – 13, 2016. ISSN 0165-1684.

WHATWHENHOW. **BLOB Analysis (Introduction to Video and Image Processing) Part 1**. 2017. [Online; accessado em 18 de maio de 2017]. Disponível em: <http://what-when-how.com/introduction-to-video-and-image-processing/blob-analysis-introduction-to-video-and-image-processing-part-1/>.

WILIEM, A. et al. Automatic classification of human epithelial type 2 cell indirect immunofluorescence images using cell pyramid matching. **Pattern Recognition**, v. 47, n. 7, p. 2315 – 2324, 2014. ISSN 0031-3203.

YOU, Z. et al. Automated cell individualization and counting in cerebral microscopic images. In: **2016 IEEE International Conference on Image Processing (ICIP)**. [S.l.: s.n.], 2016. p. 3389–3393.

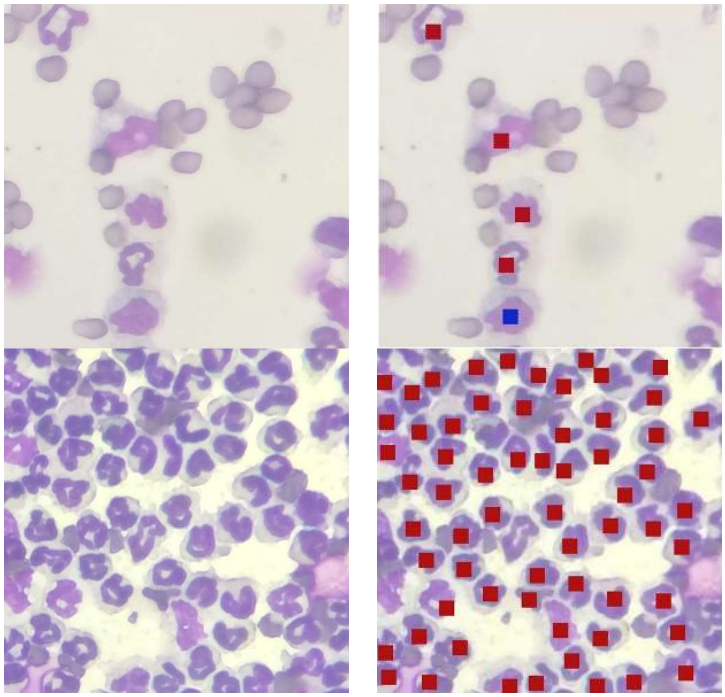
APÊNDICE A – ANOTAÇÕES PELO ESPECIALISTA

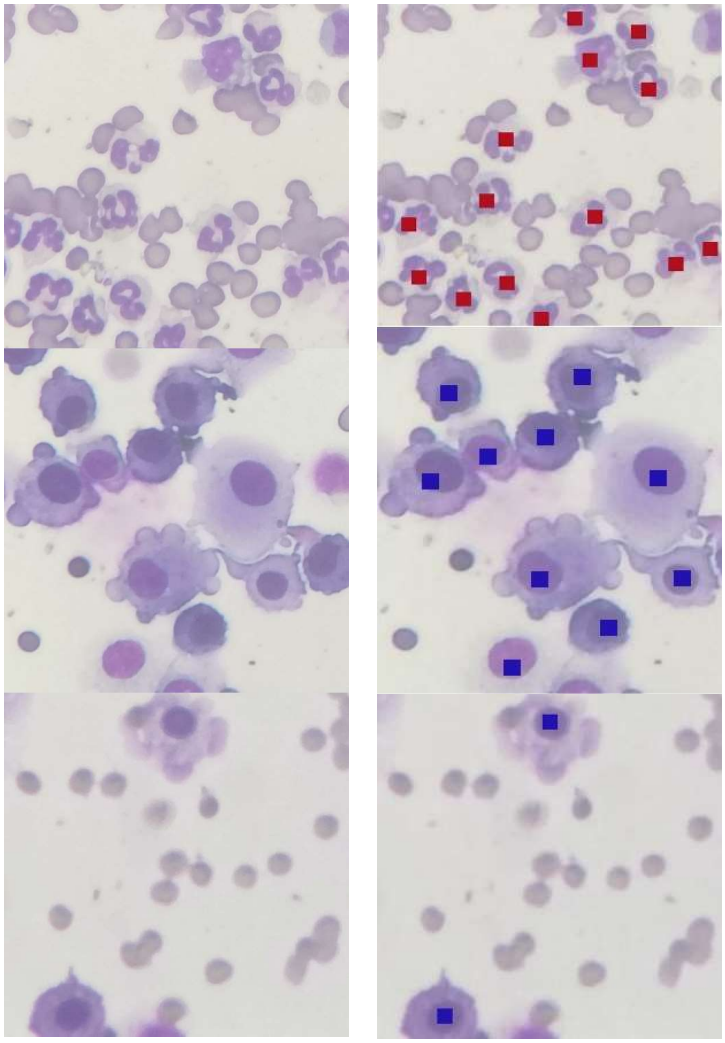
Anotações pelo Especialista

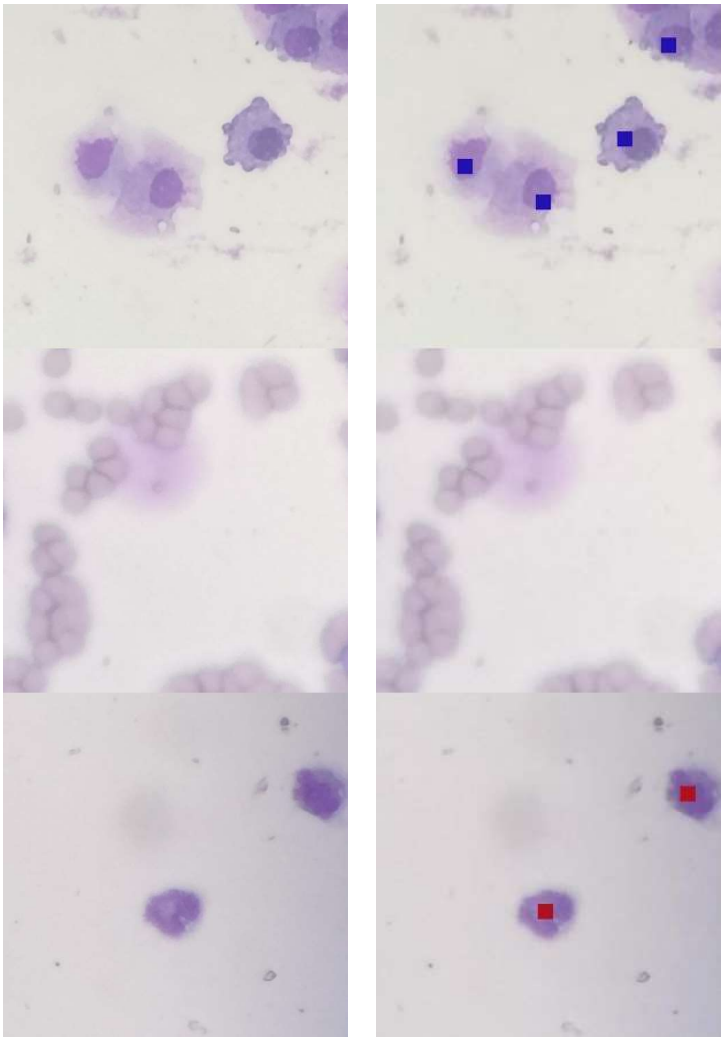
Este documento contempla 16 *patchs* de tamanho 500x500 *pixels*, cada um posicionado aleatoriamente no interior das lâminas de imagens aleatórias da base (sem repetição).

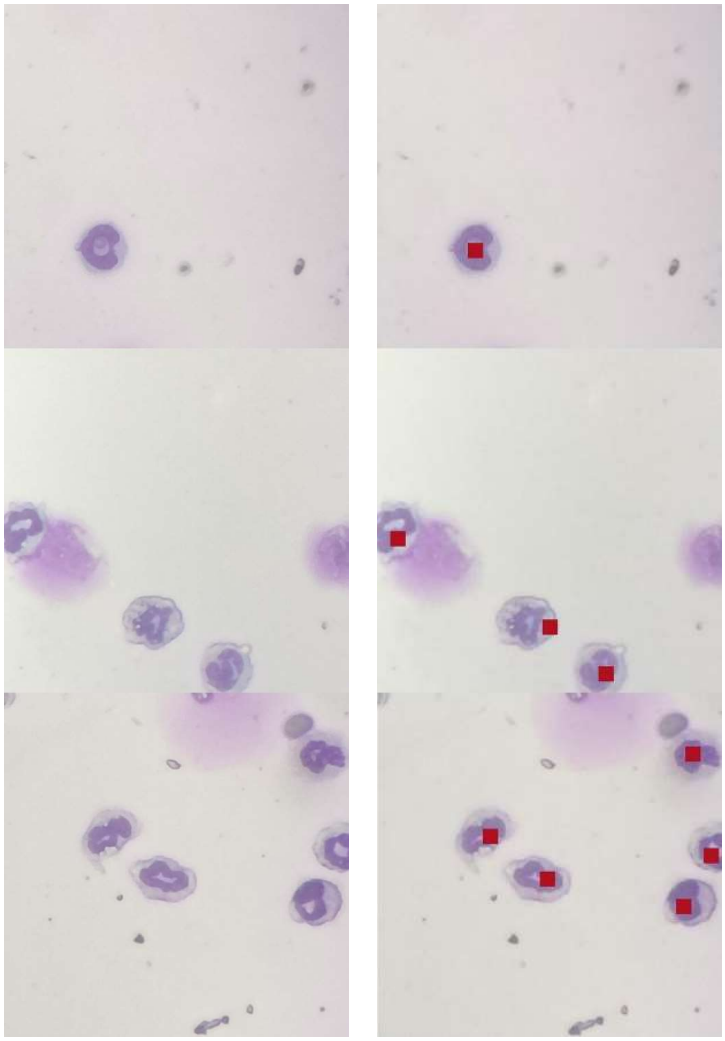
Os neutrófilos e linfócitos nos *patchs* foram rotulados manualmente pela especialista Gabriela Araujo de Azevedo. Foram utilizados quadrados vermelhos e azuis, respectivamente, para indicar a posição dessas células. Não houve preocupação em se rotular as células na borda dos *patchs*.

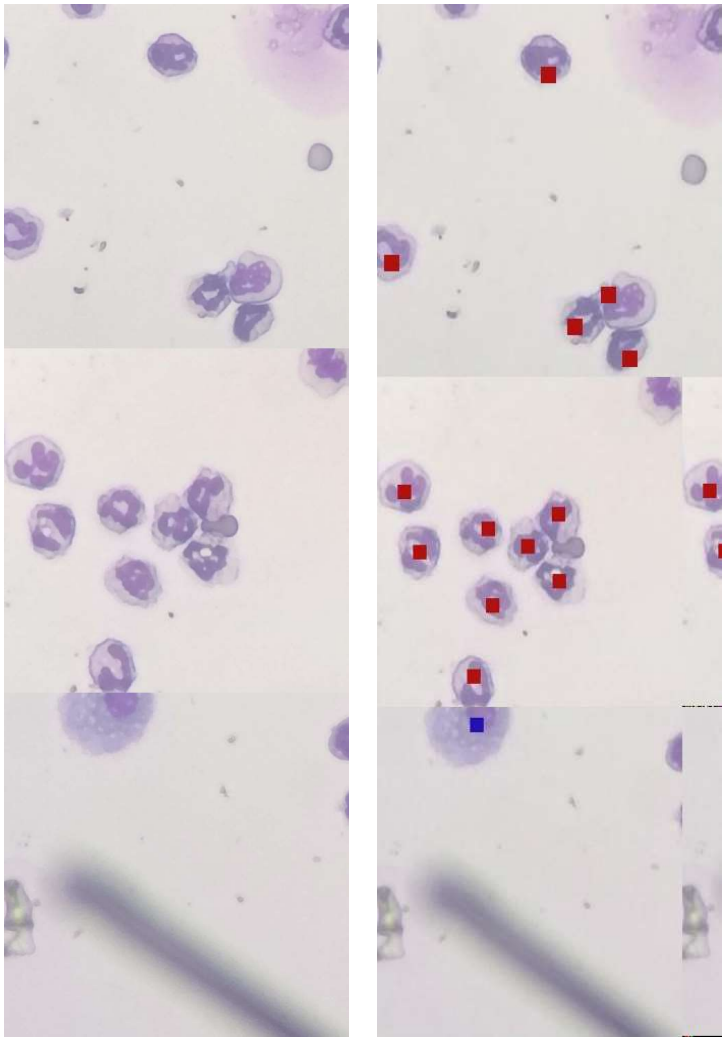
Abaixo são apresentadas as imagens originais e, ao lado de cada uma delas, a correspondente imagem contendo as anotações.

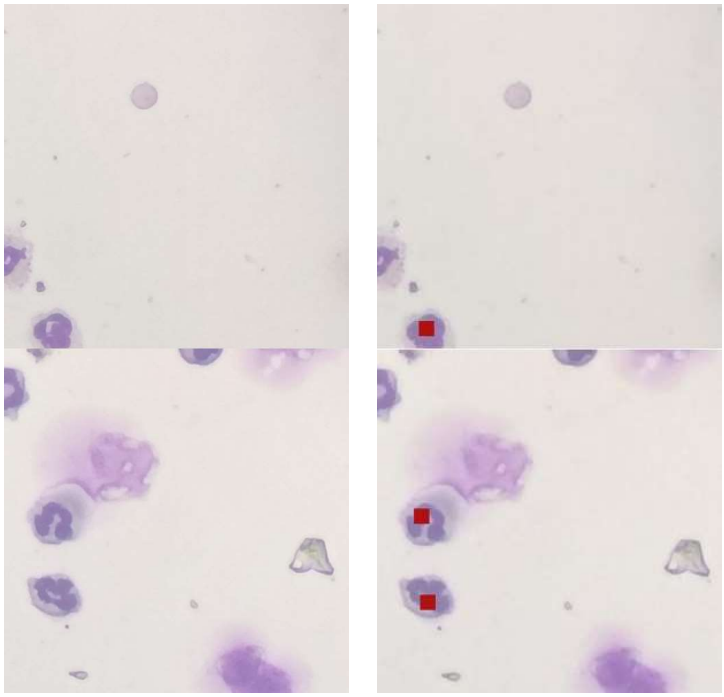












APÊNDICE B – TERMO DE AUTORIZAÇÃO PARA USO DAS IMAGENS

TERMO DE AUTORIZAÇÃO DE USO DE IMAGENS OBTIDAS NO LABORATÓRIO DA UNIVERSIDADE FEDERAL DE SÃO PAULO CAMPUS DIADEMA

Declaramos que nós, Gabriela Araujo de Azevedo, portador da cédula de identidade RG nº 32.513.023-1, e Richardt Gama Landgraf , portador da cédula de identidade RG nº 12.242.612-5, estamos de acordo com a condução do Trabalho de Conclusão de Curso "CONTAGEM DIFERENCIAL AUTOMÁTICA DE LEUCÓCITOS EM IMAGENS MICROSCÓPICAS" sob a responsabilidade de Daniel Luis Costa, portador da célula de identidade RG nº 39.214.896-1, e Nayara Viana Gerolomo, portadorda célula de identidade RG nº 38.079.066-x.

Estamos cientes que serão utilizados imagens microscópicas de amostras obtidas no Laboratório de Programação Fetal e Inflamação da Universidade Federal de São Paulo – Campus Diadema.

Gabriela Araujo de Azevedo
Mestranda de Ciências Farmacêuticas Unifesp

Richardt Gama Landgraf
Mestranda de Ciências Farmacêuticas Unifesp

Prof. Dr. Richardt G. Landgraf
Dep. de Ciências Biológicas
UNIFESP - Campus Diadema

APÊNDICE C – CÓDIGOS

C.1 MÉTODO

C.1.1 Delimitação da Lâmina

```
import cv2
import cv2.cv
from exibir_imagens import mostra_imagens
import numpy as np

def delimitacao_otsu(imagem, cor=0, removeEspurias=True, encaixaCirculo=True, analise=False):
    #Pega apenas o canal (ou a imagem em grayscale) especificado
    im_c = imagem[:,:,cor] if cor <= 2 else cv2.cvtColor(imagem, cv2.cv.CV_BGR2GRAY)

    if analise:
        print "- Delimitacao por limiarizacao de Otsu -"
        mostra_imagens([im_c], "", ["Imagem de entrada, apenas um canal"])

    #Binarizacao por Otsu
    limiar, binaria = cv2.threshold(im_c, 0, 255,
                                    cv2.THRESH_BINARY+cv2.THRESH_OTSU)

    if analise:
        mostra_imagens([binaria], "",
                       ["Imagem binaria limiarizada em " + str(limiar)])

    if removeEspurias:
        #Remocao de regioes menores dentro ou fora da lamina
        contours, hierarchy = cv2.findContours(binaria,
                                               cv2.RETR_EXTERNAL,
                                               cv2.CHAIN_APPROX_NONE)
        lamina = max(contours, key=cv2.contourArea) #assume que a lamina eh a regiao
        #com contorno de maior area

        binaria = np.zeros(binaria.shape, dtype=np.uint8)
        cv2.drawContours(binaria, [lamina], 0, 255, -1)

        if analise:
            mostra_imagens([binaria], "",
                       ["Apos remocao das regioes espurias"])

        if encaixaCirculo:
            M = cv2.moments(lamina)
            x = int(M["m10"] / M["m00"])
            y = int(M["m01"] / M["m00"])
            centro = (x, y)
            import math
            raio = int(math.sqrt(cv2.contourArea(lamina)/math.pi))

            im = None
            if analise:
                #Indice, sobre a regiao original, onde estao o centro e perimetro
                #do circulo encaixados.
                im = cv2.cvtColor(binaria, cv2.cv.CV_GRAY2BGR)
                cv2.circle(im, centro, raio, (0, 0, 255), 3)
                cv2.circle(im, centro, 10, (0, 0, 255), -1)

            binaria = np.zeros(binaria.shape, dtype=np.uint8)
            cv2.circle(binaria, centro, raio, 255, -1)
```

```
55
56              if analise:
57                  mostra_imagens([im, binaria], "Circulo encaixado")
58
59      return binaria
```

C.1.2 Remoção de Ruídos

```
1   import cv2
2
3   def filtro_media(imagem, size=5):
4       return cv2.blur(imagem, (size, size))
5
6   def filtro_mediana(imagem, size=5):
7       return cv2.medianBlur(src=imagem, ksize=size)
8
9   def filtro_gaussiano(imagem, size=5):
10      return cv2.GaussianBlur(imagem, (size, size), sigmaX=0)
11
12  def remocao_ruidos(imagem, analise=False):
13      #Aplica o filtro de mediana com kernel de tamanho 9x9.
14      return filtro_mediana(imagem, 9)
```

C.1.3 Limiarização

```
1   import cv2
2   import cv2.cv
3
4   def limiarizacao(imagem, lamina=None, cor=1, limiar=146, analise=False):
5       #Pega apenas o canal (ou a imagem em grayscale) especificado
6       im_c = imagem[:,:,cor] if cor <= 2 else cv2.cvtColor(imagem, cv2.cv.CV_BGR2GRAY)
7
8       #Limiariza no valor especificado;
9       #pixels de intensidade inferior ou igual a do limiar passam a compor o foreground.
10      l, im_b = cv2.threshold(im_c, limiar, 255, cv2.THRESH_BINARY_INV)
11
12      #Aplica a mascara da lamina, removendo as regioes do foreground
13      #que sejam exteriores a lamina
14      if lamina is not None:
15          im_b[lamina == 0] = 0
16
17      return im_b
```

C.1.4 Limiarização por Divergência Fuzzy

```
1   import cv2
2   import cv2.cv
3   from exibir_imagens import mostra_imagens
4
5   def limiarizacao_divergencia_fuzzy(imagem, mascara=None, c=float(1)/(255 - 0),
6                                      gamma=1, analise=False):
7       if analise:
8           from matplotlib import pyplot as plt
```

```
9           print "- Limiarizacao por divergencia fuzzy -"
10
11      if analise:
12          titulo = "Imagem de entrada, apenas um canal"
13          if mascara is not None:
14              cortada = cv2.bitwise_and(imagem, imagem, mask=mascara)
15              mostra_imagens([imagem, cortada], titulo,
16                             ["sem aplicacao da mascara", "apos aplicacao da mascara"])
17          else:
18              mostra_imagens(imagem, titulo)
19
20      #Extrai o histograma.
21      histograma = cv2.calcHist([imagem], [0], mascara, [256], [0, 256])
22      if analise:
23          plt.figure()
24          plt.title("Histograma da imagem")
25          plt.plot(range(0, 256), histograma, color = 'b')
26          plt.show()
27
28      #----- Fuzzy Divergence -----
29      from math import pi as pi
30      from math import e as e
31      #Divergencia fuzzy para com uma imagem idealmente segmentada se a imagem
32      #for binarizada por um determinado limiar
33      def divergencia_fuzzy(limiar, mi_b, mi_f, analise=False):
34          divergencia = 0
35          pertinencias_b, pertinencias_f = ([], [])
36          for intensidade in range(0, 256):
37              #calcula valor de pertinencia de um pixel desta intensidade a sua
38              #regiao associada
39              if intensidade <= limiar:
40                  mi = mi_b
41              else:
42                  mi = mi_f
43
44              #Com os valores de parametro padroes, a formula fica igual a em
45              #Ghosh, 2010.
46              #a distribuicao Cauchy generica nao possui o parametro "c".
47              pertinencia = (1/(c*pi*gamma))*(
48                      1/(1 + ((intensidade - mi)/gamma)**2))
49
50              if analise:
51                  if intensidade <= limiar:
52                      pertinencias_b.append(pertinencia)
53                  else:
54                      pertinencias_f.append(pertinencia)
55
56              #Soma ao calculo da divergencia fuzzy a parcela referente a esta
57              #intensidade.
58              divergencia += histograma[intensidade, 0]*(2
59                                  - pertinencia*e**(1-pertinencia)
60                                  - (2-pertinencia)*e**(pertinencia-1)
61                                  )
62          if analise:
63              figura, (plt1, plt2) = plt.subplots(1, 2)
64
65              plt1.set_title("no background")
66              plt1.plot(range(0,len(pertinencias_b)), pertinencias_b, color = 'b')
67              plt1.axvline(mi_b)
68
```

```
69              plt2.set_title("no foreground")
70              plt2.plot(range(len(pertinencias_b), 256), pertinencias_f, color = 'b')
71              plt2.axvline(mi_f)
72
73              plt.suptitle("Distribuicoes de pertinencia, limiar=" + str(limiar))
74              plt.show()
75
76          return divergencia
77
78      #Inicia a quantidade de pixels e soma das intensidades no back e foreground.
79      limiar_inicial = 0 #ultima intensidade do background
80      sum_intensidade_b = 0
81      sum_intensidade_f = 0
82      for intensidade in range(1, 256):
83          sum_intensidade_f += intensidade*histograma[intensidade, 0]
84      qtd_b = histograma[0, 0]
85      qtd_f = histograma[:, 0].sum() - histograma[0, 0]
86      if qtd_b == 0: #evita divisao por 0
87          mi_b = 0
88      else:
89          mi_b = sum_intensidade_b/qtd_b
90      if qtd_f == 0: #evita divisao por 0
91          mi_f = 0
92      else:
93          mi_f = sum_intensidade_f/qtd_f
94
95      #Inicia o limiar inicial como sendo o otimo, para futura comparacao.
96      menor_divergencia = divergencia_fuzzy(limiar_inicial, mi_b, mi_f)
97      melhor_limiar = limiar_inicial
98
99      divergencias = [menor_divergencia]
100     #Faz a busca linear pelo limiar que otimiza a divergencia fuzzy.
101     for limiar in range(1, 255):
102         #Modifica a quantidade de pixels e soma da intensidades no background
103         #e foreground com base na quantidade de pixels mudando de regiao.
104         qtd_mudando_regiao = histograma[limiar, 0]
105         sum_intensidade_b += limiar*qtd_mudando_regiao
106         sum_intensidade_f -= limiar*qtd_mudando_regiao
107         qtd_b += qtd_mudando_regiao
108         qtd_f -= qtd_mudando_regiao
109         if qtd_b == 0: #evita divisao por 0
110             mi_b = 0
111         else:
112             mi_b = sum_intensidade_b/qtd_b
113         if qtd_f == 0: #evita divisao por 0
114             mi_f = 0
115         else:
116             mi_f = sum_intensidade_f/qtd_f
117
118         #Se essa opcao for melhor que a atualmente otima, a substitue.
119         divergencia = divergencia_fuzzy(limiar, mi_b, mi_f,
120                                         analise=((limiar-10)%50==0 and analise))
121         if divergencia < menor_divergencia:
122             menor_divergencia = divergencia
123             melhor_limiar = limiar
124
125         if analise:
126             divergencias.append(divergencia)
127
128     if analise:
```

```
129             import numpy as np
130             fig = plt.figure()
131             ax1 = fig.add_subplot(111)
132             #ax1.title("Histograma da imagem")
133             ax1.plot(range(0, 256), histograma, color = 'b')
134             ax1.axvline(melhor_limiar)
135             div_norm = divergencias - np.min(divergencias)
136             div_norm = div_norm/np.max(div_norm)
137             div_norm = div_norm*np.max(histograma)
138             ax1.plot(range(0, 256),
139                      np.concatenate((div_norm, np.array([div_norm[-1]]))),
140                      color = 'r')
141             #ax2.imshow(np.vstack((divergencias, divergencias)), aspect=1,
142             #           norm=norm, cmap='coolwarm')
143             plt.suptitle("Divergencias para cada limiar")
144             plt.show()
145 
146             from scipy import stats
147             print "Descricao das divergencias para cada limiar: " + str(stats.describe(divergencias))
148             print "Limiar ganhador e sua divergencia associada: " + str((melhor_limiar, menor_divergencia))
149 
150         #Aplica o limiar otimo encontrado na limiarizacao
151         segmentos = cv2.threshold(imagem, melhor_limiar, 255, cv2.THRESH_BINARY_INV)[1]
152         if mascara is not None:
153             segmentos = cv2.bitwise_and(segmentos, segmentos, mask=mascara)
154 
155         #Usa metade do limiar como instruido em Ghosh, 2010 para se encontrar os nucleos
156         segmentos_div_2 = cv2.threshold(imagem, melhor_limiar/2, 255, cv2.THRESH_BINARY_INV)[1]
157         if mascara is not None:
158             segmentos_div_2 = cv2.bitwise_and(segmentos_div_2, segmentos_div_2, mask=mascara)
159 
160         if analise:
161             mostra_imagens([segmentos, segmentos_div_2], "Imagens limiarizadas",
162                            ["Limiar otimo", "Metade do limiar otimo"])
163 
164         return (melhor_limiar, segmentos, segmentos_div_2)
165         #----- /Fuzzy Divergence -----
166 
167     #Parser de parametros do script.
168     import argparse
169     parser = argparse.ArgumentParser(description="Limiarizacao por divergencia fuzzy.")
170     parser.add_argument("-i", "--input", help="Nome do arquivo da imagem de entrada.",
171                         action="store")
172     parser.add_argument("-m", "--mask",
173                         help="Nome do arquivo da imagem binaria de mascara.",
174                         action="store")
175     parser.add_argument("-s", "--salvar", help="Nome do arquivo da imagem de saida.",
176                         action="store")
177     parser.add_argument("-a", "--analise", help="Modo de analise.",
178                         action="store_true")
179     args = parser.parse_args()
180 
181     #Abre a imagem de entrada e considera apenas o canal verde
182     input_filename = args.input
183     if input_filename is not None:
184         imagem = cv2.imread(input_filename, cv2.IMREAD_COLOR)[:,:,1]
185 
186         if args.mask != None:
187             mascara = cv2.imread(args.mask, cv2.IMREAD_GRAYSCALE)
188         else:
```

```
189         mascara = None
190 
191     seg_1, seg_2 = limiarizacao_divergencia_fuzzy(imagem, mascara,
192                                                   gamma=30,
193                                                   c=1,
194                                                   analise=args.analise)
195 
196     #Se especificado, salva a imagem de saida
197     if args.salvar != None:
198         cv2.imwrite(args.salvar, seg_1)
199         cv2.imwrite(args.salvar[:-4] + "_2" + args.salvar[-4:], seg_2)
```

C.1.5 Extração de Regiões

```
1  import cv2
2  import cv2.cv
3  from exibir_imagens import mostra_imagens
4  import numpy as np
5  import random as rd
6  
7  #Gera cor aleatoria com saturacao e valor maximo (bem visivel), apenas alterando o hue.
8  def cor_aleatoria(minHue=0, maxHue=180):
9      c = np.zeros((1, 1, 3), dtype=np.uint8) #imagem para conversao
10     c[0][0] = [rd.randint(minHue, maxHue), 100, 100] #cor aleatoria em HSV
11     c = cv2.cvtColor(c, cv2.COLOR_HSV2BGR) #converte para BGR
12     cor = c[0][0]
13     return np.array([int(cor[0]), int(cor[1]), int(cor[2])])
14 
15 #Extrai os componentes conexos (em contornos) de uma imagem binaria.
16 #Se especificados kerneis, aplica uma abertura sobre os componentes (contornos externos)
17 #e uma abertura sobre os buracos (contornos internos).
18 #Se fornecida uma area minima, elimina as regioes que nao satisfizerem essa condicao.
19 def extrair_regioes(imagem, kernel_ext=None, kernel_int=None, min_area=None, analise=False, original=None):
20     if analise:
21         mostra_imagens([imagem], "", ["Imagem de entrada, apenas um canal"])
22 
23     #Encontra os contornos das regioes
24     contours, hierarchy = cv2.findContours(np.copy(imagem),
25             cv2.RETR_CCOMP, #associa contornos de componentes com contornos de buracos
26             cv2.CHAIN_APPROX_NONE) #armazena todos os pontos no contorno
27 
28     #Aplica as aberturas
29     if kernel_int is not None or kernel_ext is not None:
30         #Separa os componentes (contornos externos) e os buracos em 2 imagens
31         im_ext, im_bur = np.zeros((2,) + imagem.shape, dtype=np.uint8)
32         for c,cont in enumerate(contours):
33             if hierarchy[0,c,3] == -1: #nao tem pai -> eh componente
34                 cv2.drawContours(im_ext, [cont], -1, [255], -1)
35                 cv2.drawContours(im_ext, [cont], -1, [255], 1)
36             else:
37                 cv2.drawContours(im_bur, [cont], -1, [255], -1)
38                 cv2.drawContours(im_bur, [cont], -1, [255], 1)
39         if analise:
40             mostra_imagens([im_ext, im_bur], "Componentes e buracos separados")
41 
42         #Aplica uma abertura nos componentes externos para separar as celulas grudadas
43         if kernel_ext is not None:
44             im_ext = cv2.morphologyEx(im_ext, cv2.MORPH_OPEN, kernel_ext)
```

```
45
46          #Aplica uma abertura nos buracos para remover espurios e suavizar o formato
47          if kernel_int is not None:
48              im_bur = cv2.morphologyEx(im_bur, cv2.MORPH_OPEN, kernel_int)
49
50          if analise and (kernel_ext is not None or kernel_int is not None):
51              mostra_imagens([im_ext, im_bur], "Componentes e buracos abertos")
52
53          #Desenha os buracos sobre a imagem com os componentes abertos e retira os contornos novamente
54          im_ab = np.copy(im_ext)
55          im_ab[im_bur == 255] = 0
56
57          if analise:
58              mostra_imagens([im_ab], "Regioes apos abertura")
59
60          #Extrai os contornos atualizados
61          contours, hierarchy = cv2.findContours(im_ab,
62              cv2.RETR_CCOMP, #associa contornos de componentes com contornos de buracos
63              cv2.CHAIN_APPROX_NONE) #armazena todos os pontos no contorno
64
65      #Separa as regioes (cada regiao = 1 contorno externo + n contornos internos )
66      regioes = [] #lista de regioes
67      for c, contorno in enumerate(contours):
68          #Monta uma regiao para cada contorno externo (primeiro nivel hierarquico)
69          if hierarchy[0,c,3] == -1: #nao tem pai -> eh externo
70              regiao = [contorno]
71              #Se houverem filhos, os inclue.
72              if hierarchy[0,c,2] != -1: #ha filho(s)
73                  c_filho = hierarchy[0,c,2] #primeiro filho
74                  regiao.append(contours[c_filho])
75                  while hierarchy[0,c_filho,0] != -1: #itera pelos outros filhos
76                      c_filho = hierarchy[0,c_filho,0]
77                      regiao.append(contours[c_filho])
78              #Adiciona a regiao recem criada a lista de todas as regioes
79              regioes.append(regiao)
80
81      #Remove as regioes inferiores a min_area
82      if min_area is not None:
83          from extracao_caracteristicas import area
84          regioes = [r for r in regioes if area(r) >= min_area]
85
86          if analise:
87              #Desenha apenas as regioes superiores a min_area
88              im_area = np.zeros(imagem.shape, dtype=np.uint8)
89              for regiao in regioes:
90                  cv2.drawContours(im_area, regiao[:1], -1, [255], -1)
91                  cv2.drawContours(im_area, regiao[:1], -1, [255], 1)
92                  cv2.drawContours(im_area, regiao[1:], -1, [0], -1)
93                  cv2.drawContours(im_area, regiao[1:], -1, [0], 1)
94              mostra_imagens([im_area], "Apenas regioes superiores a area minima")
95
96      if original is not None and analise:
97          #Desenha os contornos obtidos sobre a imagem original
98          im_cont = np.copy(original)
99          for regiao in regioes:
100             cv2.drawContours(im_cont, regiao, -1, cor_aleatoria(), 2)
101         mostra_imagens([im_cont], "Regioes encontradas sobre a imagem original")
102
103     return regioes
```

C.1.6 Extração de Características

```
import cv2
import numpy as np

#Caracteristicas geometricas

def area(regiao): #0
    return (cv2.contourArea(regiao[0]) - sum(map(cv2.contourArea, regiao[1:])))

def area_convex_hull(regiao): #1
    return cv2.contourArea(cv2.convexHull(regiao[0]))

def solidez(regiao): #2
    ach = area_convex_hull(regiao)
    return area(regiao)/ach if ach != 0 else 0

def perimetro(regiao): #3
    return sum(map(lambda cnt: cv2.arcLength(cnt, True), regiao))

def compacidade(regiao): #4
    p2 = perimetro(regiao)**2
    return area(regiao)/p2 if p2 != 0 else 0

def circularidade(regiao): #5
    _, raio = cv2.minEnclosingCircle(regiao[0])
    a = area(regiao)
    return (np.pi * raio**2)/a if a != 0 else 0

def retangularidade(regiao): #6
    _, medidas, _ = cv2.minAreaRect(regiao[0])
    w, h = medidas
    a = area(regiao)
    return float(w*h)/a if a != 0 else 0

def tem_buraco(regiao): #7
    return len(regiao) > 1

caracteristicas_geometricas = [area, area_convex_hull, solidez, perimetro, compacidade, circularidade, retangularidade, te


#Caracteristicas de textura

def media(pixels): #8
    return np.mean(pixels)

def variancia(pixels): #9
    return np.var(pixels)

def desvio_padrao(pixels): #10
    return np.std(pixels)

caracteristicas_textura = [media, variancia, desvio_padrao]

medias, desvios_padroes = np.zeros((2, len(caracteristicas_geometricas) + len(caracteristicas_textura)), dtype=float)
normalizado = False

#Calcula a media e o desvio padrao das imagens em um conjunto de treinamento
#para posterior normalizacao das caracteristicas
```

```
58  def define_normalizacao(regioes, imagens):
59      caracteristicas = map(lambda x: np.array(extracao_caracteristicas(*x, idx_caracteristicas = range(11), normalizar=Fals
60
61      global medias
62      medias = np.mean(caracteristicas, axis=0)
63
64      global desvios_padroes
65      desvios_padroes = np.std(caracteristicas, axis=0)
66
67      global normalizado
68      normalizado = True
69
70  def checa_normalizacao():
71      return (medias, desvios_padroes)
72
73  def extracao_caracteristicas(regiao, #lista de contornos, sendo o primeiro externo e os demais internos
74                               imagem,
75                               idx_caracteristicas, #lista com os indices das caracteristicas utilizadas
76                               normalizar=True):
77
78      c_g = map(lambda car: car(regiao),[caracteristicas_geometricas[i]
79                for i in idx_caracteristicas if i < len(caracteristicas_geometricas)])
80
81      #Encontra os pixels internos a regiao
82      im_c = cv2.cvtColor(imagem, cv2.cv.CV_BGR2GRAY)
83      mascara = np.zeros(im_c.shape, dtype=np.uint8)
84      cv2.drawContours(mascara, [regiao[0]], -1, 255, -1)
85      cv2.drawContours(mascara, regiao[1:], -1, 0, -1)
86      pixels_internos = im_c[mascara == 255]
87
88      c_t = map(lambda car: car(pixels_internos),[caracteristicas_textura[i-len(caracteristicas_geometricas)]
89                for i in idx_caracteristicas if i >= len(caracteristicas_geometricas)])
90
91      cs = np.array(c_g + c_t)
92
93      if normalizar:
94          if normalizado:
95              cs = (cs - medias)/desvios_padroes
96          else:
97              raise("Normalizacao nao foi definida!")
98
99      return cs
```

C.1.7 Classificação

```
1  import numpy as np
2  from sklearn import svm
3
4  svm_neutrofilo = None
5  svm_linfocito = None
6
7  def treina_classificacao(vetores_caracteristicas, classes):
8      global svm_neutrofilo
9      svm_neutrofilo = svm.SVC(kernel="linear", probability=True)
10     svm_neutrofilo.fit(vetores_caracteristicas, np.array(classes) == "n")
11
12     global svm_linfocito
13     svm_linfocito = svm.SVC(kernel="linear", probability=True)
```

```
    svm_linfocito.fit(vetores_caracteristicas, np.array(classes) == "l")

def classifica(caracteristicas):
    if svm_neutrofilo is None:
        raise Exception("SVM nao foi treinada")

    certeza_neut = svm_neutrofilo.predict_proba([caracteristicas])
    certeza_linfo = svm_linfocito.predict_proba([caracteristicas])
    classe = max(range(3),  key=(lambda x: ([certeza_neut[0,1], certeza_linfo[0,1], 0.5])[x]))
    return [["n", "l", "o"][classe], certeza_neut[0,1], certeza_linfo[0,1]]
```

C.1.8 Treinamento e Curva ROC

```
import extracao_caracteristicas
import classificacao
import base
from scipy import interp
import numpy as np
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import StratifiedKFold
import matplotlib.pyplot as plt

def treina(regioes):
    imagens = map(lambda x: x.get_imagem(), base.base.get_imagens())
    caracteristicas = extracao_caracteristicas.define_normalizacao(
            map(lambda x: x.contornos, regioes),
            map(lambda x: imagens[x.imagem-1], regioes))

    classificacao.treina_classificacao(caracteristicas,
                                       map(lambda x:x.classe, regioes))

def plota_roc():
    regioes = np.array(base.base.get_regioes(lambda x: x.classe in ["o", "l", "n"]))
    cv = StratifiedKFold(n_splits=4)
    classes = np.array(map(lambda x: x.classe, regioes))

    tprs = []
    aucs = []
    mean_fpr = np.linspace(0, 1, 100)

    i = 0
    for train, test in cv.split(regioes, classes):
        treina(regioes[train])
        ec = extracao_caracteristicas.extracao_caracteristicas
        vectores_caracteristicas = [ec(regiao.contornos,
                                       base.base.get_imagens()[regiao.imagem-1].get_imagem(),
                                       range(11))
                                    for regiao in regioes[test]]
        pred = np.array([classificacao.classifica(carac) for carac in vectores_caracteristicas])

        #neutrofilo

        fpr, tpr, thresholds = roc_curve(np.array(map(lambda x: int(x), classes[test] == "l")),
                                         np.array(map(lambda x: float(x), pred[:, 2])))
        tprs.append(interp(mean_fpr, fpr, tpr))
        tprs[-1][0] = 0.0
        roc_auc = auc(fpr, tpr)
        aucs.append(roc_auc)
```

```
46          plt.plot(fpr, tpr, lw=1, alpha=0.3,
47                   label='ROC da dobra %d (AUC = %0.2f)' % (i, roc_auc))
48          i += 1
49
50      plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
51               label='Aleatorio', alpha=.8)
52
53      mean_tpr = np.mean(tprs, axis=0)
54      mean_tpr[-1] = 1.0
55      mean_auc = auc(mean_fpr, mean_tpr)
56      std_auc = np.std(aucs)
57      plt.plot(mean_fpr, mean_tpr, color='b',
58               label=r'ROC medio (AUC = %0.2f $\pm$ %0.2f)' % (mean_auc, std_auc),
59               lw=2, alpha=.8)
60
61      std_tpr = np.std(tprs, axis=0)
62      tprs_upper = np.minimum(mean_tpr + std_tpr, 1)
63      tprs_lower = np.maximum(mean_tpr - std_tpr, 0)
64      plt.fill_between(mean_fpr, tprs_lower, tprs_upper, color='grey', alpha=.2,
65                       label=r'$\pm$ 1 desv. pad.')
66
67      plt.xlim([-0.05, 1.05])
68      plt.ylim([-0.05, 1.05])
69      plt.xlabel('Razao Falso Positivo')
70      plt.ylabel('Razao Verdadeiro Positivo')
71      plt.title('Reconhecimento de Linfocitos')
72      plt.legend(loc="lower right")
73      plt.show()
```

C.2 EXPERIMENTOS

C.2.1 Experimento 1

```
1  #Varia os os parametros da delimitacao_otsu e avalia a saida usando a
2  #metrica da distancia de Jaccard, comparando a mascara produzida com a lamina anotada.
3
4  import base
5  from delimitacao_otsu import delimitacao_otsu
6  import numpy as np
7  import cv2
8
9  imagens = base.base.get_imagens(lambda im: im.lamina is not None)
10
11 #Parametros variados
12 cores = ["blue", "green", "red", "grey"] #BGR e cinza
13 rem_reg_esp = [True, False] #remocao de regioes espurias
14 #(eh mantida a com maior area e removidas as demais)
15 enc_circ = [True, False] #encaixe de um circulo
16
17 shape = (len(cores), #cores
18          len(rem_reg_esp),
19          len(enc_circ),
20          len(imagens))
21 notas = np.zeros(shape, dtype=float)
22
23 for i, imagem in enumerate(imagens): #para cada imagem
```

```
    im_ori = imagem.get_imagem()
    im_lam = imagem.lamina.get_imagem() #lamina anotadas
    for c, cor in enumerate(cores): #em cada canal de cor e nos tons de cinza
        im_cor = im_ori[:,:,c] if cor != "grey" else cv2.cvtColor(im_ori, cv2.cv.CV_BGR2GRAY)

        #Conduz a delimitacao da lamina limiarizando por otsu.
        #Varia os dois parametros da delimitacao_otsu.
        for r, rem in enumerate(rem_reg_esp):
            for e, enc in enumerate(enc_circ):
                im_otsu = delimitacao_otsu(im_cor, analise=False,
                                           removeEspurias=rem,
                                           encaixaCirculo=enc)
                #Armazena a distancia de jaccard da mascara computada para com a anotada.
                notas[c,r,e,i] = base.distancia_jaccard(im_lam, im_otsu)

                print(cor,rem,enc,i,notas[c,r,e,i])

import pickle
with open("experimentos//delimitacao02.pkl", "wb") as f:
    pickle.dump(notas, f)
    f.close()
```

C.2.2 Experimentos 2 e 3

```
from remocao_ruidos import filtro_media, filtro_mediana, filtro_gaussiano
import base
import cv2
import numpy as np

#Parametros variados
filtros = [lambda im, ks: im, filtro_media, filtro_mediana, filtro_gaussiano]
tamanhos_kernels = [3, 5, 7, 9]
cores = [0, 1, 2, 3]
limiares = range(70, 241)
imagens = base.base.get_imagens(lambda im:im.lamina is not None #lamina anotada
                                and im.get_patch().celulas is not None) #celulas anotadas

notas = np.zeros(shape=(len(filtros), len(tamanhos_kernels), len(cores), len(limiares), len(imagens)),
                 dtype=float)

for i in range(len(imagens)):
    imagem = imagens[i]
    im_total = imagem.get_imagem() #carrega a imagem
    patch = imagem.get_patch()

    for c in range(len(cores)):
        #Pega a imagem na cor escolhida
        cor = cores[c]
        im_cor = None
        if cor < 3:
            im_cor = im_total[:,:,cor]
        else:
            im_cor = cv2.cvtColor(im_total, cv2.cv.CV_BGR2GRAY)

        for f in range(len(filtros)):
            filtro = filtros[f]
            for t in range(len(tamanhos_kernels)):
                tamanho_kernel = tamanhos_kernels[t]
```

```
35
36                  #Aplica o filtro sobre a imagem completa para nao haver prejuizo nas extremidades do patch
37                  im_filt = filtro(im_cor, tamanho_kernel)
38                  x, y = patch.canto
39                  h, w = base.shape_patch
40                  im_patch = im_filt[y:y+h, x:x+h]
41
42                  for l in range(len(limiares)):
43                      #Aplica o limiar e salva a distancia de jaccard para com o padrao ouro
44                      limiar = limiares[l]
45                      im_bin = cv2.threshold(im_patch, limiar, 255, cv2.THRESH_BINARY_INV)[1]
46                      notas[f, t, c, l, i] = patch.avaliacao_distancia_jaccard(im_bin)
47
48                      print (f, tamanho_kernel, cor, limiar, i, notas[f, t, c, l, i])
49
50  import pickle
51  with open("experimentos/remocao_ruidos01.pkl", "wb") as f:
52      pickle.dump(notas, f)
53      f.close()
```

C.2.3 Experimento 4

```
1   #Para as imagens com as celulas anotadas, apos aplicacao do filtro de mediana (tamanho kernel variavel),
2   #checa no patch a qualidade da segmentacao por limiarizacao nos limiares 146(que o de maior indice de Jaccard medio) +- 5
3   #sobre o canal verde, apos extracao de regioes com variacao nos kerneis das aberturas morfologicas.
4
5   import base
6   import cv2
7   import numpy as np
8   import pickle
9   from remocao_ruidos import filtro_mediana
10  from extrair_regioes import extrair_regioes
11
12  #Parametros variaveis
13  kernels_filt_med = [5, 7, 9] #tamanhos do kernel do filtro de mediana
14  limiares = range(146-10, 146+10+1)
15  imagens = base.base.get_imagens(lambda im:im.lamina is not None and im.get_patch().celulas is not None)
16
17  #Define os kerneis de abertura (variavel)
18  morphs = [cv2.MORPH_CROSS, cv2.MORPH_RECT, cv2.MORPH_ELLIPSE]
19  kernels_ext = [None] + [cv2.getStructuringElement(morph, (ksize, ksize))
20                  for ksize in range(3, 8, 2) for morph in morphs]
21  kernels_int = [None]
22  # + [cv2.getStructuringElement(morph, (ksize, ksize))
23  #                 for ksize in range(3, 6, 2) for morph in morphs]
24  #areas_min = range(500, 1000, 100)
25
26  notas = np.zeros((len(kernels_filt_med),
27                    len(kernels_ext),
28                    len(kernels_int),
29                    #len(areas_min),
30                    len(limiares),
31                    len(imagens)),dtype=float)
32
33  for i, imagem in enumerate(imagens): #para cada imagem
34      patch = imagem.get_patch()
35      im_g = imagem.get_imagem()[:,:,1] #pega apenas o canal verde
36
```

```
37      for kf,kernel_filt_med in enumerate(kernels_filt_med):
38          im_f = filtro_mediana(im_g, kernel_filt_med) #aplica o filtro de mediana
39          x, y = patch.canto
40          h, w = base.shape_patch
41          im_p = im_f[y:y+h, x:x+w] #pega apenas a regiao do patch
42
43          for l, limiar in enumerate(limiares):
44              im_b = cv2.threshold(im_p, limiar, 255, cv2.THRESH_BINARY_INV)[1] #limiariza
45
46              for ke, kernel_ext in enumerate(kernels_ext): #para cada kernel externo
47                  for ki, kernel_int in enumerate(kernels_int): #para cada kernel interno
48                      #for a, area_min in enumerate(areas_min): #para cada area minima
49                      regioes = extrair_regioes(im_b, kernel_ext, kernel_int) #extrai as regioes
50
51                      #Calcula a media de, para cada celula, a distancia de Jaccard para com a regiao mais semelhante
52                      notas[kf,ke,ki,l,i] = patch.avaliacao_forca_bruta(regioes, base.distancia_jaccard, ignora_borda=True)
53
54                      print(kf,ke,ki,limiar,imagem.indice,notas[kf,ke,ki,l,i])
55
56      with open("experimentos//extrair_regioes02.pkl", "wb") as f:
57          pickle.dump(notas, f)
58          f.close()
```