LÓGICA FUZZY E PROCESSAMENTO DE IMAGEM APLICADOS AO MONITORAMENTO DA QUALIDADE NA INDÚSTRIA ALIMENTÍCIA

Ítalo T. F. da Silva[1], Thadeu R. B. Milfont[1], Ivan Mezzomo[1], Pedro L. de S. Andrade[1]

[1]Universidade Federal Rural do Semi-Árido-UFERSA, Mossoró-RN, Brasil

(italo.silva68394@alunos.ufersa.edu.br)

Resumo: A segurança dos alimentos é um pilar fundamental para a saúde pública e para a sustentabilidade econômica das indústrias do setor. A presença de contaminantes, sejam eles biológicos, químicos ou físicos, apresenta riscos à saúde dos consumidores e pode gerar perdas financeiras significativas. O sistema de Análise de Perigos e Pontos Críticos de Controle (APPCC) é a abordagem científica e preventiva reconhecida internacionalmente para mitigar esses riscos, focando na prevenção durante o processo produtivo. Este trabalho propõe a arquitetura de um controlador baseado em lógica fuzzy para detecção de contaminantes do tipo físico, para atuar como um Ponto Crítico de Controle (PCC) automatizado no monitoramento de processos produtivos. A metodologia utiliza técnicas de visão computacional que podem analisar produtos alimentícios em um transportador do tipo esteira, nesse tipo de equipamento o produto fica distribuído de forma uniforme podendo ser analisado pelo sistema. O sistema compara imagens em tempo real com um banco de dados de referência "Gabarito" usando dois critérios principais: verificando similaridade de bytes, medida pela Distância de Hamming, e o método de cor RGB. Estes dados de entrada são fuzzificados e processados por um conjunto de nove regras de inferência "SE", "E" e "$ENTÃO$", onde a lógica "E" é modelada pela t-norma de intersecção (mínimo). Dessa forma o sistema classifica o nível de qualidade do produto em "Aprovado", "Alerta" ou "Reprovado". A abordagem demonstra a capacidade da lógica fuzzy de lidar com a incerteza e a imprecisão inerentes aos processos de inspeção complexos.

Palavras-chave: Lógica Fuzzy; Segurança de Alimentos; Processamento de Imagem; APPCC.

INTRODUÇÃO

A segurança dos alimentos nos processos produtivos é um pilar importante para a saúde pública e econômica das empresas (CODEX ALIMENTARIUS, 2020). Contaminantes nos alimentos têm a capacidade de lesar a saúde dos consumidores e podem influenciar o aspecto econômico da indústria alimentícia, gerando consequências como perdas financeiras, danos à imagem e recolhimento de produtos (SILVA JÚNIOR, 2014). Em processos industriais de beneficiamento e envase de alimentos, podem existir perigos do tipo biológicos, químicos e físicos (BRASIL, 2004). Os perigos biológicos podem ser microrganismos patogênicos como bactérias, vírus, parasitas e fungos (FORSYTHE, 2013); Perigos químicos incluem substâncias químicas prejudiciais à saúde: pesticidas, antibióticos, produtos de limpeza, aditivos em excesso e metais pesados (SILVA JÚNIOR, 2014). Os perigos físicos são quaisquer objetos estranhos que não deveriam estar no alimento e tem potencial de causar danos à saúde do consumidor, esses materiais podem ser: fragmentos de vidro, metal, plástico, madeira, pedras e cabelos (CODEX ALIMENTARIUS, 2020).

Os riscos na produção de alimentos são mitigados pelos Programas de Pré-Requisitos (PPRs) (FORSYTHE, 2013) e pela aplicação do sistema de Análise de Perigos e Pontos Críticos de Controle (APPCC). O sistema APPCC foi desenvolvido nos anos 1960 pela NASA, em parceria com a Pillsbury Company, para garantir que os alimentos para as missões espaciais fossem 100% seguros e tornou-se um padrão global essencial para a indústria alimentícia, e se concentra na prevenção de perigos em vez da inspeção do produto final (SILVA JÚNIOR, 2014; CODEX ALIMENTARIUS, 2020). Um dos princípios do plano APPCC é a Determinação dos Pontos Críticos de Controle (PCCs), que identifica pontos no processo onde o

controle é essencial para prevenir, eliminar ou reduzir o perigo a um nível aceitável.

A lógica fuzzy, introduzida por Lotfi A. Zadeh em 1965, é uma extensão da lógica booleana clássica que lida com o conceito de verdade parcial. Em vez de variáveis serem estritamente verdadeiras (1) ou falsas (0), a lógica fuzzy permite valores no intervalo [0, 1], chamados de graus de pertinência. Essa característica gera a capacidade de lidar com a incerteza e a imprecisão da linguagem humana e de sistemas complexos, sendo ideal para a modelagem de controle e processos de tomada de decisão (ZADEH, 1965). Nas teorias de controle clássicas, é necessário desenvolver um modelo matemático preciso do processo, o que nem sempre é factível se o processo é muito complicado ou se o conhecimento disponível é de forma qualitativa.

Desse modo, um controlador fuzzy pode ser utilizado para o monitoramento de processos produtivos de alimentos, baseado em regras que simulam a capacidade humana de tomar decisões. O objetivo deste trabalho é propor a arquitetura de um controlador fuzzy que possa ser aplicado como um Ponto Crítico de Controle (PCC) automatizado para identificar perigos físicos em tempo real, utilizando processamento de imagem como fonte de dados de entrada para a tomada de decisão.

MATERIAL E MÉTODOS

Nesta seção apresentaremos as definições, conceitos e metódologias: conjuntos fuzzy, operações com conjuntos fuzzy, operadores lógicos e regras de inferência, processos de fuzzificação e defuzzificação, e técnicas de processamento de imagem que serão usadas neste trabalho.

Conjuntos Fuzzy

Um conjunto fuzzy (nebuloso) é uma classe de objetos com graus de pertinência contínuo. Esse conjunto é caracterizado por uma função de pertinência que atribui a cada objeto um grau de pertinência variando entre zero e um (ZADEH, 1965).

Zadeh (1965), define o conjunto fuzzy A sobre um conjunto não vazio X, como uma função de pertinência $f_A(x)$ que associa a cada ponto de X um número real no intervalo [0, 1], onde o valor $f_A(x)$ representa o "grau de pertinência" de x em A. Assim, quanto mais próximo o valor de $f_A(x)$ esteja da unidade, maior será o grau de pertinência de x em A. Quando A é um conjunto no sentido clássico, sua função de pertinência pode assumir apenas dois valores 0 e 1, com $f_A(x) = 1 \; ou \; f_A(x) = 0$ de acordo com o fato de x pertencer ou não pertencer a A. Neste caso, $f_A(x)$ se reduz à função característica familiar de um conjunto A (ZADEH, 1965).

Como nos conjuntos clássicos, a relação de inclusão desempenha um papel central também no caso dos conjuntos fuzzy. Esta noção e as relações de união e interseção são definidas da seguinte forma (ZADEH, 1965):

a) **Contenção:** A está contido em B (ou equivalente A é um subconjunto de B, ou A é menor ou igual a B) se e somente se $f_A \leq f_B$. Em símbolos: $A \subset B \Leftrightarrow f_A \leq f_B$.

b) **União:** A união de dois conjuntos fuzzy A e B com respectivas funções de pertinência $f_A(x)$ e $f_B(x)$ é um conjunto fuzzy C, escrito como $C = A \cup B$, cuja função de pertinência está relacionada às de A e B por: $f_c(x) = Max\left[f_A(x)\,,\, f_B(x)\right],\; x \in X$. Ou, de forma abreviada: $f_c = f_A \vee f_B$.

c) **Interseção:** A interseção de dois conjuntos fuzzy A e B com respectivas funções de pertinência $f_A(x)$ e $f_B(x)$ é um conjunto fuzzy C, escrito como $C = A \cap B$, cuja função de pertinência está relacionada às de A e B por: $f_c(x) = Min\left[f_A(x)\,,\, f_B(x)\right],\; x \in X$. Ou, de forma abreviada: $f_c = f_A \wedge f_B$. É fácil observar que, a união de A e B é o menor conjunto fuzzy que contém ambos os conjuntos A e B, enquanto a interseção de A e B é o maior conjunto fuzzy que está contido em ambos os conjuntos A e B. Como no caso de conjuntos ordinários, A e B são disjuntos se $A \cap B$ é vazio. Note que ∩, assim como ∪, tem a propriedade associativa. A noção de "pertencer", que desempenha um papel fundamental no caso de conjuntos clássicos, não têm o mesmo papel no caso de conjuntos fuzzy. Assim, não faz sentido falar de um ponto x "pertencendo" a um conjunto fuzzy A exceto no sentido trivial de $f_A(x)$ ser positivo, pode-se introduzir dois níveis α e β, onde $0 < \alpha < 1$, $0 < \beta < 1$, $\alpha > \beta$ e concordar em dizer que $f_{A'} = 1 - f_A$ "x pertence a A' se $f_A(x) \geq \alpha$; "x não pertence a A' se $f_A(x) \leq \beta$; e "x tem um status indeterminado relativo a A' se $\beta < f_A(x) < \alpha$.

Operadores e Regras de Inferência

No domínio fuzzy as operações são generalizadas por meio de funções de agregação conhecidas como normas triangulares (t-normas) e conormas triangulares (t-conormas). Segundo Beliakov, Pradera e Calvo (2007), esses operadores não são apenas extensões matemáticas, mas ferramentas para

modelar a imprecisão inerente ao raciocínio humano e aos sistemas de tomada de decisão.

Operadores

a) T-norma: representa o operador de interseção fuzzy, matematicamente, uma t-norma é uma função $T: [0, 1]x[0, 1] \rightarrow [0, 1]$ que deve satisfazer um conjunto rigoroso de axiomas para garantir a consistência lógica (KLIR; YUAN, 1995). O conectivo lógico "E" (intersecção) é modelado por essa classe de funções. Uma t-norma, $T(a,\ b)$, é uma função binária que combina dois graus de pertinência (a, b) em um único valor, representando o resultado da operação "E". Para ser uma t-norma, a função deve satisfazer os seguintes axiomas: condição de fronteira: $T(a,\ 1) = a$ (se uma das variáveis analisadas o grau de pertinência é máximo, ou seja, igual a um, o resultado da intersecção dependerá do grau de pertinência da outra variável a; monotonicidade: Se $a \leq c$ e $b \leq d$, então $T(a, b) \leq T(c, d)$ (garante que o aumento nos graus de pertinência de entrada não resulte em um decréscimo no resultado da agregação); comutatividade: $T(a, b) = T(b, a)$ (ordem das premissas não deve alterar o valor da conjunção); e associatividade: $T(a, T(b, c)) = T(T(a, b), c)$ (Permite a extensão do operador para múltiplos argumentos de forma consistente) (BELIAKOV; CALVO; MESIAR, 2007).
Embora existam várias t-normas, como o produto algébrico $T(a,\ b) = a.b$ e t-norma de Lukasiewicz $T(a,\ b) = max(0,\ a + b - 1)$, a t-norma padrão, escolhida para esta implementação é a t-norma de mínimo: $T(a,\ b) = Min(a, b)$.
b) T-conorma (ou s-normas): são as funções que modelam a união fuzzy, uma t-conorma S satisfaz os mesmos axiomas de monotonicidade, comutatividade e associatividade, porém sua condição de fronteira define 0 como elemento neutro $S(a, 0) = a$. A relação de dualidade expressa pelas Leis de De Morgan $S(a, b) = 1 - T(1 - a, 1 - b)$ permite que, para cada t-norma conhecida, exista uma t-conorma correspondente. Por exemplo, para t-conorma do máximo $S(a,\ b) = Max(a, b)$ é dual t-norma do mínimo, e t-conorma da soma algébrica $S(a,\ b) = a + b - ab$ é dual da t-norma do produto algébrico (BELIAKOV; CALVO; MESIAR, 2007).

Processos de Fuzzificação, Inferência e Defuzzificação

a) Fuzzificação: é a transformação de uma entrada numérica em uma variável linguística definido por um conjunto fuzzy, o grau de pertinência representa o quanto essa entrada pertence a um determinado conceito. Segundo Lee (2005), uma variável linguística é caracterizada por uma quíntupla $(x,\ T(x),\ U,\ G,\ M)$, onde x é o nome da variável; $T(x)$ representa o conjunto de termos linguísticos de x (exemplo: frio, alto, médio); U é o universo do discurso (exemplo: se a variável linguística for temperatura é o intervalo numérico em graus celsius); G é regra sintática que gera os termos em $T(x)$; e M é a regra semântica (significado) que associa a cada termo linguístico um conjunto fuzzy.
b) A inferência opera sobre uma base de regras composta por sentenças condicionais do tipo "SE e ENTÃO". O método de Mamdani é amplamente empregado devido à sua estrutura baseada na composição max-min (LEE, 2005). Para dois conjuntos fuzzy quaisquer (A_i e B_i); $1 \leq i \leq n$, que foram definidos na etapa de fuzzificação, utiliza-se o operador de intersecção (t-norma) o grau de pertinência é definido $\alpha_i = min(\mu_{A_i}(x), \mu_{B_i}(y));\ 1 \leq i \leq n$, o grau de pertinência α_i é aplicado ao conjunto fuzzy consequente da respectiva regra ("ENTÃO") denotado de saída original C_i resultando em um conjunto de saída modificado C'_i. No modelo de Mamdani, utiliza-se a técnica de truncamento: $\mu_{C'_i}(z) = min(\alpha_i,\ \mu_{C_i}(z)); 1 \leq i \leq n.$ Para operadores de união (t-conorma), usualmente o operador máximo, é utilizado para gerar o conjunto fuzzy agregado C:

$$\mu_C(z) = max_{i=1}^{n}\left\{\mu_{C'_i}(z)\right\}.$$

c) Defuzzificação: É o processo de converter o conjunto fuzzy resultante em um valor numérico z *. Segundo Lee (2005) a escolha do método de defuzzificação impacta diretamente a precisão do controle. Os métodos principais são:
1. Centro de Gravidade: Também conhecido como Centróide, é o método mais difundido e calcula o ponto de equilíbrio da área sob a curva de pertinência:

$$z\ *= \frac{\int \mu_C(z).z dz}{\int \mu_C(z) dz};$$

2. Média dos Máximos: Identifica os valores de z que possuem o grau máximo de pertinência e calculam a média aritmética, Se Z for o conjunto de N valores exatos onde a função atinge o seu pico máximo, a defuzzificação é dada por:

$$z\ *= \frac{\sum_{j=1}^{N} z_j}{N};$$

3. Primeiro e Último dos Máximos: Selecionam, respectivamente, o menor ou o maior valor do universo de discurso que atinge a pertinência máxima

no conjunto agregado, definido pelas seguintes equações:
Primeiro dos Máximos:

$$z\ *= min\{z \in Z \,/\, \mu_c(z) = max\mu_c(z)\};$$

Último dos Máximos:

$$z\ *= max\{z \in Z \,/\, \mu_c(z) = max\mu_c(z)\}$$

Processamento De Imagem Para Extração De Características

A tomada de decisão de um controlador fuzzy exige a transformação de informações visuais complexas em dados numéricos de entradas. O estado de conformidade do produto alimentício é avaliado por meio de técnicas de visão computacional que extraem características das amostras inspecionadas. A imagem capturada é processada para isolar duas variáveis independentes que descrevem o produto por sua forma e textura utilizando a Distância de Hamming e a sua correspondência cor utilizando o método RGB.

Hashing Perceptual e Distância de Hamming

Para a avaliação das características estruturais do produto, o sistema utiliza o algoritmo de *Hashing Perceptual* (`pHash`) e a Distância de Hamming.
a) De acordo com Zauner (2010), as funções de hash perceptual produzem valores de *hash* baseados na aparência visual da imagem. Ao contrário das funções de *hash* criptográficas que exigem que cada bit corresponda exatamente ao original e mudam drasticamente com qualquer alteração, uma função de *hash perceptual* calcula valores de *hash* semelhantes para imagens semelhantes.
b) Introduzida originalmente por Richard W. Hamming em 1950, essa métrica surgiu da necessidade de garantir confiabilidade em operações de máquinas de computação de larga escala. Em sua formulação teórica original, Hamming propôs um modelo geométrico inovador, no qual sequências binárias de comprimento n são representadas como vértices de um cubo unitário de n dimensões. Nesse espaço n-dimensional, a distância entre dois pontos é formalmente definida como o número de coordenadas nas quais eles diferem (HAMMING, 1950). Hamming estabelece que distância $D(x,y) = \sum_{i=1}^{n} [x_i - y_i]$ satisfaça três condições:

a) identidade: $D(x,y) = 0,\ se\ x = y$;

b)simetria: $D(x,y) > 0,\ se\ x \neq y$; e

c) desigualdade triangular: $D(z,y) + D(y,z) \geq D(x,z)$.

No contexto de vetores binários h_1 e h_2 de comprimento n (neste sistema, os hashes de 64 bits), a distância d_H é formulada matematicamente pelo somatório da operação lógica utilizada na ciência da computação *Exclusive-OR* ($\oplus$) (abreviação: XOR) entre os bits correspondentes (O XOR compara bit a bit, onde os bits forem idênticos, ele anota 0 e onde houver uma divergência, ele anota 1.): $d_H(h_1, h_2) = \sum_{i=1}^{n} (h_{1,i} \oplus h_{2,i})$.
Nesta métrica, a distância é definida estritamente como o número de posições nas quais os símbolos correspondentes de duas sequências divergem. No contexto do processamento de imagens, após o algoritmo `pHash` extrair as características da imagem e convertê-las em uma "impressão digital", a Distância de Hamming atua contando o número exato de posições em que os *bits* do *hash* capturado em tempo real diferem dos *bits* do *hash* de referência (gabarito). Uma distância baixa caracteriza alta similaridade perceptual entre as imagens, indicando que a forma e a textura do produto inspecionado estão em conformidade com o gabarito (ZAUNER, 2010).

Modelo de Cor RGB e Distância Euclidiana

O modelo de cores RGB (*Red, Green, Blue*) forma um amplo espectro de cores por meio da combinação de diferentes intensidades de luz vermelha, verde e azul. Neste espaço de cor tridimensional, cada canal cromático é quantificado, variando numericamente de 0 (ausência total de intensidade luminosa) a 255 (intensidade máxima). Dessa forma, um pixel preto absoluto é representado pelas coordenadas cartesianas $(0, 0, 0)$, o branco puro por $(255, 255, 255)$, enquanto outras cores assumem vetores específicos, como o verde puro em $(0, 255, 0)$, vermelho puro $(255, 0, 0)$, azul puro $(0, 0, 255)$ ou um tom de amarelo em $(200, 200, 0)$ (FOLEY, 1990).
No sistema proposto, o algoritmo realiza uma varredura na região de interesse e calcula a cor média do produto alimentício, gerando um vetor único (R, G, B) que representa a cor global da amostra. Para quantificar matematicamente a diferença de cor entre a imagem inspecionada em tempo real (R_1, G_1, B_1) e o gabarito de referência (R_2, G_2, B_2), emprega-se a métrica Euclidiana que determina o comprimento do segmento de reta que conecta os dois pontos de cor, sendo a métrica padrão para quantificar a similaridade cromática em processamento de imagens (GONZALEZ; WOODS, 2010), definida pela seguinte equação:

$$d = \sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2}.$$

MÉTODO COMPUTACIONAL APLICADO

Esta seção trata da descrição detalhada do estudo que envolve a coleta, análise e apresentação dos resultados. O sistema proposto consiste em uma arquitetura computacional projetada para atuar como um Ponto Crítico de Controle (PCC) para detecção de contaminantes físicos em alimentos. A arquitetura do software utiliza linguagem de programação Python, integrando bibliotecas de visão computacional como a OpenCV (`cv2`), processamento matemático (`NumPy`) para análise de cor e a Distância Euclidiana, comparação de similaridade visual de imagens (`ImageHash`) utilizado para calcular a Distância de Hamming, (`scikit-fuzzy`) utilizada para criar algoritmos e sistemas de controle utilizando lógica fuzzy, e a biblioteca (`pyModbusTCP`) utilizada para realizar comunicação entre o programa em python e um CLP via Ethernet (TCP/IP).

O método proposta pode ser dividido em quatro passos:

Aquisição de Dados e Construção do Gabarito

O primeiro estágio caracteriza-se pela fase de calibração do sistema. O operador registra amostras físicas do produto alimentício em suas condições ideais de conformidade. Através da interface de vídeo, uma região de interesse (ROI - *Region of Interest*) centralizada de dimensões fixas é delimitada na câmera para padronizar a área de análise e salvando na pasta Gabarito, o trecho do código responsável por essa operação:

```
      # Definição e Recorte da ROI
Centralizada no Modo Captura
      centro_x = w // 2
      centro_y = text_bar_height +
(h - text_bar_height) // 2
      # Cálculo das coordenadas da
caixa de dimensões fixas
      x1, y1 = max(0, centro_x -
TAMANHO_CAIXA_ROI),
max(text_bar_height, centro_y -
TAMANHO_CAIXA_ROI)
      x2, y2 = min(w, centro_x +
TAMANHO_CAIXA_ROI), min(h, centro_y
+ TAMANHO_CAIXA_ROI)
      # Salvamento da amostra de
referência (Gabarito)
      if key == ord('s'):
          roi_captura = frame[y1:y2,
x1:x2]
cv2.imwrite(caminho_salvar,
roi_captura) # Armazenamento físico
da imagem
```

Ao salvar as amostras de referência (gabarito), o sistema processa as imagens extraindo e armazenando dois conjuntos de dados fundamentais:

a) As características estruturais de cada imagem capturada é calculada através do algoritmo de `Hashing` Perceptual e salva, servindo como o padrão geométrico e de textura do produto ideal:

```
      # Extração de Hash Perceptual
para padrão geométrico e de textura
      imagem_pil =
Image.open(caminho_arquivo)
      hash_img =
imagehash.phash(imagem_pil) #
Algoritmo pHash
      hashes_referencia.append(hash_
img)
```

e

b) Referência de cor, o algoritmo calcula a intensidade média dos *pixels* de todas as imagens capturadas, resultando em um vetor tridimensional único, que representa a cor global padrão:

```
      # Cálculo da intensidade média
dos pixels e geração do vetor RGB
único
      media_cores_bgr =
np.mean(imagem_cv, axis=(0, 1))
      media_cores_rgb =
(media_cores_bgr[2],
media_cores_bgr[1],
media_cores_bgr[0])
      rgbs_referencia_lista.append(m
edia_cores_rgb)
      # Consolidação da Cor Global
Padrão (Vetor Médio de todas as
amostras)
      gabarito_rgb_medio =
np.mean(rgbs_referencia_lista,
axis=0)
```

Monitoramento e Extração de Características em Tempo Real

Com o gabarito criado, o sistema faz o monitoramento contínuo capturando quadros da câmera, a captura extrai dois critérios quantitativos simultaneamente, que atuarão como a base para a fuzzificação.

Extração da Distância de Hamming

A extração da Distância de Hamming é executada a cada quadro de vídeo processado, o

algoritmo inicialmente converte a submatriz de *pixels* do padrão nativo da biblioteca `OpenCV` (BGR) para RGB, transformando em um objeto compatível com a biblioteca `ImageHash`. Em seguida, a função `phash` atua gerando a "impressão digital" perceptual de 64 *bits* da cena atual.

A quantificação da Distância de Hamming é obtida através do operador `(-)` da biblioteca `imagehash`, que aplica internamente a operação lógica *Exclusive-OR* (XOR) *bit* a *bit* entre o *hash* atual e a lista de *hashes* de referência armazenada durante o modo de captura. A média aritmética dessas divergências é calculada via biblioteca `NumPy` para gerar o valor:

```
      # Conversão do espaço de cor e formatação da matriz para a biblioteca de hash
      frame_rgb = cv2.cvtColor(frame_processar, cv2.COLOR_BGR2RGB)
      imagem_pil = Image.fromarray(frame_rgb)

      # Extração da assinatura estrutural (64 bits) da imagem em tempo real
      hash_atual = imagehash.phash(imagem_pil)

      # Cálculo iterativo da Distância de Hamming (operação lógica XOR subjacente)
      distancias_hamm = [hash_atual - h_ref for h_ref in hashes_referencia]

      # Obtenção do valor numérico contínuo para a primeira entrada do controlador fuzzy
      dist_hamm_media = np.mean(distancias_hamm)
```

Divergência de Cor RGB

Para comparação de cor, iremos realizar o fracionamento da região de interesse (dividir em várias regiões), pois dependendo do tamanho do contaminante, mesmo com cor diferente do gabarito, a cor média RGB não se distancia do gabarito devido à diluição da cor na média global da imagem, dessa forma a tela é fracionada, e é medido a distância de cor do gabarito em cada tela fracionada, dessa forma evitando diluir a cor do contaminante na média global da imagem. O algoritmo fraciona a Região de Interesse (ROI) capturada em tempo real em 16 quadrantes independentes, utilizando técnicas de fatiamento de matrizes multidimensionais (*`array slicing`*).

Para cada iteração do algoritmo: 1) extrai a submatriz de *pixels* correspondente ao quadrante; 2) calcula as médias cromáticas independentes nos canais R, G e B `(np.mean)`; 3) aplica a métrica vetorial da Distância Euclidiana `(np.linalg.norm)` em relação à cor do gabarito; 4) armazena a divergência em uma estrutura de dados. Ao final da varredura, a função `max()` seleciona a maior distância Euclidiana:

```
      h_roi, w_roi = frame_processar.shape[:2]
      dy, dx = h_roi / 4.0, w_roi / 4.0
      distancias_quadrantes = []

      # Varredura matricial da ROI (4x4)
      for i in range(4):
          for j in range(4):
              # Delimitação espacial das coordenadas do quadrante local
              qx1, qy1 = int(j * dx), int(i * dy)
              qx2, qy2 = int((j + 1) * dx), int((i + 1) * dy)

              # Extração da submatriz e cálculo da cor média RGB
              quadrante = frame_processar[qy1:qy2, qx1:qx2]
              m_bgr = np.mean(quadrante, axis=(0, 1))
              m_rgb = np.array((m_bgr[2], m_bgr[1], m_bgr[0]))

              # Cálculo matemático da Distância Euclidiana contra a referência
              dist_euc = np.linalg.norm(m_rgb - gabarito_rgb_medio)

              distancias_quadrantes.append(dist_euc)

      # Seleção da maior divergência Euclidiana identificada
      dist_rgb_media = max(distancias_quadrantes) if distancias_quadrantes else 0.0
```

Modelagem e Inferência Fuzzy

Após extração dos valores numéricos do Passo 2 (Distância de Hamming e Distância RGB

Máxima), esses dados são utilizados pelo controlador realizando o processo de Fuzzificação, Inferência e Defuzzificação do controlador de qualidade:

Inicialização e Fuzzificação Computacional

A modelagem matemática dos conjuntos fuzzy foi traduzida para o ambiente de software utilizando a biblioteca `scikit-fuzzy`. As variáveis quantitativas extraídas da imagem (Distância de Hamming e Distância Euclidiana RGB) são instanciadas na memória como objetos `Antecedent` (entradas), enquanto o Nível de Qualidade é alocado como um `Consequent` (saída). Na defuzzificação é utilizado o método do centróide geométrico `(defuzzify_method='centroid')`. Os universos de discurso dos gráficos (eixo X) são gerados pela função `np.arange` da biblioteca `NumPy`, para a Distância de Hamming vai ser de 0 a 64 *bits*, e para a Distância RGB, a régua vai de 0 a 150.

Para modelar as funções de pertinência, o sistema utiliza funções trapezoidais (`trapmf`) e triangulares (`trimf`), os pontos de vértices ao longo do eixo X que dar forma aos triângulos e trapézios são representados pelas variáveis (`h_x`) para Distância de Hamming e (`c_x`) para cor RGB, não são fixos no código, são dados de entrada para calibrar o sistema, toda vez que o programa é inicializado, sem precisar reprogramar o sistema, o algoritmo a seguir demonstra essa estrutura, na qual os limites (`h_x` e `c_x`) são recebidos dinamicamente durante a calibração do sistema:

```
      # Definição dos universos de
discurso (intervalos de análise)
      universo_hamm = np.arange(0,
65, 1) # Distância de Hamming (0 a
64 bits)
      universo_rgb = np.arange(0,
151, 1) # Distância Euclidiana (0 a
150)
      universo_qual = np.arange(0,
101, 1) # Nível de Qualidade (0 a
100)

      # Instanciação dos
Antecedentes e Consequente
      dist_hamm =
ctrl.Antecedent(universo_hamm,
'dist_hamm')
      dist_rgb =
ctrl.Antecedent(universo_rgb,
'dist_rgb')
      qualidade =
ctrl.Consequent(universo_qual,
'qualidade',
defuzzify_method='centroid')
      # Mapeamento das funções de
pertinência - Entrada 1 (Hamming)
dist_hamm['Pequena'] =
fuzz.trapmf(universo_hamm, [0, 0,
h_x1, h_x2]) dist_hamm['Média'] =
fuzz.trimf(universo_hamm, [h_x1,
h_x2, h_x3]) dist_hamm['Grande'] =
fuzz.trapmf(universo_hamm, [h_x2,
h_x3, 64, 64])
      # Mapeamento das funções de
pertinência - Entrada 2 (Cor RGB)
dist_rgb['Pequena'] =
fuzz.trapmf(universo_rgb, [0, 0,
c_x1, c_x2]) dist_rgb['Média'] =
fuzz.trimf(universo_rgb, [c_x1,
c_x2, c_x3]) dist_rgb['Grande'] =
fuzz.trapmf(universo_rgb, [c_x2,
c_x3, 150, 150])
```

Base de Regras (Inferência) e Defuzzificação

O controle foi construído com 9 regras, a relação condicional lógica "*E*" foi modelada pela t-norma do mínimo, de acordo com o processo de inferência de Mamdani. A Tabela 1 detalha a matriz de decisão, que origina as regras:

Tabela 1 – Regras de Controle 3x3

		Distância de Cor RGB		
		Pequena	Média	Grande
Distância de Hamming	Pequena	Aprovado	Alerta	Reprovad
	Média	Alerta	Alerta	Reprovad
	Grande	Reprovado	Reprovado	Reprovad

Fonte: Autoria Própria.

Agora o passo consiste na determinação do sinal de controle a ser enviado ao processo, obtido a partir da defuzzificação e método do centro de área. As nove regras são escritas no formato "*SE*", "*E*", e "*ENTÃO*", com base na tabela 3x3, chamaremos a Distância de Hamming de *X*, Distância de Cor RGB de *Y* e Nível de Qualidade de *Z*:

Regra 1: SE *X* é Pequena E *Y* é Pequena ENTÃO o *Z* é Aprovado;
Regra 2: SE *X* é Pequena E *Y* é Média ENTÃO o *Z* é Alerta;
Regra 3: SE *X* é Pequena E *Y* é Grande ENTÃO o *Z* é Reprovado;
Regra 4: SE *X* é Média E *Y* é Pequena ENTÃO o *Z* é Alerta;
Regra 5: SE *X* é Média E *Y* é Média ENTÃO o *Z* é Alerta;

Regra 6: SE *X* é Média E *Y* é Grande ENTÃO o *Z* é Reprovado;
Regra 7: SE *X* é Grande E *Y* é Pequena ENTÃO *Z* é Reprovado;
Regra 8: SE *X* é Grande E *Y* é "Média ENTÃO *Z* é Reprovado;
Regra 9: SE *X* é Grande E *Y* é Grande ENTÃO *Z* é Reprovado.

No código para criação das regras utilizamos a biblioteca `scikit-fuzzy` que implementa internamente o método de Mamdani e a t-norma de intersecção, processando as regras simultaneamente. A lógica condicional é programada através do operador "`&`", (observação: em Python convencionalmente, quando se quer dizer "*A* e *B*", escreve-se a palavra `and`, mas na biblioteca `scikit-fuzzy` utiliza o símbolo `&` que na computação é chamado de operador *bitwise*, quando escreve `dist_hamm['Pequena'] & dist_rgb['Pequena']`, a biblioteca aplica a t-norma do mínimo entre esses dois gráficos, que atua combinando os antecedentes para gerar o grau de ativação do consequente, abaixo segue exemplo do código com essa operação:

```
      # Definição do bloco de Regras de Controle
      regra1 = ctrl.Rule(dist_hamm['Pequena'] & dist_rgb['Pequena'], qualidade['Aprovado'])
      regra2 = ctrl.Rule(dist_hamm['Pequena'] & dist_rgb['Média'], qualidade['Alerta'])
      # ...
      regra9 = ctrl.Rule(dist_hamm['Grande'] & dist_rgb['Grande'], qualidade['Reprovado']).
```

Chamaremos a variável de saída de "Nível de Qualidade" e irá atuar em um universo de 0 à 100, utilizando os termos linguísticos: Reprovado, Alerta e Aprovado. o código abaixo é um exemplo para definir as funções de pertinência do mesmo:

```
      # Mapeamento das funções de pertinência Nível de Qualidade
      qualidade['Reprovado'] = fuzz.trapmf(universo_qual, [0, 0, 50, 65])
      qualidade['Alerta'] = fuzz.trimf(universo_qual, [50, 65, 80])
      qualidade['Aprovado'] = fuzz.trapmf(universo_qual, [65, 80, 100, 100])
```

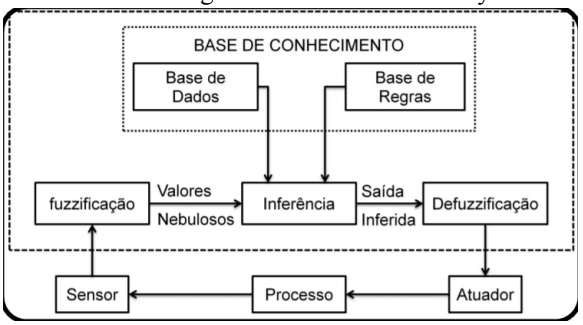
O comando `ctrl.ControlSystem` agrupa as regras formando a base de conhecimento do controlador e o comando `ctrl.ControlSystemSimulation` faz o processamento das regras em tempo real e calculando o centróide em tempo real, abaixo segue exemplo do código para nove regras:

```
      # Construção do Simulador de Controle Fuzzy
      sistema_controle = ctrl.ControlSystem([regra1, regra2, regra3, regra4, regra5, regra6, regra7, regra8, regra9]
      simulador = ctrl.ControlSystemSimulation(sistema_controle)
```

Tomada De Decisão E Atuação No Processo

A arquitetura de um Controlador Fuzzy é apresentada por (SIMÕES; SHAW, 1999) conforme diagrama da Figura 1, estabelece que o sistema computacional deve interagir fisicamente com o ambiente para que o ciclo de automação seja completo, ou seja, para que a qualidade do produto seja garantida, a saída inferida deve ser transmitida a um Atuador, responsável por modificar o estado do Processo. No caso de reprovação, o controlador pode enviar um sinal para um CLP que pode atuar fisicamente no processo, com a rejeição do produto, dessa forma eliminando o risco de contaminação para o consumidor, no estado de Alerta pode-se enviar um sinal para o CLP acionar um sinaleiro luminoso e salvar a imagem desse momento em uma pasta registrando data e hora, sendo possível verificar posteriormente por um operador humano.

Figura 1 - Controlador Fuzzy



Fonte: Adaptado de Simões e Shaw (1999)

A integração de automação fazendo a atuação no processo do sistema de controle de qualidade é realizada por comunicação via rede IP.

Utilizando o pacote `pyModbusTCP`, o *software* atua como um *Master* Modbus, gravando em nível lógico (`0` ou `1`) os registradores responsáveis pela atuação no processo, `0` se o produto for Reprovado e `1` se estiver em estado de Alerta para acionar sinaleira e registrar a imagem:

```
      #CONFIGURAÇÕES DE REDE (CLP
      PLC_HOST = "192.168.1.10"
      PLC_PORT = 502
      # Endereços dos REGISTRADORES
no CLP (ex: N7:0, N7:1)
      ALERT_REGISTER_ADDRESS = 1  #
Registrador para acionar a sinaleira
amarela
      REJECT_REGISTER_ADDRESS = 0 #
Registrador para acionar o
desvio/parada

      #CONFIGURAÇÕES DO CONTROLADOR
FUZZY
      # Limiares da Saída (Nivel de
Qualidade 0-100) para acionamento
      LIMIAR_APROVADO = 65.0  #
Níveis acima deste são "Aprovado"
      LIMIAR_ALERTA = 35.0 # Níveis
entre este e APROVADO são "Alerta"
                           # Níveis
abaixo deste são "Reprovado"

      #TOMADA DE DECISÃO E AÇÕES
(CLP) (MANTIDO CONFORME ORIGINAL)

               if nivel_qualidade >=
LIMIAR_APROVADO:
                   estado =
"APROVADO"
                   sinal_alerta = 0
                   sinal_rejeicao = 0
                   cor_texto = (0,
200, 0)

               elif nivel_qualidade
>= LIMIAR_ALERTA:
                   estado = "ALERTA"
                   sinal_alerta = 1
                   sinal_rejeicao = 0
                   cor_texto = (0,
215, 255)
                   # Salva uma foto
do produto em alerta (COM DATA E
HORA)
                   agora =
datetime.datetime.now().strftime("%Y
%m%d_%H%M%S_%f")
                   nome_alerta =
os.path.join(PASTA_ALERTAS,
f"alerta_{agora}.jpg")
cv2.imwrite(nome_alerta, frame)

               else: #
(nivel_qualidade < LIMIAR_ALERTA)
                   estado =
"REPROVADO"
                   sinal_alerta = 0
                   sinal_rejeicao = 1
                   cor_texto = (0, 0,
255)
               # Envia os sinais para
o CLP (se conectado)
               if client:
                   try:
                       if not
client.is_open:

client.open()

client.write_single_register(ALERT_R
EGISTER_ADDRESS, sinal_alerta)

client.write_single_register(REJECT_
REGISTER_ADDRESS, sinal_rejeicao)
                       sinal_clp_txt
=
f"Alerta(R{ALERT_REGISTER_ADDRESS}):
{sinal_alerta} |
Rejeicao(R{REJECT_REGISTER_ADDRESS})
: {sinal_rejeicao}"
                   except Exception
as e:
                       sinal_clp_txt
= f"ERRO CLP"
                       client.close()
               else:
                   sinal_clp_txt =
"CLP DESCONECTADO"
```

RESULTADOS E DISCUSSÃO

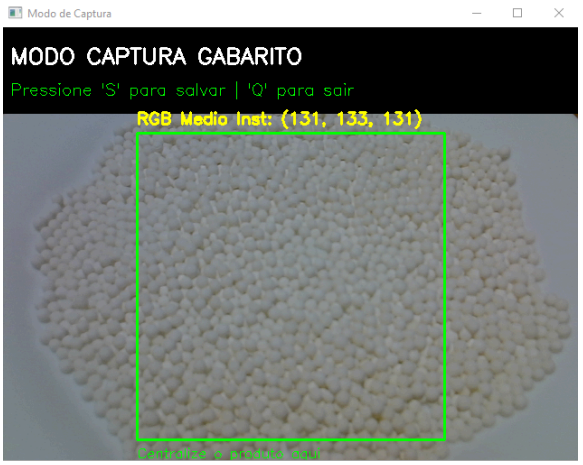
Esta seção apresenta resultados preliminares de simulações controladas para teste e validação do Controlador de Qualidade conforme metodologia proposta para primeira etapa do trabalho. O objetivo primário é avaliar a capacidade do controlador fuzzy de identificar contaminantes físicos de diferentes naturezas (forma, cor e tamanho), processando as operações de fuzzificação, inferência e defuzzificação em tempo real. A segunda etapa do trabalho, que será realizada em uma fase posterior, consiste na aplicação do controlador em um sistema

real de produção de alimentos, com a construção de um sistema de transporte e detecção de anomalias no produto.

Inicialmente, o sistema foi operado no "Modo de Captura" para a construção do banco de dados de referência (Gabarito). Foram capturadas 13 amostras de um produto padrão (granulado de coloração clara e formato esférico uniforme) (Figura 2) e as imagens foram salvas para extração das características na pasta gabarito (Figuras 3). Em seguida é realizada à calibração das funções de pertinência do controlador fuzzy, os valores inseridos foram: a) Para Distância de Hamming**:** O conjunto Pequena foi calibrado com grau máximo até 8 bits, caindo a zero em 16 bits; Média com pico em 16 bits; e Grande a partir de 24 bits. b) Para Distância RGB**:** Para a divergência euclidiana, o limite de aceitação Pequena foi ajustado até o valor 15, com pico do conjunto Média em 25, e consideração de anomalia severa (conjunto Grande) a partir de 35. c) Os limiares de atuação do sistema no Nível de Qualidade também foram estabelecidos. As simulações foram realizadas com o CLP desconectado, o objetivo é validar o Controlador sem atuação no processo, em outro momento será realizado teste com integração a automação de processos.

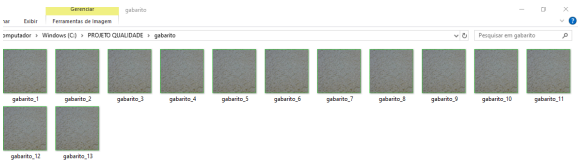
Após configuração do sistema e criação do banco de dados, foi iniciado o monitoramento do processo conforme Figura 4, foram realizadas algumas simulações com implantação de contaminantes físicos que podem surgir em um processo industrial: na Figura 5, foi inserido um objeto do tipo adorno “anel” e houve a Reprovação sendo afetado pela Distância de Hamming com valor de 24 *bits*; na Figura 6, foi inserido um parafuso gerando a Reprovação pela Distância de Cor RGB com valor de 33,4; na Figura 7, foi inserido um pequeno contaminante de cor diferente do produto base, o sistema apresentou estado de Alerta pela Distância de Hamming com 16,5 *bits*. Através das simulações podemos verificar que o Controlador Fuzzy pode ser utilizado como um Ponto Crítico de Controle em um processo industrial de produção de alimentos, apresentou capacidade de aprovar o produto, reprovar e iniciar um estado de alerta.

Figura 2 - Imagem em tempo real da captura de imagens para gabarito



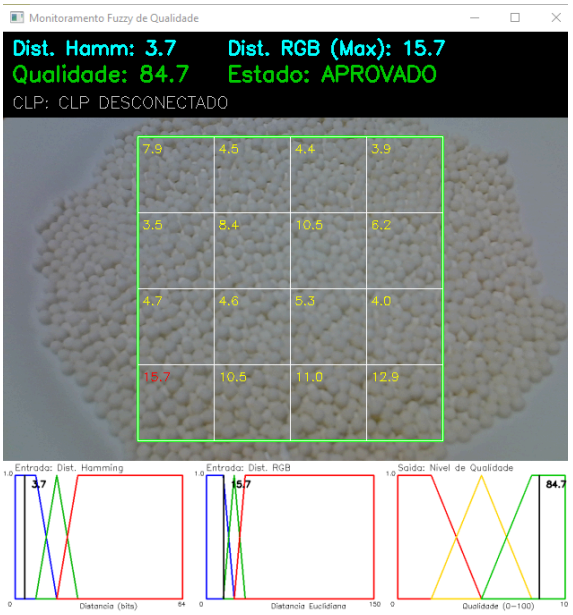
Fonte: Autoria Própria.

Figura 3 - Imagens do produto de referência salvos na pasta gabarito



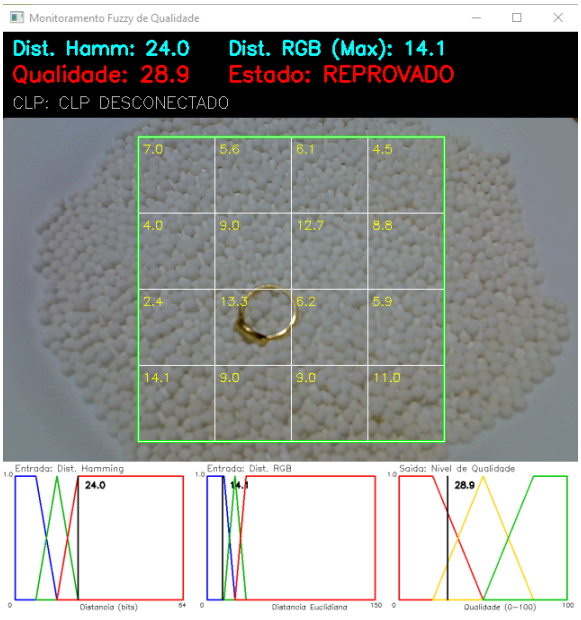
Fonte: Autoria Própria.

Figura 4 - Inicialização do monitoramento online



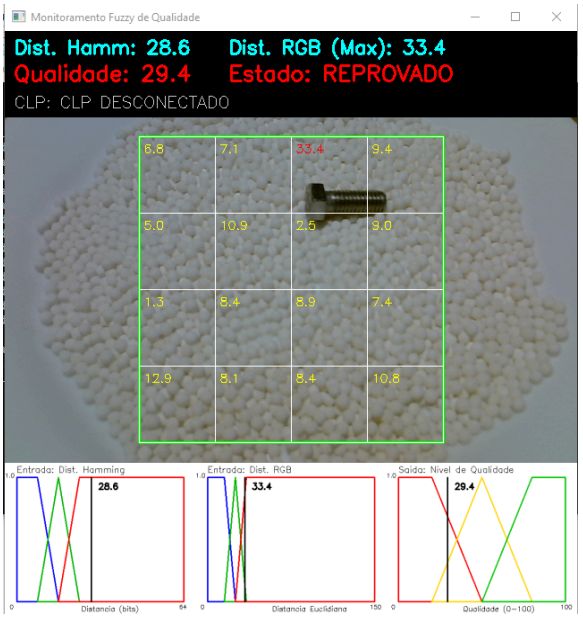
Fonte: Autoria Própria.

Figura 5 - Simulação de Contaminação (Anel)



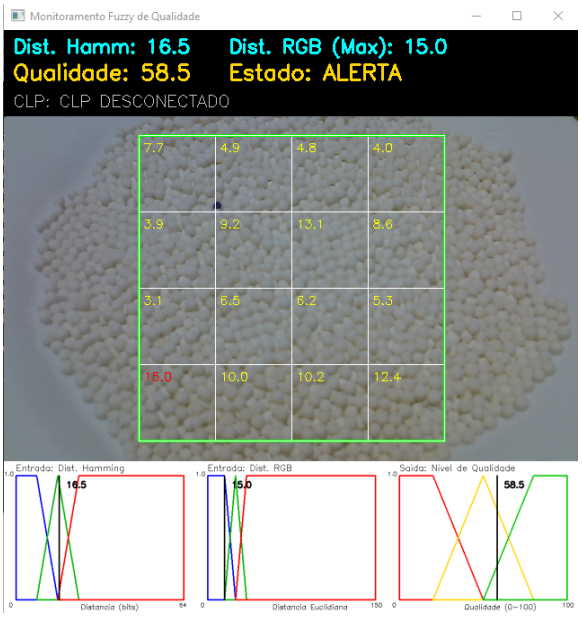
Fonte: Autoria Própria.

Figura 6 - Simulação de Contaminação (Parafuso)



Fonte: Autoria Própria.

Figura 7 - Pequeno Contaminante (esfera roxa)



Fonte: Autoria Própria.

CONCLUSÃO

Este artigo propôs e validou, por meio de simulação, um controlador fuzzy projetado para atuar como um Ponto Crítico de Controle (PCC) automatizado na detecção de perigos físicos na indústria alimentícia. As simulações demonstraram que o sistema, ao aliar a teoria fuzzy ao processamento de imagem foi capaz de atuar de forma assertiva.

O controlador apresentou a capacidade de aprovar produto em conformidade e reprovar produtos que continham contaminações físicas evidentes. É importante ressaltar que a calibração das funções de pertinência, executada no início da simulação, provou ser um fator crítico para o sucesso do sistema. Esta etapa torna o sistema mais sensível, ou não, de modo que pequenas anomalias sejam detectadas de forma eficaz.

Como recomendação para pesquisas futuras, sugere-se a implementação física do sistema com a integração a um controlador lógico programável (CLP), a fim de validar a eficácia dos atuadores em um ambiente fabril. Por fim, recomenda-se que a iluminação do sistema deve ser superior em lúmens em relação à iluminação do ambiente para evitar qualquer interferência externa.

REFERÊNCIAS

BELIAKOV, Gleb. A Practical Guide to Averaging Functions. Springer, Berlin, 2007.

BELIAKOV, Gleb; JAMES, Simon; WU, Jun. Discrete Aggregation Operators: An In-depth Look. Springer, Cham, Switzerland, 2016.

BRASIL. Ministério da Saúde. Agência Nacional de Vigilância Sanitária. Resolução RDC nº 216, de 15 de setembro de 2004. Dispõe sobre Regulamento Técnico de Boas Práticas para Serviços de Alimentação. Diário Oficial da União, Brasília, DF, 16 set. 2004.

CODEX ALIMENTARIUS. General principles of food hygiene (CXC 1-1969). Adopted in 1969. Amended in 1999. Editorial corrections in 2011. Revised in 1997, 2003, 2020. Food and Agriculture Organization of the United Nations / World Health Organization, 2020.

FOLEY, James D.; VAN DAM, Andries; FEINER, Steven K.; HUGHES, John F. Computer Graphics: Principles and Practice. 2. ed. Addison-Wesley, Reading, 1990.

FORSYTHE, S. J. Microbiologia da Segurança dos Alimentos. 2. ed. Artmed, Porto Alegre, 2013.

GOMIDE, Fernando Antonio Campos; GUDWIN, Ricardo Ribeiro. MODELAGEM, CONTROLE, SISTEMAS E LÓGICA FUZZY. SBA Controle & Automação, v. 4, p. 97-115, 1994.

GONZALEZ, R. C.; WOODS, R. E. Processamento Digital de Imagens. 3. ed. Pearson Prentice Hall, São Paulo, 2010.

HAMMING, R. W. Error detecting and error correcting codes. The Bell System Technical Journal, v. 29, p. 147-160, 1950.

KLIR, George J.; YUAN, Bo. Fuzzy Sets and Fuzzy Logic: Theory and Applications. Prentice Hall, Upper Saddle River, NJ, 1995.

LEE, K. H. First Course on Fuzzy Theory and Applications. Springer-Verlag, Berlin, 2005.

SILVA JÚNIOR, E. A. da. Manual de Controle Higiênico-Sanitário em Alimentos. 7. ed. Varela, São Paulo, 2014.

SIMÕES, Marcelo Godoy; SHAW, Ian S. Controle e modelagem Fuzzy. Edgar Blücher, São Paulo, 1999.

ZADEH, L. A. Fuzzy sets. Information and Control, v. 8, p. 338-353, 1965.

ZAUNER, C. Implementation and Benchmarking of Perceptual Image Hash Functions. 2010.