UNICAMP

Universidade Estadual de Campinas

Faculdade de Engenharia Mecânica

IM563 – Processamento de Imagens
Aplicado à Automação e Robótica

5 de abril de 2024

Docente: Paulo R. G. Kurka

Discente:

– Gabriel Toffanetto França da Rocha – 289320

Trabalho 1

Sumário

1 Redução da imagem

A imagem colorida trabalhada, possuí três canais, para as componentes RGB, e um tamanho de 960 colunas e 540 linhas, ou seja, 960x540 pixels. Para realizar a redução do tamanho da imagem, foi construída uma função na linguagem Python, que a partir da transformação da imagem em um *array* tridimensional, realiza a redução de acordo com a porcentagem escolhida.

A redução é realizada por meio da divisão da imagem original em macro pixels, que são regiões da imagem original onde todos os pixels ali contidos irão formar um único pixel na imagem reduzida. Os macro pixels da imagem original são varridos, realizando a média dos três canais de cor e reconstruindo um pixel para a imagem reduzida.

Nas Figuras 1, 2 e 3, pode-se observar a imagem original, a imagem com redução à 50% do seu tamanho original e com redução à 25% do tamanho original, respectivamente. Como as três imagens são exibidas neste documento no mesmo tamanho físico, a densidade de pixels de cada uma delas é diferente, o que está ligado diretamente à nitidez e definição da imagem. Esses efeitos podem ser percebidos nos componentes da imagem, como por exemplo o piso ao redor da piscina. Na imagem original, as formas que compõem a decoração do piso são facilmente percebidas, enquanto nas imagens reduzidas o mesmo se faz mais difícil, e quase impossível na imagem reduzida à 25%. A percepção das bordas da piscina também passam a apresentar serrilhados, e a definição da face do cachorro presente na imagem também é degradada.

Dessa forma, observa-se que ao reduzir a imagem, perde-se informação na forma de nitidez e definição. Por outro lado, o número de pixels é reduzido, o que reduz a necessidade de poder de processamento e armazenamento, o que é vital para a aplicações em sistemas embarcados, principalmente os móveis, onde o consumo energético também é uma informação muito relevante. Logo, a redução da imagem, desde que não comprometa o funcionamento do algoritmo que se pretende aplicar pode ser vital para a viabilidade da aplicação.

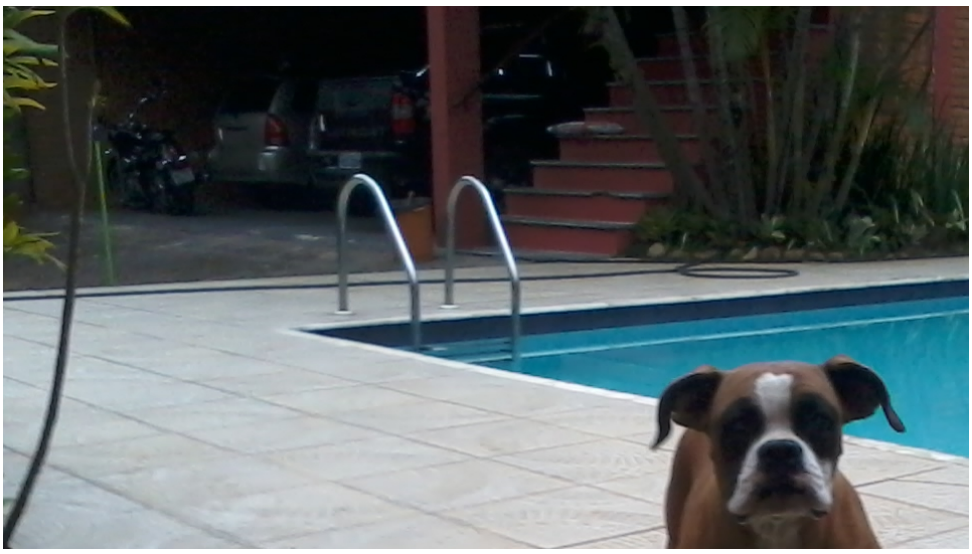


Figura 1: Imagem original com tamanho 540x960.

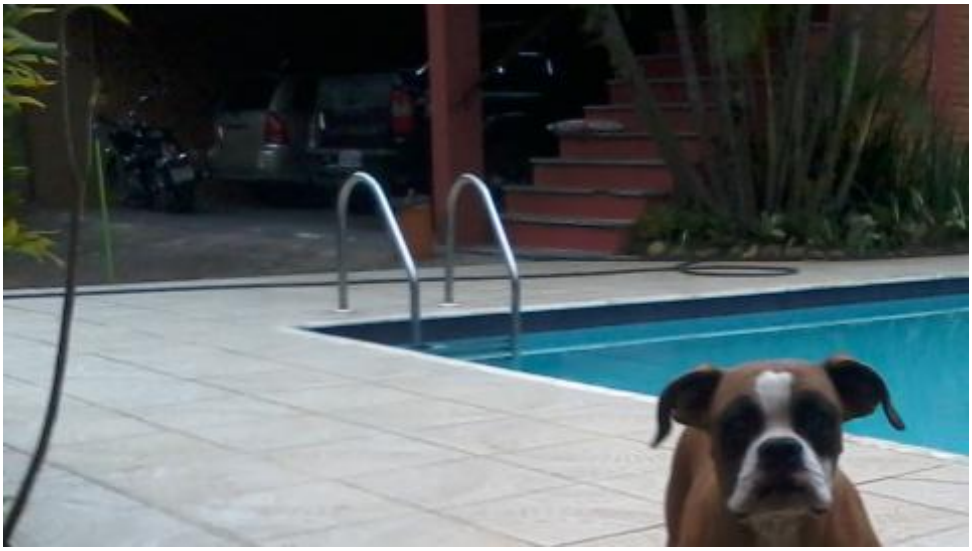


Figura 2: Imagem com redução de 50%, com tamanho 270x480.

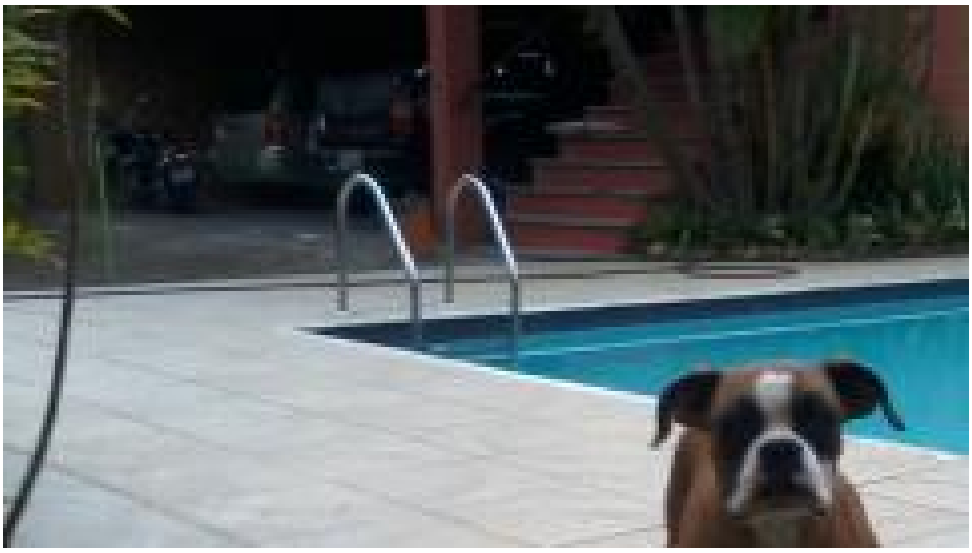


Figura 3: Imagem com redução de 25%, com tamanho 135x240.

2 Reconstrução da imagem em tons de cinza

A reconstrução das imagens em escala de cinza se deu por meio da junção dos canais vermelho(R), verde(G) e azul(B), em um único canal, de escalas de cinza, onde a intensidade é diretamente proporcional por o quão claro é aquele pixel.

Uma vez utilizando a linguagem Python, juntamente da biblioteca Imageio, carregou-se a imagem em memória na forma de um *array* tridimensional, onde a as dimensões do mesmo representam as colunas e linhas da matriz de pixels da imagem, e o canal de RGB de cor respectivo daquela matriz de linhas e colunas.

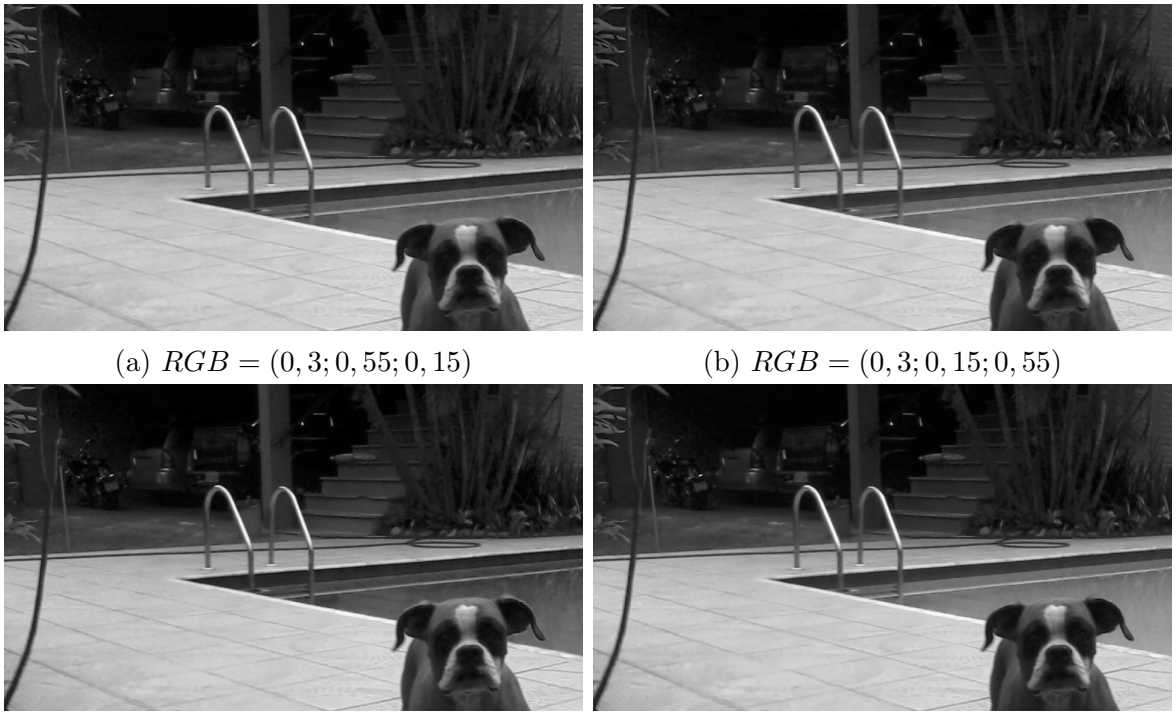
2.1 Resultados

Realizando a reconstrução da imagem segundo a combinação $Y_1 = 0,3 \cdot R + 0,55 \cdot G + 0,15 \cdot B$, obteve-se a imagem da Figura 4a.

Já por meio da combinação $Y_2 = 0,3 \cdot R + 0,15 \cdot G + 0,55 \cdot B$, a reconstrução da imagem é mostrada na Figura 4b.

Fazendo $Y_3 = 0,55 \cdot R + 0,3 \cdot G + 0,15 \cdot B$, pode-se reconstruir a imagem original em escala de cinzas, obtendo o resultado exibido na Figura 4c.

Por fim, a imagem foi reconstruída a partir da lei $Y_4 = 0,15 \cdot R + 0,55 \cdot G + 0,3 \cdot B$, sendo seu resultado a Figura 4d.



(a) $RGB = (0,3; 0,55; 0,15)$ (b) $RGB = (0,3; 0,15; 0,55)$

(c) $RGB = (0,55; 0,3; 0,15)$ (d) $RGB = (0,15; 0,55; 0,3)$

Figura 4: Imagem reconstruída em escala de cinza para diferentes combinações das componentes RGB.

2.2 Análises

Analisando o contexto da imagem trabalhada, observa-se que a mesma possuí componentes formados com forte presença das cores primárias e secundárias, como a piscina que é ciano, a construção que possuí um tom de vermelho, e a vegetação verde escuro. Dessa forma, espera-se que com o ponderamento das cores, esses três componentes da imagem sejam destacados ou ofuscados, de acordo com o quanto a sua cor principal será relevante para a intensidade em tons de cinza.

Com isso, pode-se observar na Figura 4c, que possuí a maior contribuição da cor vermelha, que a construção tem aparência mais clara, destacando mais os tons de vermelho na escala de cinzas, que aparecem com luz mais intensa. O contrário também se verifica, uma vez que a Figura 4d que tem a menor contribuição do vermelho, a construção está mais escura.

A cor ciano, é uma cor secundária, formada pela junção do azul e do verde. Dessa forma, observamos que mesmo a piscina tendo um tom azulado, ela se destaca mais na Figura 4d do que na Figura 4b, onde a contribuição do azul é maior. Isso ocorre justamente pela contribuição também do ver, onde somando a participação de azul e verde, é maior na ultima imagem.

Já a vegetação, se mostra mais destacada na Figura 4a, que possuí uma maior representação da cor verde na escala de cinzas, e o contrário se vê na Figura 4b, onde os galhos são até de difícil distinção devido a falta de luminosidade na representação da imagem para tal informações.

Dessa forma, o ponderamento das componentes vermelho, verde e azul na construção de uma intensidade de luz para reconstrução da imagem em escala de cinzas tem a capacidade de destacar ou ofuscar partes da imagem de acordo com a cor dessas regiões, onde, quanto maior contribuição uma cor tiver na luminância, mais clara (mais destaque) a mesma irá apresentar em escala de cinzas. Esse fenômeno pode ser utilizado tanto para o equilíbrio da luz em imagens em escala de cinzas, como também para destacas pontos de interesse, e aumentar o contraste entre regiões que se deseja detectar borda ou segmentar, facilitando a realização do processo. Um exemplo disso, é ao fazer um veículo autônomo seguir a faixa central de rodovia por meio de visão computacional, pode-se utilizar a imagem em escala de cinzas, onde as cores vermelho e verde tem maior contribuição, dessa forma fazendo com que o amarelo da faixa seja destacado e o contraste com o asfalto seja maior, facilitando a extração da linha para controle do veículo.

3 Equalização via Historiograma

A equalização do histograma de uma imagem visa o aproveitamento total da faixa de intensidade do pixel, que no caso de uma intensidade de 8 bits, varia de 0 (preto), até o 255 (branco). Devido às condições do ambiente, e a configuração da câmera utilizada, esse *range* pode ser mal aproveitando, ocupando apenas um intervalo reduzido da faixa de intensidades que podem ser representadas.

Utilizando a imagem reconstruída em escala de cinzas da Figura 4a, obteve-se o histograma normalizado da imagem, realizando a varredura de todos os pixels da imagem e realizando a contagem da quantidade de pixels para cada intensidade de luminosidade, e por fim realizando a normalização do mesmo, obtendo assim valores com a ideia de probabilidade de ocorrência de cada intensidade na imagem. O histograma normalizado da imagem é apresentado na Figura 5a. O histograma cumulativo também foi implementado, realizando o somatório das classes do histograma normalizado, sendo apresentado na Figura 6b.

Como é observado no histograma da Figura 5a, ocorre um pequeno mal aproveitamento das menores intensidades de luminosidade, e uma mal distribuição sobre as intensidade maiores, acima de aproximadamente 220. O mesmo pode ser visto também pelo histograma cumulativo da Figura 6b, onde existe um platô próximo de zero, e a partir de aproximadamente 220, mostrando que essas faixas de valores de intensidades não são aproveitadas na imagem.

Para a solução do problema, o histograma foi varrido com uma estratégia de detecção das faixas superiores e inferiores do histograma, comparando o valor da probabilidade de cada intensidade com um valor de limiar. O valor inferior do histograma foi nomeado como `bias` e o superior como `scale`, sendo tais valores responsáveis, respectivamente, pela translação e o escalonamento do histograma.

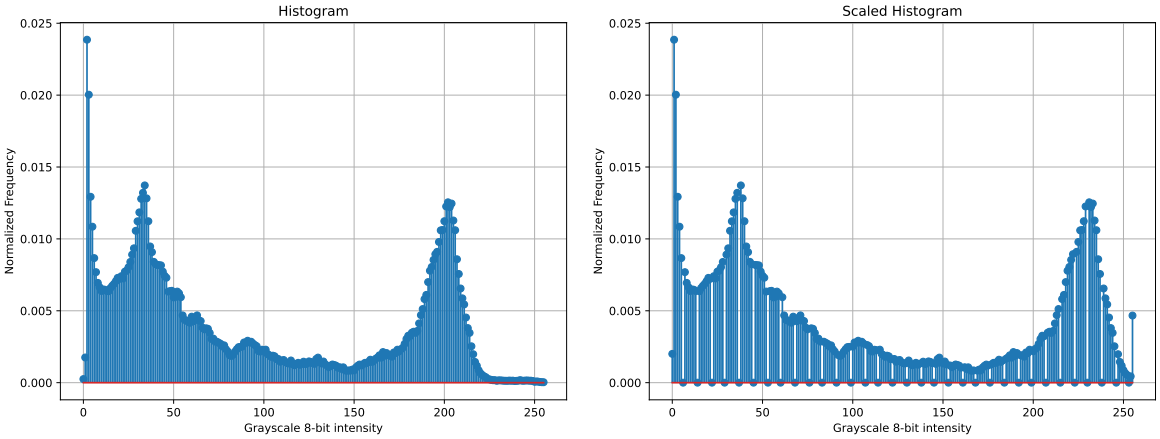
O procedimento de escalonamento e translação foi realizado conforme (1), onde L é número de valores de intensidade possíveis, calculando a intensidade conforme o fator de correção de escala e translação, além de realizar a saturação do valor entre 0 e 255 para garantir que os valores obtidos estão na faixa aceita pelo mapeamento de intensidades. Destacasse que como foi utilizado o histograma normalizado, não se faz necessária a inclusão do número total de pixels da imagem na equação.

$$I_s(x,y) = \left[\frac{L-1}{scale} \cdot I(x,y) - bias \right] \in (0, 255) \tag{1}$$

Por meio da correção, utilizando (1), utilizando os valores $bias = 1$ e $scale = 222$, obtém-se o histograma da Figura 5b, onde pode-se facilmente perceber que toda a faixa de 0 a 255 foi aproveitada, mesmo que com uma concentração heterogênea. O efeito de tal equalização também é visto no histograma cumulativo da Figura 6a, de forma que os platôs existentes foram dissolvidos pela melhor representação da intensidade dos pixels da imagem.

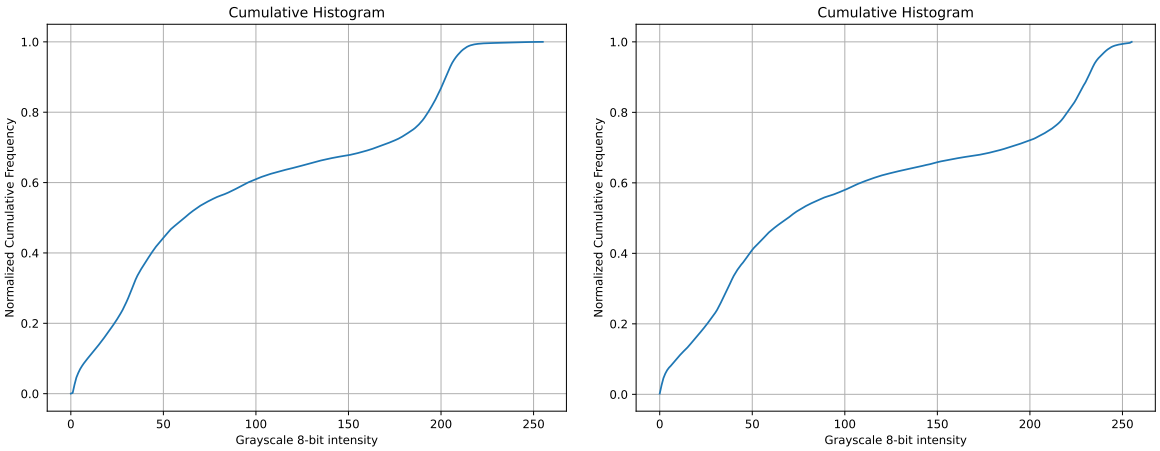
O resultado do tratamento na imagem pode ser comparado nas Figuras 7a e 7b, que apresenta um maior brilho, devido ao aproveitamento da faixa superior do histograma. Como a correção da faixa inferior do histograma foi mínima, não se observa grandes mudanças nas

partes mais escuras da imagem.



(a) Histograma da imagem original. (b) Histograma transladado e escalonado.

Figura 5: Histograma normalizado da imagem antes e depois da correção de translação e escalonamento do histograma.



(a) Histograma cumulativo da imagem original. (b) Histograma cumulativo da imagem após escalonamento.

Figura 6: Histograma cumulativo normalizado da imagem antes e depois da correção de translação e escalonamento do histograma.

(a) Imagem original. (b) Imagem após correção.

Figura 7: Imagem antes e depois da correção de translação e escalonamento do histograma.

4 Equalização via Historiograma Acumulado

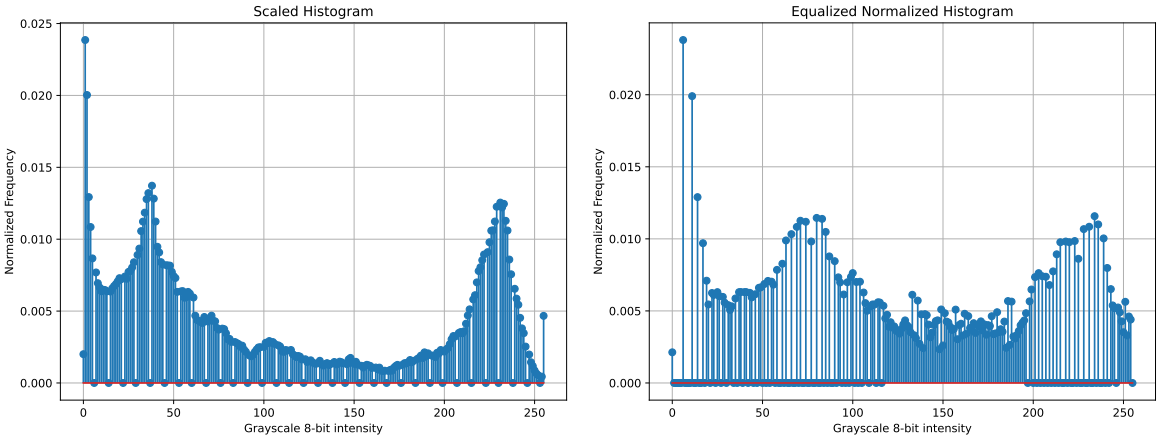
Como o escalonamento do histograma realizado anteriormente, foi possível obter o aproveitamento de toda faixa de intensidades disponível para os pixels, porém, essa distribuição ainda é bastante irregular. Essa irregularidade pode ser vista pela falta de constância do histograma na Figura 8a, e também pela falta de linearidade do histograma acumulado, na Figura 9a. Dessa forma, é necessário realizar a equalização do histograma, buscando uma distribuição homogênea das intensidades dos pixels.

Para realizar tal equalização, construiu-se uma função que dado os pixels da imagem a o histograma cumulativo normalizado, implementa (2), onde a intensidade equalizada de cada pixel se dá de acordo com o histograma acumulado normalizado para a intensidade original do pixel, linearizado pelo valor de máxima intensidade do pixel.

$$I_e(x, y) = (L - 1) \cdot C_f[I(x, y)] \tag{2}$$

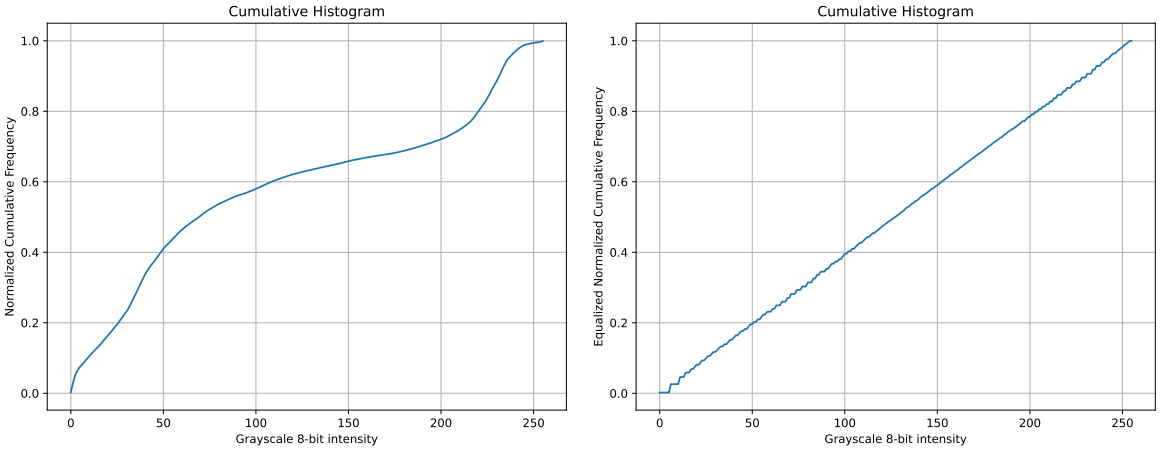
Ao realizar a equalização, observa-se na Figura 8b uma distribuição, mesmo que ainda longe de estar contínua, mais suave, apresentado vales menos profundos e picos mais baixos. Ao observar o histograma cumulativo, Figura 9b, vê-se que o mesmo se apresenta com uma forma linear, mostrando que foi possível deixar a distribuição final bastante homogênea.

Na imagem final, isso impacta em um aumento do contraste entre partes claras e escuras, o que pode ser observado comparando as imagens das Figuras 10a 10b, onde a visualização dos veículos ao fundo da imagem foi melhorada, e outros detalhes e bordas de difícil diferenciação foram também melhor destacados.



(a) Histograma da imagem original. (b) Histograma transladado e escalonado.

Figura 8: Histograma normalizado da imagem antes e depois da correção de translação e escalonamento do histograma.



(a) Histograma cumulativo da imagem original.

(b) Histograma cumulativo da imagem após escalonamento.

Figura 9: Histograma cumulativo normalizado da imagem antes e depois da correção de translação e escalonamento do histograma.

(a) Imagem original.

(b) Imagem após correção.

Figura 10: Imagem antes e depois da correção de translação e escalonamento do histograma.

5 Extração de bordas

Para a realização da extração de bordas da imagem, foi utilizado o método de Canny (CANNY, 1986), realizando a suavização da imagem por meio da convolução da mesma com um filtro Gaussiano de ordem 5×5, apresentado em (3), resultando na imagem borrada, I_b, a partir de (4).

$$B = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 2 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 2 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{3}$$

$$I_b = B * I \therefore I_b(i,j) = \sum_{n=0}^{5}\sum_{m=0}^{5} B(n,m) \cdot I(i-2+n, j-2+m) \tag{4}$$

Por meio da aplicação do filtro Gaussiano, à imagem anteriormente equalizada, obteve-se a imagem borrada presente na Figura 11.

Figura 11: Imagem suavizada por meio da aplicação de um filtro Gaussiano.

Para a detecção de bordas, o primeiro passo foi a obtenção dos gradientes de intensidade em x e y, procedimento que foi feito por meio da convolução da imagem com filtros de Sobel, (5), como mostrado em (6) e (7), onde I é a matriz de pixels da imagem.

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}; \quad K_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & -1 \end{bmatrix} \tag{5}$$

$$I_x = K_x * I_b \therefore I_x(i,j) = \sum_{n=0}^{N\cdot 2+1}\sum_{m=0}^{M\cdot 2+1} K_x(n,m) \cdot I_b(i-N+n, j-M+m) \tag{6}$$

$$I_y = K_y * I_b \therefore I_y(i,j) = \sum_{n=0}^{N\cdot 2+1} \sum_{m=0}^{M\cdot 2+1} K_y(n,m) \cdot I_b(i-N+n, j-M+m) \tag{7}$$

Com a obtenção dos gradientes de intensidade nas duas direções, foi calculado gradiente da imagem, contendo a matriz de intensidades, G, e a matriz de direções, θ, como mostrado em (8).

$$G = \sqrt{I_x^2 + I_y^2}; \quad \theta = \arctan\left(\frac{I_y}{I_x}\right) \tag{8}$$

Para a reconstrução da imagem que contém somente as bordas, a imagem foi refinada por meio da escolha do pixel que melhor representa a borda, comparando a maior intensidade do gradiente em cada ponto com a sua direção, e dessa forma comparado a um limiar, saturou-se a intensidade do pixel em 0 ou 255, gerando assim uma imagem binária como mostrado na Figura 12. Esse limiar pode ser modificado para calibrar a sensibilidade do algoritmo às bordas detectadas.

Figura 12: Bordas detectadas por meio do método de Canny.

Para melhoria da informação entregue pela detecção de bordas, foi aplicado máscaras de erosão e dilatação, conforme as Figuras 13 e 14, respectivamente. Pode-se observar que com a erosão, algumas bordas mais finas se tornam falhadas, porém vários ruídos da imagem são removidos. Já ao dilatar a imagem, as bordas se tornam mais definidas e contínuas, porém também amplifica os ruídos da detecção.

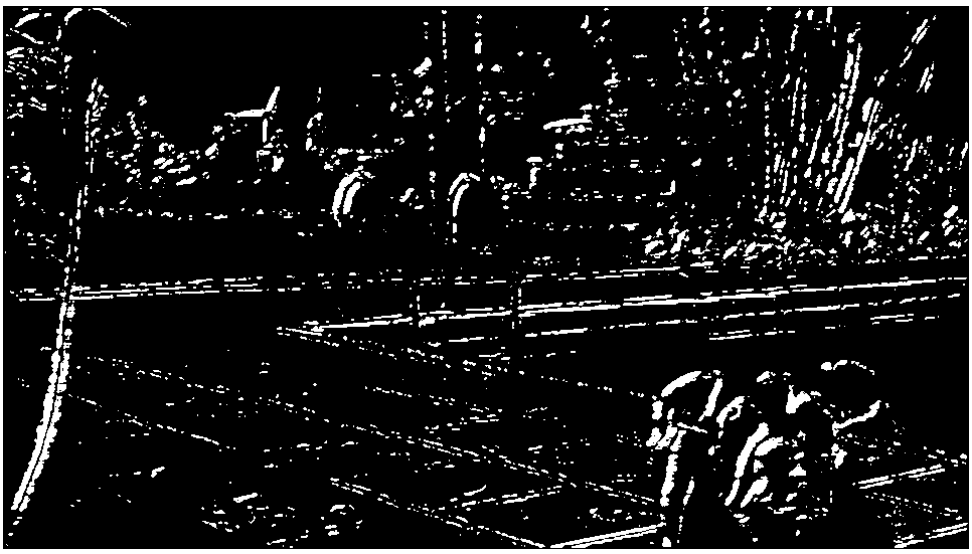


Figura 13: Bordas detectadas por meio do método de Canny com aplicação de máscara de erosão.

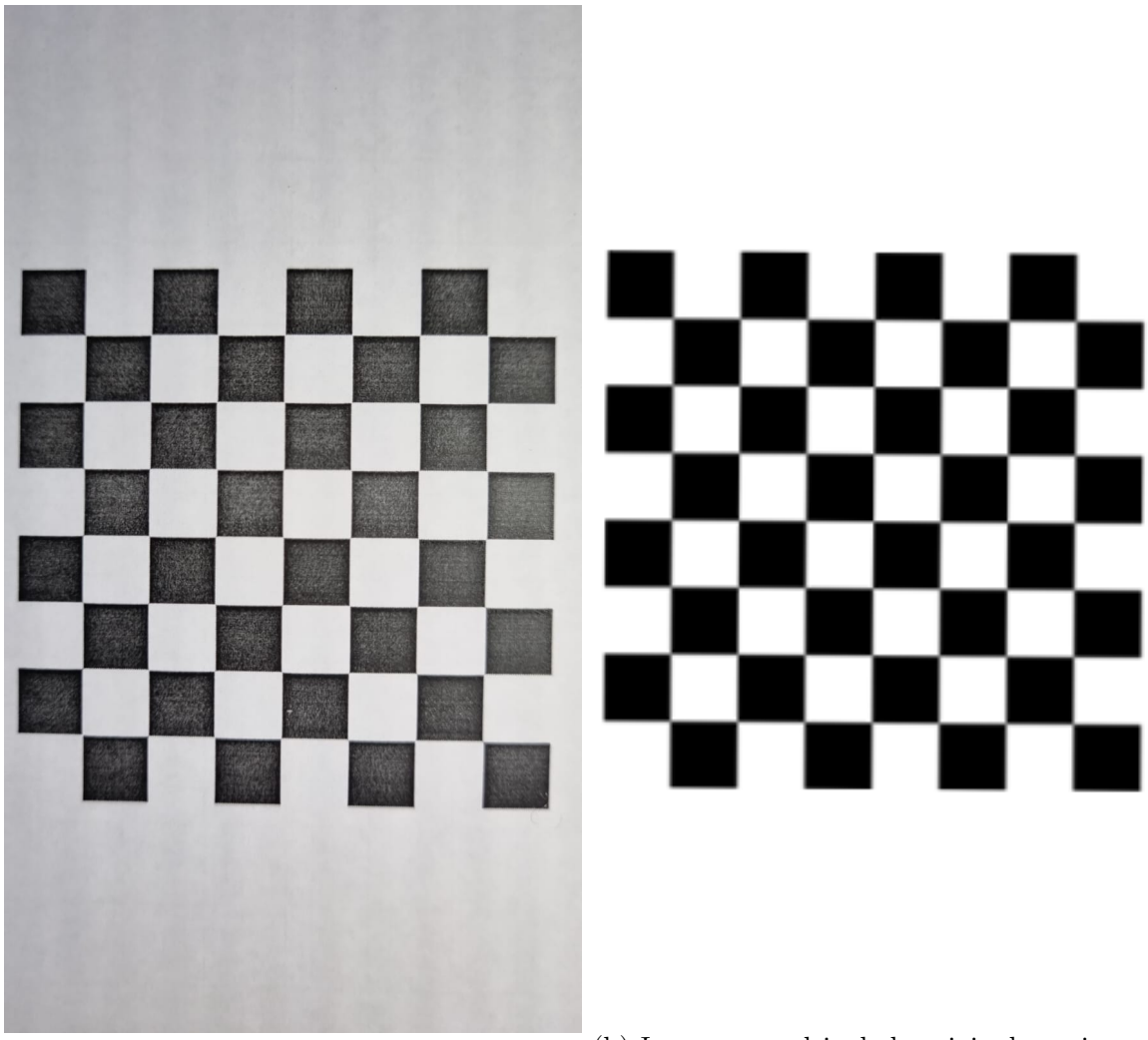
Figura 14: Bordas detectadas por meio do método de Canny com aplicação de máscara de dilatação.

6 Correção de distorção

Com a inserção de lentes no modelo da câmera *pinhole*, é possível além da obtenção de mais luz dentro da câmara fotográfica, controlar também a distância focal. Porém, as lentes inserem distorções na imagem, que podem ser corrigidas por meio de algoritmos computacionais.

Para a obtenção da demonstração de como ocorre a distorção, foi impressa uma imagem quadriculada, e essa capturada por meio de uma câmera paralela à folha de papel, o mais próximo possível, obtendo assim a Figura 15a. Observa-se que nas extremidades da imagem, o quadriculado toma um aspecto arredondado, causado pela distorção esférica da lente.

A Figura 15b é uma manipulação do tabuleiro quadriculado impresso, sobreposto à imagem capturada para servir de gabarito para a correção de deformação, onde o quadriculado se encontra na mesma posição e rotação visto na imagem com deformação.



(a) Imagem quadriculada obtida com distorções da câmera.

(b) Imagem quadriculada original manipulada para a mesma posição e rotação da imagem capturada com deformações.

Figura 15: Imagens utilizadas no processo de correção de distorção.

A imagem capturada foi tratada para a realização do processamento de correção de distorção. Para facilitar a identificação do padrão quadriculado, a mesma teve seu tamanho reduzido

à 25% da original, reduzindo o custo computacional do seu processamento, além da reconstrução em escala de cinzas, para aplicação do filtro binário e da máscara de erosão, obtendo assim uma imagem binária com boa definição do padrão quadriculado, e mantendo a distorção gerada pela câmera.

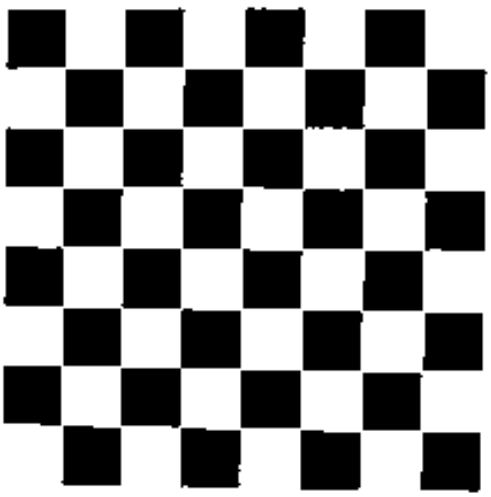
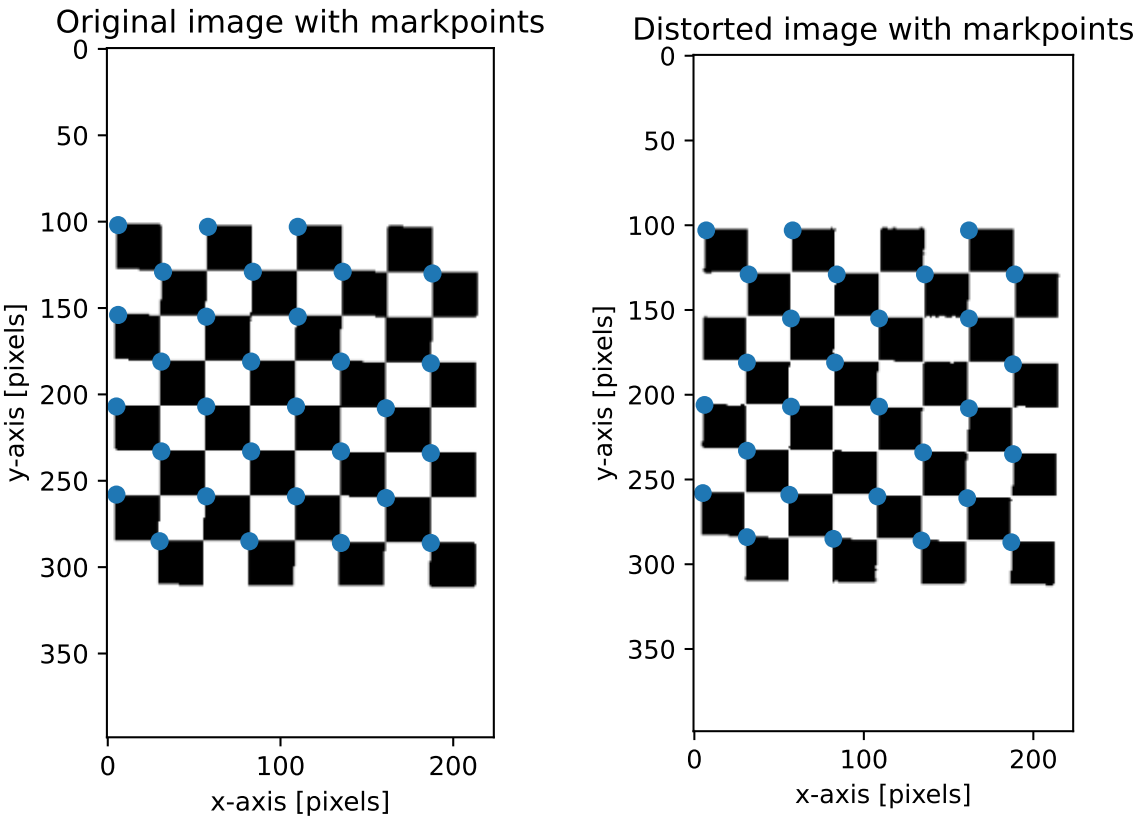


Figura 16: Imagem capturada com distorções após o redução, reconstrução em escala de cinzas, aplicação de filtro binário e erosão.

A heurística aplicada para a correção de distorção se dá pela detecção dos pontos superiores esquerdos dos quadrados pretos da imagem, por meio da convolução da imagem com o padrão original, e com o padrão distorcido pelo filtro (9). Ao detectar o ponto de interesse, esse filtro retorna o valor 4, e o *markpoint* é salvo. A Figura 17a mostra a obtenção dos *markpoints* para a imagem original, enquanto a Figura 17b para a imagem com distorções.

$$K = \begin{bmatrix} 0 & 1 & 1 \\ 1 & -1 & -1 \\ 1 & -1 & -1 \end{bmatrix} \tag{9}$$



(a) Imagem original com *markpoints* para comparação com a imagem distorcida.

(b) Imagem distorcida com *markpoints* para detecção da distorção.

Figura 17: Adição de *markpoints* às imagens original e distorcida.

Realizando a catalogação desses pontos, agrupando-os por pares, é possível obter a posição de onde o ponto se encontra na imagem distorcida, e onde ele deveria estar posicionado, de acordo com a imagem original de gabarito. A Figura 18 exibe a imagem distorcida, com a sobreposição dos *markpoints*, e a definição do centro da imagem. A partir de tais dados, se faz possível a obtenção dos coeficientes da câmera, e da reconstrução da imagem original por meio da correção de distorções da imagem capturada pela câmera.

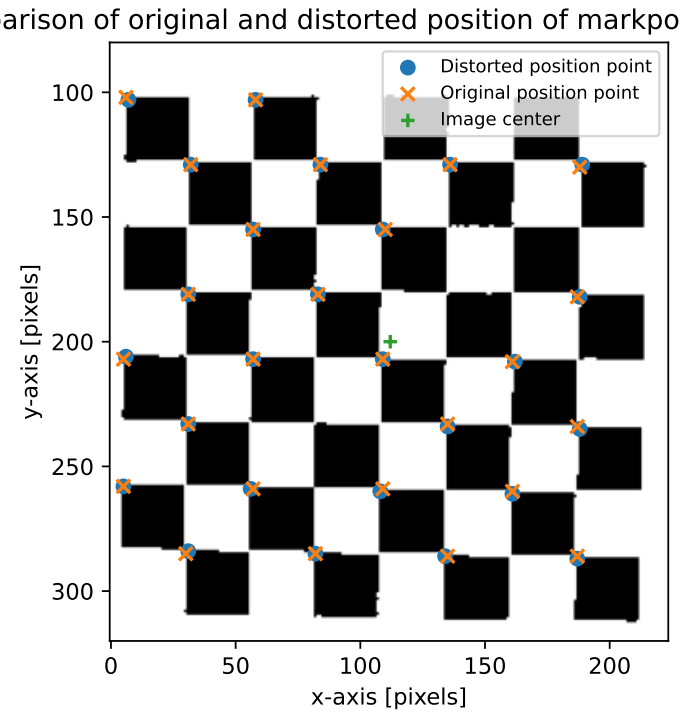


Figura 18: Sobreposição dos *markpoints* da imagem original e distorcidas na imagem com distorções.

Segundo Kurka e Díaz Salazar (2019), as equações de distorções em imagens são dadas por (10) e (11).

$$x_u = (x_d - \overline{x_d})K(r) + \left[P_1\left(r^2 + 2\cdot(x_d - \overline{x_d})^2\right) + 2\cdot P_2(x_d - \overline{x_d})(y_d - \overline{y_d})\right]P(r) \tag{10}$$

$$y_u = (y_d - \overline{y_d})K(r) + \left[P_2\left(r^2 + 2\cdot(y_d - \overline{y_d})^2\right) + 2\cdot P_1(x_d - \overline{x_d})(y_d - \overline{y_d})\right]P(r) \tag{11}$$

Onde:

$$r = \sqrt{(x_d - \overline{x_d})^2 + (y_d - \overline{y_d})^2} \tag{12}$$

$$K(r) = 1 + \sum_{k=1}^{\infty} K_k \cdot r^{2k} \tag{13}$$

$$P(r) = 1 + \sum_{k=3}^{\infty} P_k \cdot r^{2k(k-2)} \tag{14}$$

Fazendo $P_k = 0 \ \forall \ k \geq 3$ e $K_k = 0 \ \forall \ k \geq 3$, e sendo $\Delta a_d = a_d - \overline{a_d}$, para a uma variável qualquer, pode-se reescrever (10) e (11) como:

$$x_u - \Delta x_d = \Delta x_d \cdot K_1 \cdot r^2 + \Delta x_d \cdot K_2 \cdot r^4 + P_1\left(r^2 + 2\cdot(\Delta x_d)^2\right) + 2\cdot P_2 \cdot \Delta x_d \cdot \Delta y_d \tag{15}$$

$$y_u - \Delta y_d = \Delta y_d \cdot K_1 \cdot r^2 + \Delta y_d \cdot K_2 \cdot r^4 + P_2 \left(r^2 + 2 \cdot (\Delta y_d)^2\right) + 2 \cdot P_1 \cdot \Delta x_d \cdot \Delta y_d \tag{16}$$

Considerando cada um dos N pares de *markpoints*, mostrados na Figura 18 pode-se montar o modelo linear $\mathbf{Y} = \mathbf{A}\mathbf{w}$ da (17).

$$\begin{bmatrix} x_{u_1} - \Delta x_{d_1} \\ y_{u_1} - \Delta y_{d_1} \\ \vdots \\ x_{u_N} - \Delta x_{d_N} \\ y_{u_N} - \Delta y_{d_N} \end{bmatrix} = \begin{bmatrix} \Delta x_{d_1} r^2 & \Delta x_{d_1} r^4 & r^2 + 2\Delta x_{d_1}^2 & 2\Delta x_{d_1} \Delta y_{d_1} \\ \Delta y_{d_1} r^2 & \Delta y_{d_1} r^4 & 2\Delta x_{d_1} \Delta y_{d_1} & r^2 + 2\Delta y_{d_1}^2 \\ \vdots & \vdots & \vdots & \vdots \\ \Delta x_{d_N} r^2 & \Delta x_{d_N} r^4 & r^2 + 2\Delta x_{d_N}^2 & 2\Delta x_{d_N} \Delta y_{d_N} \\ \Delta y_{d_N} r^2 & \Delta y_{d_N} r^4 & 2\Delta x_{d_N} \Delta y_{d_N} & r^2 + 2\Delta y_{d_N}^2 \end{bmatrix} \times \begin{bmatrix} K_1 \\ K_2 \\ P_1 \\ P_2 \end{bmatrix} \tag{17}$$

Obtém-se os parâmetros de correção por meio da aplicação da pseudo-inversa, como mostrado em (18).

$$\mathbf{w} = \texttt{pinv}(\mathbf{A}) \times \mathbf{Y} \tag{18}$$

Por meio da resolução do sistema linear, têm-se que os parâmetros de correção da lente são:

$$K_1 = -1,0106 \cdot 10^{-4}; \quad K_2 = 9,7659 \cdot 10^{-9}$$
$$P_1 = 4,9843 \cdot 10^{-3}; \quad P_2 = 8,3069 \cdot 10^{-3}$$

Aplicando a correção de deformação, por meio de (10) e (11) observa-se na Figura 19 que a sensação de inflado no quadriculado é corrigida, porém ao custo da perda de informações na parte direita da imagem.

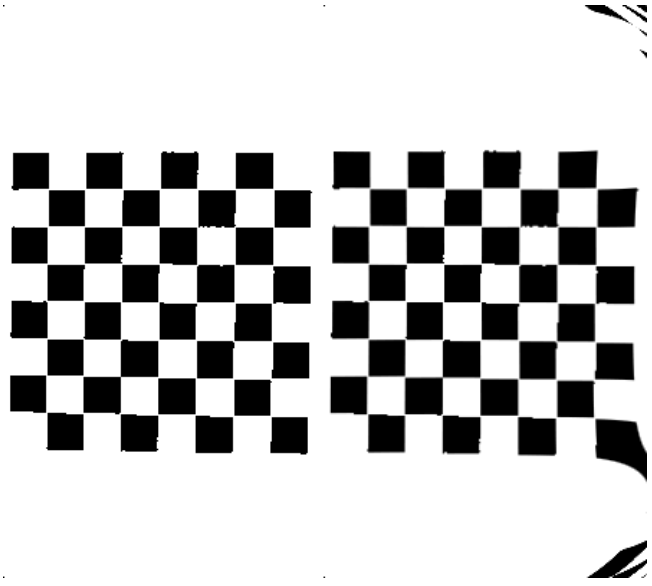


Figura 19: Comparação da imagem com deformação e a corrigida.

7 OCR – *Optical Caracter Recognizer*

O reconhecimento óptico de caracteres, OCR (do inglês *Optical Caracter Recognizer*), é uma ferramenta de grande utilidade para a extração de textos de imagens obtidas por meio de *scanners* ou câmeras, e hoje vem se mostrando utilizado até para recursos de pesquisa em tempo real com base na imagem apresentada na tela de computadores e *smartphones*.

Para o reconhecimento dos caracteres em uma imagem, propõem-se a abordagem em duas etapas: (i) Identificação das caracteres na imagem; (ii) Reconhecimento das caracteres encontradas.

A imagem obtida via câmera foi pré-processada para a identificação das caracteres, sendo aplicado uma máscara binária que representa a informação da imagem (texto) em branco, facilitando assim o processamento. Para identificar as caracteres, foi utilizada a ideia da análise de componentes conectados, onde, considera cada caractere como um objeto onde todos seus pixels são conectados, e com isso, cada segmento interconectado de pixels é considerado uma caractere. Essa análise se baseia na teoria de grafos, por meio da busca e classificação de ramos não-conectos do grafo formado pela matriz de pixels. A aplicação do método foi feita com apoio da biblioteca `scikit-image`, para simplificação da implementação computacional.

7.1 Imagem sem acentuação e fonte monoespaçada

Para um teste inicial do reconhecimento de caracteres, foi utilizado um texto sem acentuação com fonte monoespaçada, o que garante uma maior distância entre os caracteres, e uma regularidade entre o espaçamento. A imagem utilizada é exibida na Figura 20, e a mesma conta com 105 caracteres, considerando letras e pontuações.

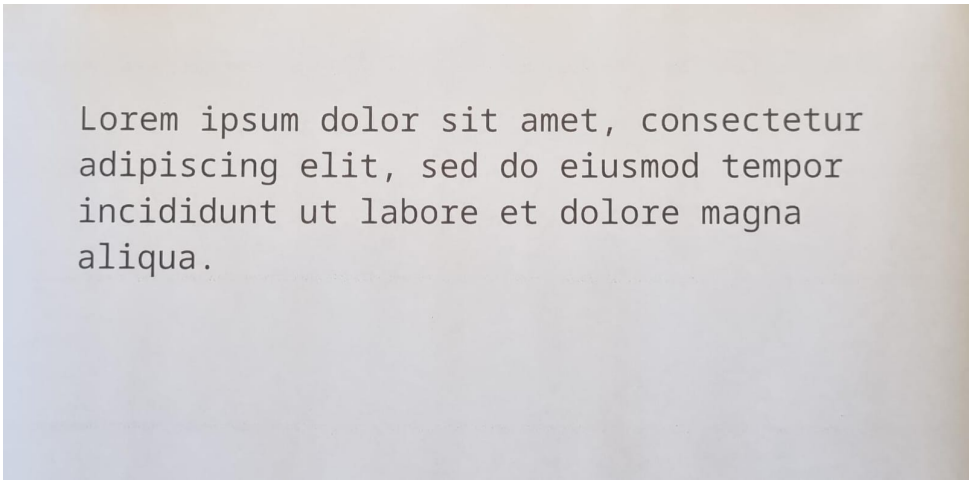


Figura 20: Texto com letras acentuadas capturado para identificação dos caracteres.

Reconstruindo a imagem em escala de cinzas e aplicando uma máscara binária, chega-se a imagem da Figura 21. Por meio de uma análise preliminar da representação do texto na imagem tratada, não se julgou necessário a aplicação de máscaras de erosão ou dilatação, uma vez que se vê uma boa definição dos caracteres, sem falhas que podem atrapalhar o algoritmo de busca de componentes conectos.

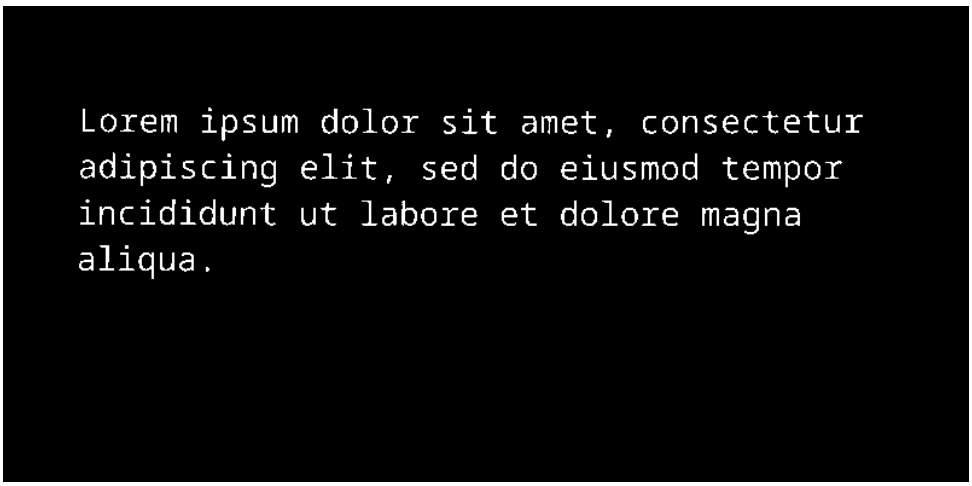


Figura 21: Texto com letras acentuadas após aplicação da máscara binária.

Aplicando o algoritmo de rotulação de componentes conectos, obtém-se na Figura 22 a identificação dos caracteres, onde cada cor representa um carácter diferente. Observa-se que o algoritmo conseguiu separar bem as letras e pontuações, porém também considerou o pingo da letra "i"como um carácter separado da própria letra, chegando a uma contagem de 116 caracteres, uma vez que existem exatamente 11 ocorrências da letra "i"no texto. Nesse caso, a taxa de acurácia na identificação do número de caracteres é de 89,52%.

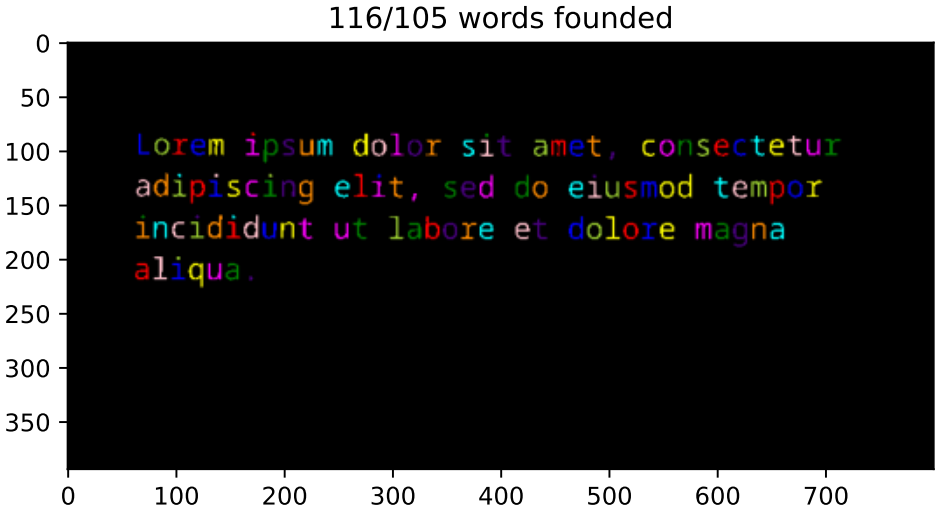


Figura 22: Identificação das caracteres por meio do método de analise de componentes conectadas.

Para suprimir esse problema, foi considerado a aplicação de um limiar para a contagem dos caracteres, por meio da sua área em pixels. Realizando uma análise da imagem, constatou-se que todos os caracteres do texto capturado possuem uma área maior que 8 pixels, enquanto o pingo das letras "i"possuem área menor que 8 pixels. Dessa forma, ao considerar apenas os componentes com área maior que esse limiar, chega-se ao resultado na Figura 23, onde a taxa de acerto é de 100%.

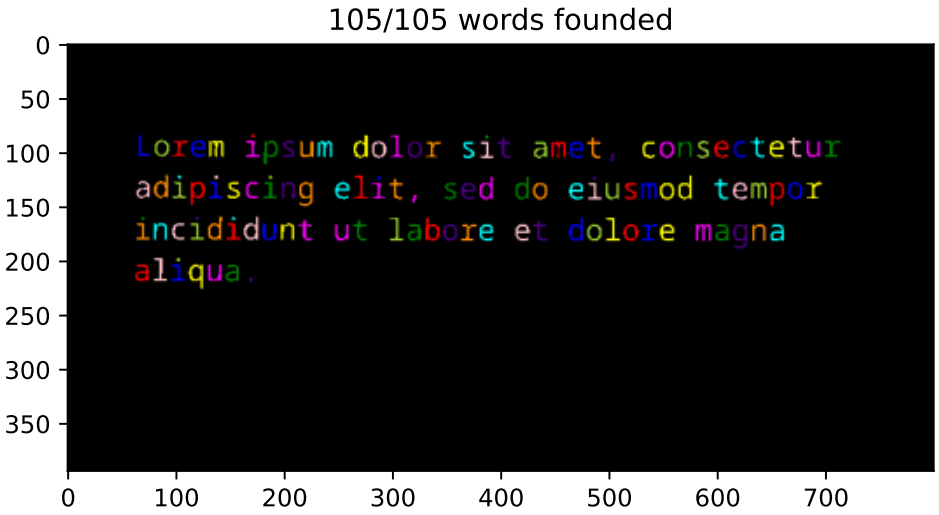


Figura 23: Identificação das caracteres por meio do método de analise de componentes conectadas desconsiderando os objetos com área menor que 8 pixels.

7.2 Imagem acentuada com fonte Latin Modern

Foi realizado a identificação de texto para a imagem mostrada na Figura 24, onde o texto possuí caracteres especiais, e letras acentuadas, além de utilizar uma fonte que não apresnenta um espaçamento regular entre os caracteres. Com a reconstrução da imagem em escala de cinzas, e a aplicação da máscara binária, obtém-se como resultado a Figura 25, a qual está pronta para o processamento com o algoritmo de componentes conectas.

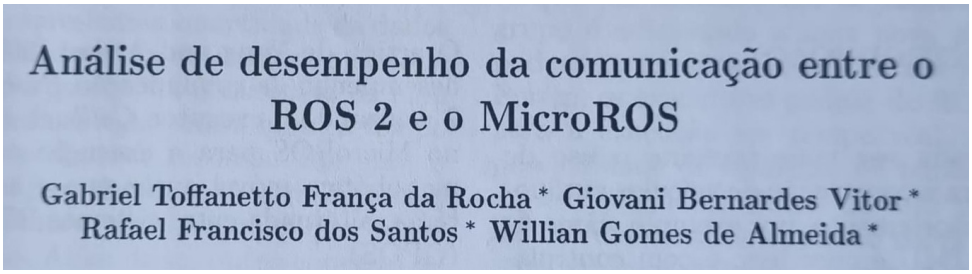


Figura 24: Texto com letras acentuadas capturado para identificação dos caracteres.

Com a análise da imagem binária, não se julgou interessante a aplicação de máscaras de erosão ou dilatação, uma vez que no caso da dilatação, ocorreu a junção de caracteres, enquanto para a erosão as caracteres foram segmentadas. Os dois casos apresentados são problemas para o algoritmo de componentes conectados, onde ele pode considerar dois caracteres como um só, ou classificar o carácter segmentado como vários caracteres diferentes.

Figura 25: Texto com letras acentuadas após aplicação da máscara binária.

Aplicando o algoritmo, observa-se que acentos e pingos das letras "i"são classificadas como caracteres separadas de suas respectivas letras. Além disso, ocorrem letras que devido a sua proximidade ou ruídos de processamento, são identificadas como só um carácter. Dessa forma, o algoritmo conseguiu para esse caso identificar 143 de 152 caracteres, apresentando uma taxa de acurácia de 94,01%.

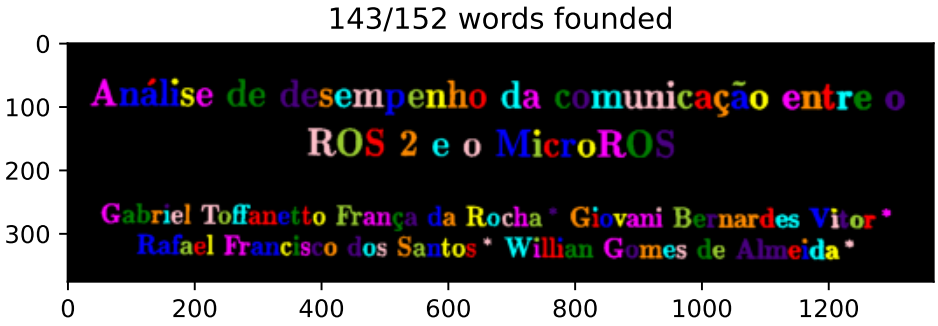


Figura 26: Identificação das caracteres por meio do método de analise de componentes conectadas.

7.3 Discussão

Observa-se que o reconhecimento óptico de caracteres é um problema de grande importância, e que pode utilizar de diversas ferramentas para sua solução, desde técnicas de Álgebra Linear e Teoria de Grafos, até o uso de redes neurais profundas. A ferramenta utilizada se liga diretamente ao resultado obtido.

Com o uso da Teoria de Grafos, por meio da análise de componentes conectos, conseguiu-se separar as caracteres, porém se observa que acentos e pingo das letras "i"não são considerados, e que além disso, caso as letras se apresentem muito próximas, ou ocorra algum ruído que as liguem, o algoritmo tende a falhar. Por outro lado, ao treinar redes neurais profundas para resolução do problema, pode-se conseguir uma solução mais robusta, desde que haja dados suficientes para descrever os caracteres de diversas fontes, e em diversas situações, incluindo por exemplo a presença de ruído e deformações.

Referências

CANNY, J. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8, n. 6, p. 679–698, 1986.

KURKA, P. R. G.; Díaz Salazar, A. A. Applications of image processing in robotics and instrumentation. *Mechanical Systems and Signal Processing*, v. 124, p. 142–169, 2019. ISSN 0888-3270. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0888327019300160>.

Anexos

Códigos fonte

Todos os códigos fonte e arquivos de dados utilizados para a elaboração deste documento podem ser encontrados no repositório do GitHub no link: github.com/toffanetto/im563.