UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Lucas Antonio Demeneck

Seleção de matéria prima em esteira industrial utilizando câmera hiper espectral de infravermelho próximo e ação pneumática

Florianópolis
2022

Lucas Antonio Demeneck

Seleção de matéria prima em esteira industrial utilizando câmera hiper espectral de infravermelho próximo e ação pneumática

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.
Orientador: Prof. Marcelo De Lellis Costa de Oliveira, Dr.
Supervisor: Douglas William Menezes Flores, Dr.

Florianópolis
2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Demeneck, Lucas
Seleção de matéria prima em esteira industrial utilizando câmera hiper espectral de infravermelho próximo e ação pneumática / Lucas Demeneck ; orientador, Marcelo De Lellis Costa de Oliveira, 2022.
53 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2022.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Espectroscopia. 3. Controle de Qualidade. 4. Ciência de dados. 5. Tempo- real. I. De Lellis Costa de Oliveira, Marcelo. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Lucas Antonio Demeneck

Seleção de matéria prima em esteira industrial utilizando câmera hiper espectral de infravermelho próximo e ação pneumática

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 12 de Dezembro de 2022.

Prof. Hector Bessa Silveira, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Marcelo De Lellis Costa de Oliveira , Dr.
Orientador
UFSC/CTC/DAS

Douglas William Menezes Flores, Dr.
Supervisor
Spectral Solutions Comércio e Serviços LTDA

Patrícia Mayer, Me.
Avaliadora
Instituição UFSC/CTC/PosAutomação

Prof. Eduardo Camponogara, Dr.
Presidente da Banca
UFSC/CTC/DAS

RESUMO

A presença de contaminantes ou de alimentos de baixa qualidade em meio à matéria prima utilizada em processos industriais alimentícios pode causar perda de qualidade do produto ou até mesmo torná-lo impróprio para consumo. Para abordar tal problema de forma não invasiva e sem alteração da matéria prima, este trabalho descreve o desenvolvimento de um sistema de seleção que utiliza uma câmera hiper espectral de infravermelho próximo para a identificação de objetos em não conformidade e os remove do processo produtivo por meio de atuadores pneumáticos. Os dados de refletância obtidos pela câmera são preprocessados e inseridos em um modelo de classificação para identificação dos objetos de interesse. Ao fim da esteira a matéria prima considerada adequada deve cair em um silo de onde seguirá para o resto do processo, enquanto bicos de ar posicionados para atuar entre a esteira e o silo devem ser acionados de forma que a trajetória dos objetos não conformantes seja desviada para um depósito de descarte. O sistema deve monitorar a velocidade da esteira para calcular, com base nessa informação e na distância entre o ponto de leitura da câmera e o fim da esteira, o momento correto de atuação dos bicos, havendo portanto requisitos de tempo real a serem atendidos.

Palavras-chave: Espectroscopia. Controle de Qualidade. Ciência de dados. Tempo-real.

ABSTRACT

The presence of contaminants or low-quality ingredients mixed with raw materials used in manufacturing processes of the food industry may lead to products that are of low quality or even inadequate for consumption. To tackle this problem in a noninvasive way and without altering materials, this document describes the development of a sorting system that utilizes a near-infrared hyperspectral camera to identify undesirable objects and remove them from the industrial process through the use of pneumatic actuators. The reflectance data measured by the camera is preprocessed and then inputted into a classification model to identify the relevant objects. When the end of the conveyor belt is reached, the materials should fall into a silo, from where they'll move on to the rest of the process, while air nozzles positioned to release air between the belt and the silo must be activated to divert the trajectory of undesirable objects to a discard pile. The system must monitor the speed of the conveyor belt to calculate, based on this information and the distance between the point where the camera makes its measurements to the end of the belt, the correct moment to activate the nozzles, hence there are real-time requirements to be met.

Keywords: Spectroscopy. Quality Control. Data Science. Real-time.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANN	*Artificial Neural Networks*
API	*Application Programming Interface*
CNN	*Convolutional Neural Network*
CPU	*Central Processing Unit*
CRC	*Cyclic Redundancy Check*
CRUD	*Create Read Update Delete*
FSG	Filtro de Savitzky-Golay
GPU	*Graphical Processing Unit*
HSI	*Hyperspectral Imaging*
KNN	*K Nearest Neighbors*
LDA	*Linear Discriminant Analysis*
MIR	*Medium Infrared*
MVVM	*Model View ViewModel*
NIR	*Near Infrared*
NIRS	*Near Infrared Spectroscopy*
OR	Objeto Rejeitado
PCA	*Principal Component Analysis*
PLS	*Partial Least Squares*
RBF	*Radial Basis Function*
ReLU	*Rectified Linear Unit*
RGB	*Red Green Blue*
SDK	*Software Development Toolkit*
SNV	*Standard Normal Variate*
SVM	*Support Vector Machine*
TCP	*Transmission Control Protocol*
WPF	*Windows Presentation Foundation*
XAML	*Extensible Application Markup Language*
XML	*Extensible Markup Language*

LISTA DE SÍMBOLOS

E_p	Energia de um fóton
h	Constante de Planck
f	Frequência
c	Velocidade da luz
λ	Comprimento de onda
$\mathbf{X}$	Matriz de preditores
n	Número de amostras
p	Número de dimensões
σ	Desvio padrão
$\bar{\mathbf{x}}$	Média de x
$\mathbf{C}$	Matriz de covariância
K	Número de classes
$\mathbf{Y}$	Matriz de variáveis resposta
e	Número de Euler
g	Aceleração da gravidade

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

O controle de qualidade de matéria prima é etapa essencial de diversos processos industriais para assegurar a qualidade do produto final. Na indústria alimentícia esta questão tem importância especial considerando que além da perda de qualidade do produto final, o uso de matéria prima comprometida pode até mesmo torná-lo impróprio para consumo, podendo causar problemas de saúde e até óbitos entre os consumidores, além dos diversos danos à empresa responsável, como danos financeiros, danos de imagem e perda de alvará sanitário.

Para um controle de qualidade eficaz neste ramo da indústria, não basta apenas a remoção de contaminantes e matéria prima de baixa qualidade, também é importante evitar qualquer tipo de alteração na matéria prima de boa qualidade, considerando a fragilidade de diversos alimentos que são utilizados em processos produtivos do setor.

A espectroscopia é amplamente utilizada em aplicações laboratoriais e industriais para analisar a composição química de amostras de interesse, tendo como grandes vantagens a sua capacidade de fazer isso de forma rápida, não destrutiva e não invasiva. A forma como um material reflete, absorve e transmite ondas eletromagnéticas em diferentes frequências é influenciada pelos tipos de átomos que o compõe e as ligações químicas entre eles. Portanto, ao iluminar uma amostra, pode-se mensurar a forma como a luz interage com ela para extrair informações sobre sua composição química.

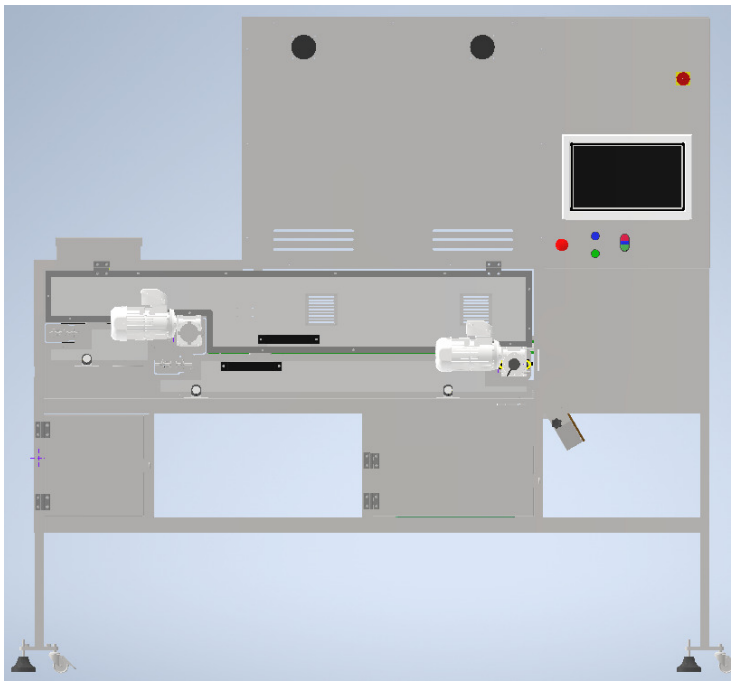
Como evidência da versatilidade e aplicabilidade da espectroscopia de infravermelho próximo, ou *Near Infrared Spectroscopy* (NIRS), na literatura científica podem ser encontrados diversos exemplos de artigos com conclusões positivas em relação ao uso desse método para avaliações qualitativas ou quantitativas de alimentos, como paprica (OLIVEIRA *et al.*, 2020), carnes (PRIETO *et al.*, 2017), cacau (QUELAL-VÁSCONEZ *et al.*, 2020), laranja (BADARÓ *et al.*, 2020), entre outros, assim como aplicações em outras áreas diversas como identificação de resíduos plásticos sólidos (ZHU *et al.*, 2019) e avaliação de qualidade do solo (MOHAMED *et al.*, 2018).

Com base nestas observações, o projeto descrito neste trabalho visa explorar o potencial da NIRS no ambiente industrial para a elaboração de um sistema automatizado de seleção de matéria prima.

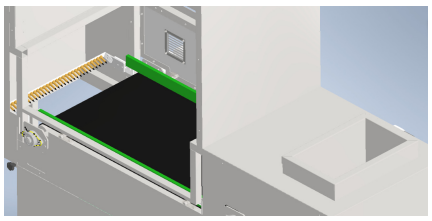
1.2 SOLUÇÃO PROPOSTA

O sistema proposto, ilustrado na figura 1, se baseia na geração de imagens hiper espectrais, ou *Hyperspectral Imaging* (HSI), no espectro de infravermelho próximo, ou *Near Infrared* (NIR). Será utilizada uma câmera hiper espectral posicionada acima

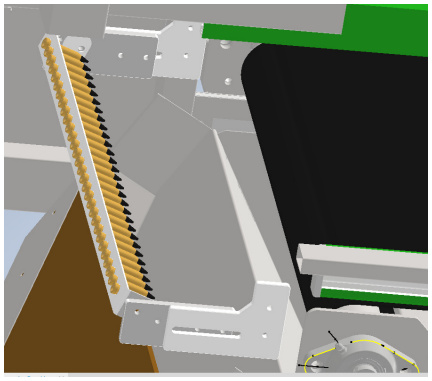
Figura 1 – Projeto da parte física do sistema em software CAD (silos não representados)



(a) Visão frontal. A entrada da matéria-prima ocorre à esquerda, sendo então movida para à direita pelas esteiras. A pilha de descarte se localiza dentro do compartimento abaixo do motor à direita.



(b) Visão da entrada dos produtos do lado oposto



(c) Ponto de acionamento dos bicos de ar

Fonte: Spectral Solutions Comércio e Serviços LTDA

de uma esteira, por onde deve passar a matéria prima a ser utilizada na produção, para mensurar a refletância dos objetos que passam por ela. As imagens geradas serão utilizadas por um modelo de classificação treinado para diferenciar entre esteira, matéria prima em conformidade e outros objetos indesejados (sejam eles contaminantes ou matéria prima alterada). Por brevidade, ao longo deste trabalho menções a estes objetos usarão o termo Objeto Rejeitado (OR).

A matéria prima é depositada em um silo de entrada, de onde é dispensada em uma esteira chamada de esteira secundária. Ao fim dela, os objetos caem em outra esteira localizada em uma altura menor e que opera em velocidade mais alta, sendo esta a esteira primária. Como a primária é a esteira sendo observada pela câmera, esta diferença de velocidades melhora o espalhamento dos objetos ao passarem pelo ponto de leitura, impedindo que alguns objetos obstruam outros na visão da câmera.

Ao fim da esteira principal, a matéria prima deve cair em um silo, de onde seguirá para as próxima etapas do processo industrial. Posicionados para atuar entre o fim da esteira e o silo, haverá bicos de ar que serão acionados durante a passagem de um OR de forma que sua trajetória seja desviada para uma pilha de descarte. Como há uma distância de aproximadamente 50 cm entre o fim da esteira e o ponto onde a

câmera faz a sua leitura, é necessário que o sistema monitore a velocidade da esteira para calcular o momento correto de acionamento dos bicos de ar.

O monitoramento da velocidade da esteira e a ativação dos bicos de ar serão responsabilidades de um microcontrolador. Já a interface com a câmera e a computação da saída do modelo de classificação exigem um alto poder de processamento para serem executadas sobre a esteira em constante movimento, portanto para essas atividades será utilizado um computador principal com *Central Processing Unit* (CPU) de 4 núcleos e *Graphical Processing Unit* (GPU). Este computador também deve apresentar um painel para monitoração e configuração do sistema, apresentar uma interface para calibração da câmera, guardar *logs* sobre a operação, registrar e guardar imagens das detecções de ORs, além de realizar a comunicação necessária com o microcontrolador e o sistema supervisório central da fábrica.

O envolvimento do autor deste trabalho no projeto ocorre principalmente na elaboração do modelo de classificação e na programação do computador principal e do microcontrolador, consequentemente o foco deste documento será na descrição destas etapas. Outras tarefas necessárias para a realização do sistema, como projeto e montagem da estrutura física, projeto e montagem da parte elétrica ou a escolha de componentes, foram executadas por outros membros da equipe e podem ser mencionadas em menos detalhes.

1.3 OBJETIVOS

O objetivo geral do projeto é a realização de um sistema automatizado de seleção de matéria prima, capaz de separar a matéria prima em conformidade com os critérios de qualidade do processo industrial de outros objetos não conformantes.

Objetivos mais específicos, necessários para atingir este objetivo geral, seriam:

- Elaborar um modelo de classificação capaz de diferenciar, com alta acurácia, entre esteira, matéria prima e outros objetos com base em suas refletâncias no espectro NIR.
- Programar o código fonte do computador principal e o do microcontrolador levando em conta o tempo de execução e possíveis atrasos, de forma a atender os requisitos de tempo real do sistema, ou seja, de forma que os bicos de ar sejam acionados no momento correto (quando um OR deixa a esteira)
- Verificar que o sistema é estável e livre de erros. Embora garantir a completa ausência de *bugs* em um *software* seja extremamente difícil, salvo casos simples, é importante que o sistema seja estável o bastante para operar continuamente por longos períodos de tempo sem atingir um estado de erro que seja irreversível

sem intervenção humana ou reinicialização do sistema, para evitar paralisações do processo produtivo.

ESTRUTURA DO DOCUMENTO

Fora este capitulo introdutório, este documento está divido em 4 outros capítulos, sendo o conteúdo deles organizado da seguinte maneira:

- Capitulo 2: São abordados conceitos teóricos de química, estatística e inteligencia artificial utilizados para o desenvolvimento dos modelos de classificação baseados em imagens hiper-espectrais.
- Capitulo 3: É descrito o processo de treinamento e validação dos modelos de classificação elaborados, incluindo a forma de como foram coletadas e pré-processadas as amostras dos espectros e as ferramentas utilizadas.
- Capitulo 4: Descreve a divisão de tarefas dos 3 componentes de *software* que operam no sistema, os motivos por serem divididos desta forma, a forma como tais tarefas são realizadas e o processo de decisão referente ao *design* e implementação deles.
- Capitulo 5: Breve discussão sobre os resultados observados até a data da entrega deste documento, além de possíveis novas aplicações, limitações e melhorias.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ESPECTROSCOPIA

No cotidiano a palavra luz é tipicamente usada para se referir às ondas eletromagnéticas percebidas pelo olho humano, o chamado espectro visível, que se restringe àquelas com comprimento entre 400 e 700 nanômetros. O espectro eletromagnético completo entretanto é muito mais abrangente e ondas com comprimentos além desta faixa também podem ser referidas como luz, como no caso do espectro NIR, que abrange as ondas com comprimento logo acima da faixa do espectro visível e é a região utilizada pelo sistema descrito neste trabalho.

Moléculas sob o efeito de ondas eletromagnéticas podem absorver sua energia, transitando do chamado estado fundamental para estados de excitação (ou de um estado de excitação para outro). Entretanto, tal interação não ocorre com toda a luz, mas apenas com aquela que carrega a quantidade correta de energia para promover a molécula de um nível energético discreto para outro (HARRIS; BERTOLUCCI, 1978). Consequentemente, pode-se concluir que a absorção da luz ocorre em frequências/comprimentos de onda específicos, considerando que a relação entre a energia de um fóton e o comprimento de onda é dado pela relação de Planck,

$$E_p = h \cdot f = \frac{h \cdot c}{\lambda}, \tag{1}$$

onde E_p é a energia do fóton, h é a constante de Planck, c é a velocidade da luz, f é a frequência e λ é o comprimento de onda, lembrando que a frequência é igual à velocidade da luz dividida pelo comprimento de onda,

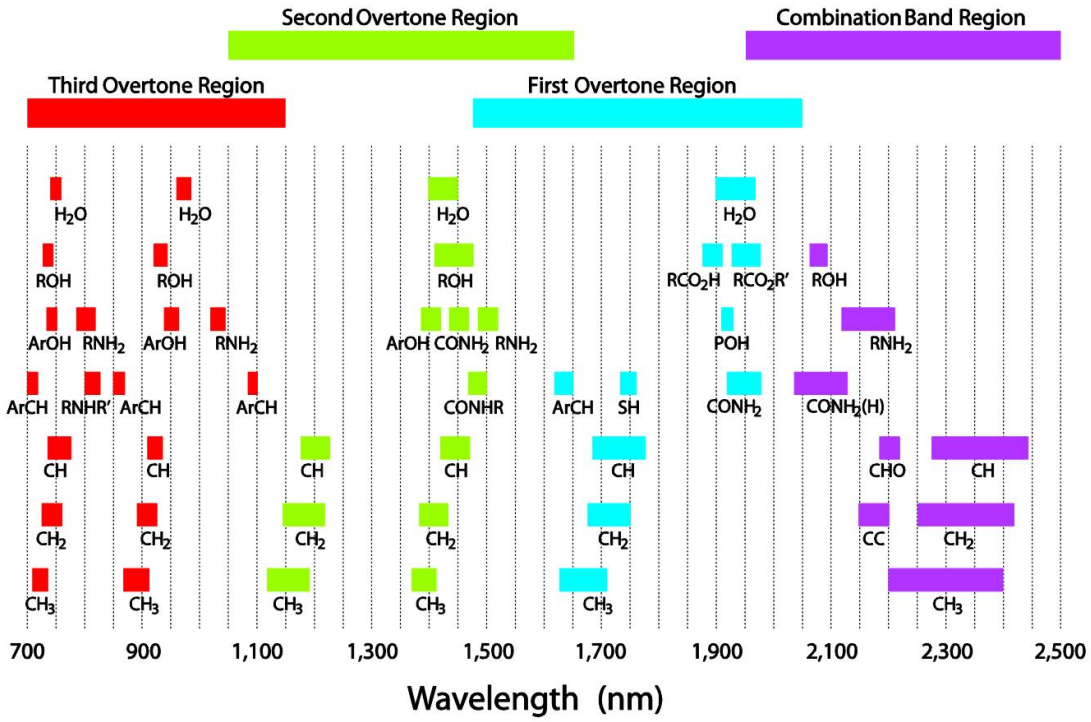
$$f = \frac{c}{\lambda}. \tag{2}$$

É por este mecanismo que a espectroscopia nos permite obter informações sobre a composição de uma amostra. Como a energia necessária para que ocorram as transições de um estado energético para outro depende dos átomos que compõem a molécula e os tipos de ligações entre eles, ao mensurar a absorção de luz da amostra em diversos comprimentos de onda obtém-se um perfil espectral que apresenta alta correlação com sua estrutura química.

A transição do estado fundamental para o primeiro estado de excitação tipicamente ocorre no espectro infravermelho médio, ou *Medium Infrared* (MIR). Entretanto, ondas com a energia necessária para tal também podem provocar transições do estado fundamental diretamente para o segundo estado de excitação (chamado primeiro sobretom), terceiro estado de excitação (segundo sobretom) e assim por diante. Diversas ligações atômicas importantes como C-H, O-H e N-H fazem com que moléculas

que as contenham apresentem sobretons no espectro NIR, conforme demonstrado na figura 2.

Figura 2 – Sobretons e bandas de combinação na região do espectro NIR



Fonte: Metrohm (2013)

Além dos sobretons, outro efeito que pode aparecer no espectro NIR é o das bandas de combinação. Ondas de frequências que carregam energia equivalente a uma combinação linear de valores de energia necessários para causar transições entre estados também podem ter sua energia absorvida.

2.2 ESTATÍSTICA E INTELIGÊNCIA ARTIFICIAL

2.2.1 Redução de Dimensionalidade

Nos dados obtidos por meio da NIRS a refletância medida em cada comprimento de onda representa uma dimensão do conjunto de dados, que pode ser representado como uma matriz $\mathbf{X}$ do tipo n x p, sendo n o número de amostras enquanto p é o número de dimensões. Como comprimentos de onda próximos tendem a ter valores de refletância similares, isso resulta em alta colinearidade entre as variáveis explicativas, de forma que técnicas de redução de dimensionalidade podem reduzir consideravelmente o volume de dados que precisa ser processado com pouca perda de informação.

2.2.1.1 PCA

A análise de componentes principais, ou *Principal Component Analysis* (PCA), se baseia em uma transformação linear que move a matriz $\mathbf{X}$ para uma nova representação, cujos vetores-base são chamados componentes principais. Tais componentes são definidas de forma que a primeira componente maximiza a variância dos vetores-linha de $\mathbf{X}$ quando projetados sobre ela, enquanto as componentes seguintes maximizam a variância atendendo a restrição adicional de que devem ser ortogonais às componentes anteriores.

Considerando a definição matemática da variância,

$$\sigma^2 = E[(\mathbf{x} - E[\mathbf{x}])^2] = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})^{\mathsf{T}}(\mathbf{x}_i - \bar{\mathbf{x}}), \tag{3}$$

se a matriz $\mathbf{X}$ for centralizada, ou seja, tiver valor médio igual a zero em todas as suas dimensões (ou vetores-coluna), a primeira componente principal pode ser definida como o vetor-coluna unitário $\mathbf{w}$ que maximiza a expressão matricial

$$||\mathbf{X}\mathbf{w}||^2 = \mathbf{w}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{X}\mathbf{w} = \frac{\mathbf{w}^{\mathsf{T}}\mathbf{X}^{\mathsf{T}}\mathbf{X}\mathbf{w}}{\mathbf{w}^{\mathsf{T}}\mathbf{w}}. \tag{4}$$

Esta expressão é conhecida como quociente de Rayleigh, sendo maximizada quando $\mathbf{w}$ é um autovetor associado ao maior autovalor da matriz $\mathbf{X}^{\mathsf{T}}\mathbf{X}$. De forma análoga, a segunda componente principal equivale ao autovetor do segundo maior autovalor, e assim por diante. Portanto a matriz de mudança de base $\mathbf{W}$ que leva $\mathbf{X}$ de seu espaço original para o espaço das componentes principais é a matriz cujos vetores-coluna são os autovetores de $\mathbf{X}^{\mathsf{T}}\mathbf{X}$.

Como na PCA supõe-se que a matriz $\mathbf{X}$ esteja centralizada, a partir da definição da matriz de covariância

$$\mathbf{C}_{XX} = \mathrm{Cov}(\mathbf{X}, \mathbf{X}) = E[(\mathbf{X} - E[\mathbf{X}])^{\mathsf{T}}(\mathbf{X} - E[\mathbf{X}])], \tag{5}$$

nota-se que a expressão se reduz a $E[\mathbf{X}^{\mathsf{T}}\mathbf{X}]$. Somando a isto o fato de que multiplicar a expressão em (4) por $\frac{1}{n}$ ou $\frac{1}{n-1}$ não altera o vetor que a maximiza, pode-se concluir que as componentes principais são os autovetores da matriz de covariância das variáveis explicativas.

Isto também implica que a matriz de covariância dos dados após a mudança de base será uma matriz diagonal, sendo os elementos na diagonal os autovalores da matriz de covariância original. Consequentemente, pode-se calcular a porcentagem da variância dos dados que é explicada pela componente principal *i* dividindo o *i*-ésimo maior autovalor pela soma de todos os autovalores, ou seja, pelo traço da matriz de covariância.

A utilidade dessa transformação como técnica de redução de dimensionalidade pode ser baseada na ideia que as últimas componentes, por capturarem pouca variância, possuem menos informação e podem ser descartadas. Tal suposição pode ser incorreta (JOLLIFFE, 1982), de forma que o uso indiscriminado de PCA como etapa de preprocessamento pode piorar a eficácia de modelos, mas em problemas baseados em NIRS tende a ser eficaz.

Por realizar a transformação baseando-se apenas na informação das variáveis de entrada, sem considerar as classes de cada amostra, a PCA pode ser considerada um método não supervisionado, o que pode ser uma vantagem ou desvantagem dependendo do contexto.

2.2.1.2 LDA

Ao contrário da PCA, a análise discriminante linear, ou *Linear Discriminant Analysis* (LDA), é um método específico para problemas de classificação que realiza redução de dimensionalidade de forma supervisionada, ou seja, utilizando a informação de a qual classe cada amostra pertence. Para isto o algoritmo busca encontrar uma nova representação que maximiza não a variância do conjunto de dados inteiro como na PCA, mas sim a variância inter classes (distâncias entre a média das classes) ao mesmo tempo que minimiza a variância intra classe.

No caso de classificação entre duas classes, isto é feito encontrando um vetor coluna $\mathbf{w}$ que maximize a expressão,

$$\frac{(\bar{x}_\alpha - \bar{x}_\beta)^2}{s_\alpha^2 + s_\beta^2}, \tag{6}$$

onde $\bar{x}_\alpha$ e $\bar{x}_\beta$ representam respectivamente o vetor médio das classes α e β projetados sobre $\mathbf{w}$ (ou seja $\mathbf{w}^\mathsf{T}\bar{\mathbf{x}}$), enquanto s_α^2 e s_β^2 representam a dispersão das classes quando projetadas sobre $\mathbf{w}$. Esta variável de dispersão é definida pela expressão

$$s^2 = \sum_{i=0}^{n} (x - \bar{x})^2, \tag{7}$$

onde pode se observar que s^2 é proporcional à variância.

A expressão em (6) também pode ser reescrita em forma matricial. Pela definição de s^2 pode-se ver que,

$$\begin{aligned} s_\alpha^2 &= \sum_{\mathbf{x}\in\mathbf{X}_\alpha} (\mathbf{w}^\mathsf{T}\mathbf{x} - \mathbf{w}^\mathsf{T}\bar{\mathbf{x}}_\alpha)^2 \\ &= \sum_{\mathbf{x}\in\mathbf{X}_\alpha} \mathbf{w}^\mathsf{T}(\mathbf{x} - \bar{\mathbf{x}}_\alpha)(\mathbf{x} - \bar{\mathbf{x}}_\alpha)^\mathsf{T}\mathbf{w} \\ &= \mathbf{w}^\mathsf{T}\mathbf{S}_\alpha\mathbf{w}, \end{aligned} \tag{8}$$

onde $\mathbf{X}_\alpha$ é o conjunto de amostras da classe α, n_α o número de amostras da classe α e $\mathbf{S}_\alpha$ a matriz de dispersão de $\mathbf{X}_\alpha$ (pode-se notar que da mesma forma como s^2 é proporcional à σ^2, a matriz $\mathbf{S}$ é proporcional à matriz $\mathbf{C}$). Assim, o denominador de (6) pode ser reescrito como $\mathbf{w}^\mathsf{T}\mathbf{S}_w\mathbf{w}$, onde $\mathbf{S}_w = \mathbf{S}_\alpha + \mathbf{S}_\beta$ é a matriz de espalhamento intra-classe. Já o numerador pode ser reescrito como

$$\begin{aligned}(\bar{x}_\alpha - \bar{x}_\beta)^2 &= (\mathbf{w}^\mathsf{T}\bar{\mathbf{x}}_\alpha^2 - \mathbf{w}^\mathsf{T}\bar{\mathbf{x}}_\beta)^2 \\ &= \mathbf{w}^\mathsf{T}(\bar{\mathbf{x}}_\alpha - \bar{\mathbf{x}}_\beta)^\mathsf{T}(\bar{\mathbf{x}}_\alpha - \bar{\mathbf{x}}_\beta)\mathbf{w} \\ &= \mathbf{w}^\mathsf{T}\mathbf{S}_b\mathbf{w},\end{aligned} \tag{9}$$

onde $\mathbf{S}_b$ é a matriz de dispersão inter-classe.

Combinando então (6), (8) e (9), a expressão a ser maximizada passa a ser

$$\frac{\mathbf{w}^\mathsf{T}\mathbf{S}_b\mathbf{w}}{\mathbf{w}^\mathsf{T}\mathbf{S}_w\mathbf{w}}, \tag{10}$$

que é a expressão do quociente de Rayleigh generalizado, sendo portanto maximizada quando $\mathbf{w}$ é o autovetor associado ao maior autovalor de $\mathbf{S}_w^{-1}\mathbf{S}_b$.

Para generalizar este método para problemas com K classes, pode-se buscar maximizar a distância da média de cada classe até a média da médias ($\bar{\bar{\mathbf{x}}}$), definida como

$$\bar{\bar{\mathbf{x}}} = \frac{1}{K}\sum_{i=1}^{K}\bar{\mathbf{x}}_i. \tag{11}$$

Assim, $\mathbf{S}_b$ pode ser redefinida como

$$\mathbf{S}_b = \sum_{i=1}^{K} n_i(\bar{\mathbf{x}}_i - \bar{\bar{\mathbf{x}}}). \tag{12}$$

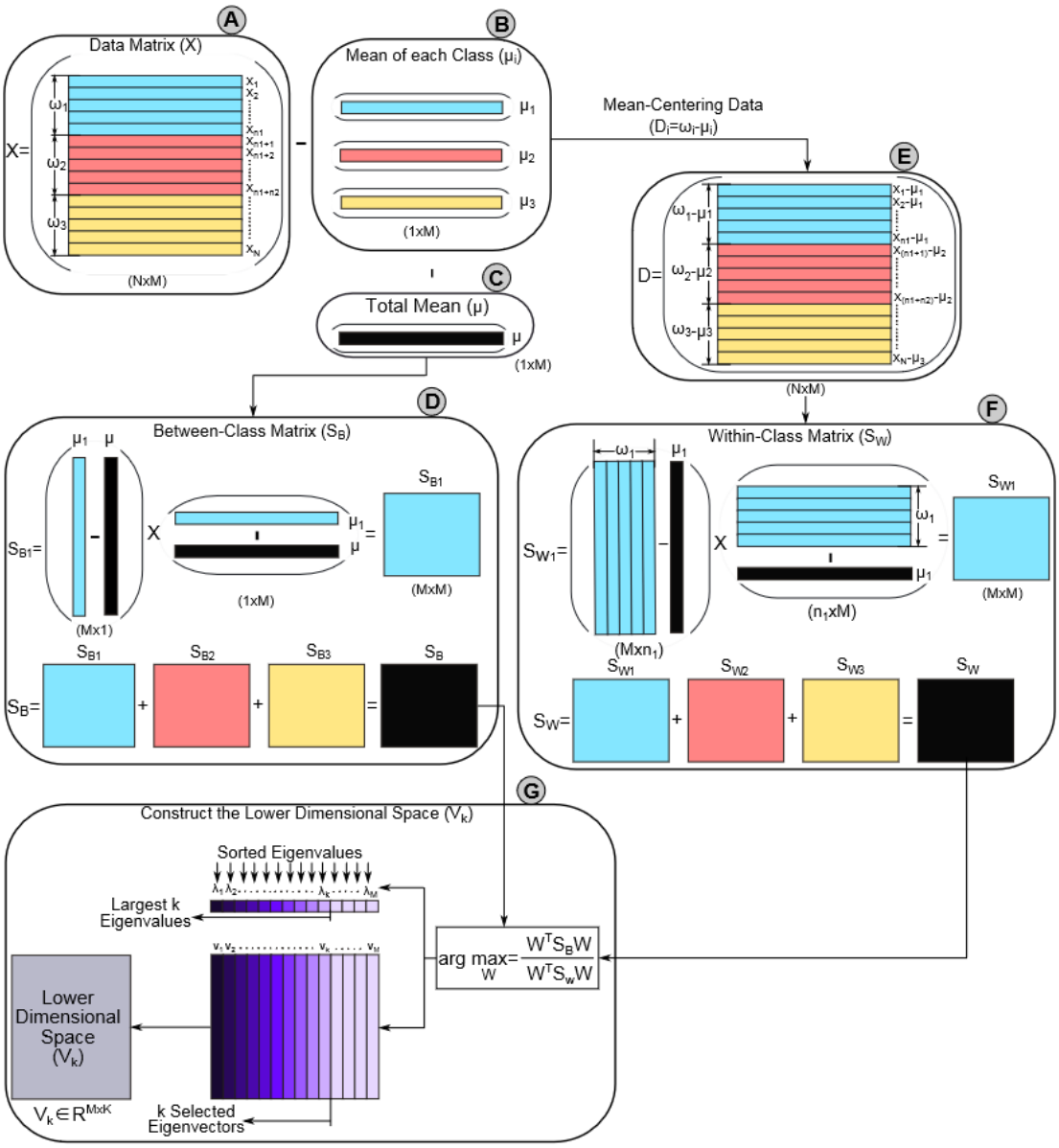
Se esta nova definição for aplicada ao caso com duas classes, o resultando será equivalente ao da definição anterior porém multiplicado por $\frac{n_\alpha n_\beta}{n}$. Como multiplicar (10) por uma constante não altera qual $\mathbf{w}$ a maximiza, ambas as definições apresentam o mesmo resultado. A figura 3 apresenta um diagrama onde pode ser visualizado este algoritmo.

Assim como a PCA, a LDA realiza uma mudança de base em $\mathbf{X}$, porém pela forma como é construída, o posto da matriz $\mathbf{S}_b$ não pode ser maior que $K-1$, de forma que a nova representação só pode ter no máximo $K-1$ dimensões (DUDA; HART; STORK, 2001).

2.2.1.3 PLS

Outro método supervisionado de redução de dimensionalidade, porém com abordagem diferente da LDA, é o método dos mínimos quadrados parciais, ou *Partial*

Figura 3 – Diagrama representando os passos do algoritmo LDA



Fonte: Tharwat *et al.* (2017)

Least Squares (PLS). Assim como a PCA, este método busca encontrar uma transformação linear que realiza uma mudança de base em $\mathbf{X}$, porém adicionalmente também busca encontrar uma outra transformação linear para realizar uma mudança de base na matriz $\mathbf{Y}$, que representa as variáveis resposta do conjunto de dados.

Os vetores-base dessas novas representações são chamados de variáveis latentes. Definindo $\mathbf{w_{X1}}$ e $\mathbf{w_{Y1}}$ como a primeira variável latente de $\mathbf{X}$ e $\mathbf{Y}$ respectivamente, a PLS busca encontrar variáveis latentes que maximizem a covariância da projeção de $\mathbf{X}$ e $\mathbf{Y}$, ou seja Cov($\mathbf{X w_{X1}}, \mathbf{Y w_{Y1}}$). Assim como em algoritmos anteriores, essa condição é maximizada quando as variáveis latentes correspondem aos autoveto-

res associados ao maior autovalor de um matriz, no caso $\mathbf{X}^T\mathbf{Y}\mathbf{Y}^T\mathbf{X}$ para $\mathbf{w}_{\mathbf{X}1}$ e de $\mathbf{Y}^T\mathbf{X}\mathbf{X}^T\mathbf{Y}$ para $\mathbf{w}_{\mathbf{Y}1}$ (WEGELIN, 2000).

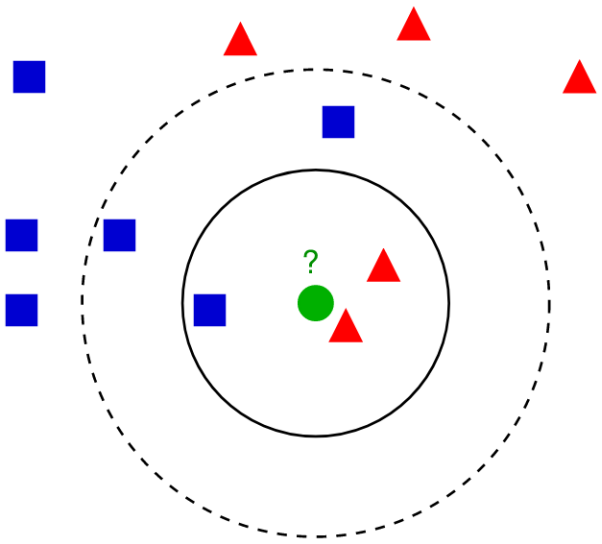
Embora tipicamente utilizado em problemas de regressão, o PLS também pode ser aplicado a problemas de classificação utilizando variáveis *dummy* na matriz $\mathbf{Y}$, ou seja, tornando $\mathbf{Y}$ uma matriz *n* x *K*, onde cada amostra tem valor 1 na coluna correspondente à sua classe e 0 nas outras. Ao fazer isto o posto da matriz $\mathbf{Y}$ passa a ser *K*, o que implica que só poderão ser encontradas *K* variáveis latentes, ao contrário de um problema de regressão, onde o PLS permite encontrar até *p* variáveis latentes assim como acontece na PCA.

2.2.2 Algoritmos de Classificação

2.2.2.1 KNN

Apesar de conceitualmente muito simples, o algoritmo dos k vizinhos mais próximos, ou *K Nearest Neighbors* (KNN), é capaz de capturar relações não lineares relativamente complexas. A partir de um conjunto de amostras $\mathbf{X}$, o algoritmo decide a qual classe uma nova amostra pertence encontrando as *k* amostras mais próximas (suas vizinhas) no conjunto de treinamento e atribuindo a esta nova amostra a classe mais comum entre suas vizinhas, conforme exemplificado na figura 4. O critério de proximidade pode ser escolhido de acordo com a natureza do problema, podendo ser utilizadas métricas como a distância Euclidiana, distância de Mahalanobis ou outras métricas mais complexas.

Figura 4 – Exemplo de classificação com KNN utilizando distância Euclidiana. O algoritmo atribuirá o ponto à classe vermelha no caso $k = 3$ e ao ponto azul no caso $k = 5$



Fonte: (AJANKI, 2022)

Quando $k = 1$, todo novo dado recebe a mesma classificação do ponto mais próximo no conjunto de treinamento, enquanto para $k = n$ o modelo resultante simplesmente escolheria sempre a classe mais comum do conjunto de treinamento para qualquer dado novo. Portanto em termos práticos o parâmetro k atua como regularizador, onde valores baixos aumentam a variância do modelo e o tornam propenso a sobre-ajuste (*overfitting*), enquanto valores altos de k aumentam o viés do modelo tornando-o propenso a sub-ajuste (*underfitting*).

Em termos de performance de execução o KNN apresenta uma característica peculiar. A princípio não há tempo de treinamento, a própria existência do conjunto de treinamento é tudo que o modelo precisa pra classificar novos dados, enquanto o tempo de execução do modelo para classificar novos dados cresce de acordo com o tamanho do conjunto de treinamento. Implementado na sua forma mais simples, para classificar um novo ponto o algoritmo precisaria calcular sua distância em relação a todos os pontos do conjunto de treinamento para encontrar os k vizinhos mais próximos. Para contornar esta desvantagem há implementações do algoritmo que criam subdivisões no espaço do problema ou constroem uma estrutura de árvore com o conjunto de treinamento, tornando o tempo de predição do modelo muito menor quando a quantidade de dados é grande.

Uma variação comum deste algoritmo é a utilização de diferentes pesos para cada vizinho, o que permite fazer com que vizinhos mais próximos tenham influência maior na decisão final do classificador que vizinhos mais distantes.

2.2.2.2 SVM

A máquina de vetor de suporte, ou *Support Vector Machine* (SVM), busca encontrar um hiperplano no espaço p dimensional de $\mathbf{X}$, definido como

$$\mathbf{w} \cdot \mathbf{x} - b = 0, \tag{13}$$

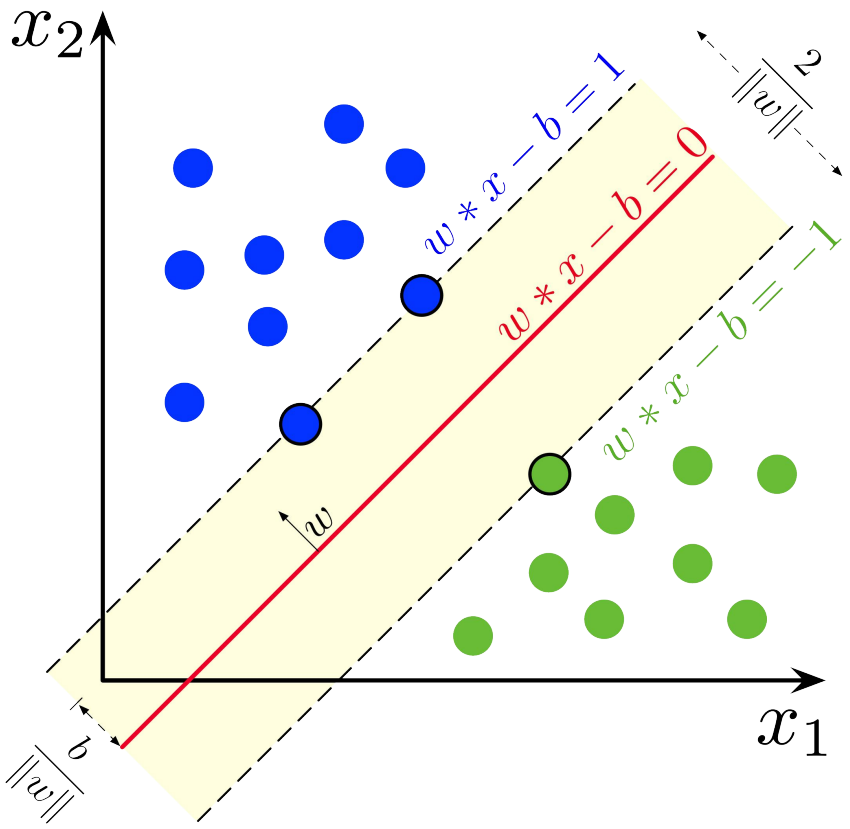
que separe duas classes baseando-se no critério de maximização da margem, ou distância, entre o hiperplano e as amostras das classes mais próximas dele.

Supondo que $\mathbf{x}_\alpha$ é a amostra da classe α mais próxima do hiperplano e que $\mathbf{x}_\beta$ seja o mesmo para a classe β. Pode-se definir que há dois outros hiperplanos paralelos ao separador, um hiperplano $\mathbf{w} \cdot \mathbf{x} - b = 1$ que passa por $\mathbf{x}_\alpha$ e outro hiperplano $\mathbf{w} \cdot \mathbf{x} - b = -1$ que um passa por $\mathbf{x}_\beta$, conforme exemplificado na figura 5 (onde há dois candidatos a $\mathbf{x}_\alpha$). Desta forma, o conceito de margem pode ser definido como a distância entre estes dois hiperplanos.

Como $\mathbf{w}$ é normal aos hiperplanos, a margem pode ser encontrada a partir da projeção do vetor $\mathbf{x}_\alpha - \mathbf{x}_\beta$ sobre um vetor unitário na direção de $\mathbf{w}$

$$(\mathbf{x}_\alpha - \mathbf{x}_\beta) \cdot \frac{\mathbf{w}}{||\mathbf{w}||} = \frac{\mathbf{x}_\alpha \cdot \mathbf{w} - \mathbf{x}_\beta \cdot \mathbf{w}}{||\mathbf{w}||} = \frac{(1 + b) - (-1 + b)}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}, \tag{14}$$

Figura 5 – Ilustração do hiperplano e margem de uma SVM



Fonte: (LARHMAM, 2022)

o que significa que o objetivo de encontrar um hiperplano que maximize a margem equivale a minimizar $||\mathbf{w}||$, atendendo a restrição que $\mathbf{w} \cdot \mathbf{x} - b \geq 1$ para toda amostra da classe α e $\mathbf{w} \cdot \mathbf{x} - b \leq -1$ para toda amostra da classe β. Definindo $\mathbf{Y}$ como um vetor-coluna onde as linhas y_i têm valor 1 para as amostras de α e -1 para as amostras de β, a restrição pode ser reescrita como

$$y_i(\mathbf{x}_i \cdot \mathbf{w} - b) \geq 1. \tag{15}$$

Esta definição da margem da SVM, conhecida como margem rígida, possui o grande defeito de assumir que as classes são linearmente separáveis. Para lidar com casos onde tal condição não se aplica, como frequentemente é o caso em situações práticas, utiliza-se a chamada margem suave. Introduzindo uma variável auxiliar ξ para representar o quanto uma amostra passou da margem

$$\xi_i = \max(0,\ 1 - y_i(\mathbf{x}_i \cdot \mathbf{w} - b)), \tag{16}$$

pode-se introduzir uma função de perda conhecida como *hinge loss*

$$||\mathbf{w}|| + k_r \frac{1}{n} \sum_{i=1}^{n} \xi_i, \tag{17}$$

onde k_r é uma constante de regularização, passando a ser esta função a expressão a ser minimizada. Note que quando a restrição (15) é atendida, então ξ é zero.

Uma propriedade importante das SVMs é que este problema de otimização, tanto no caso da margem rígida quanto da margem suave, é convexo e portanto converge para uma solução ótima global (SUPPORT-VECTOR. . . , 1995).

Outra propriedade interessante da SVM é que sua fronteira de decisão é definida estritamente por um subconjunto de $\mathbf{X}$, os vetores de suporte. No caso da margem rígida essas amostras são as que definem a margem do hiperplano, enquanto na margem suave somam-se a estas aquelas que estão do lado "errado" da margem, ou seja, que não atendem (15) (e consequentemente têm $\xi_i \neq 0$), sendo que o parâmetro k_r controla o quanto de influência as amostras que definem a margem têm em relação às amostras que não atendem (15).

Embora da forma descrita aqui o algoritmo trace uma fronteira de decisão linear entre duas classes, ele também pode ser expandido para casos multi-classe, tipicamente combinando múltiplos modelos de classificação binária usando estratégias *one-versus-one* ou *one-versus-all*, assim como para casos não lineares usando métodos de *kernel*.

Diversos modelos lineares, seja de regressão ou de classificação, podem ser utilizados para modelar relações não lineares a partir de uma transformação não linear das variáveis do problema, mas o aumento resultante da dimensionalidade do problema e possível explosão combinatória (dependendo do time de transformações lineares escolhidas) torna tal técnica inviável em diversos problemas. O método de *kernel* aplicado a SVMs ameniza este problema por permitir o treinamento e aplicação do modelo apenas através do cálculo de um produto interno no espaço transformado.

Utilizando multiplicadores de Lagrange, a expressão a ser minimizada (17) pode ser expressa em função apenas do produto interno das amostras (HASTIE; TIBSHIRANI; FRIEDMAN, 2009)

$$\sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j y_i y_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \ , \tag{18}$$

onde c são os multiplicadores de Lagrange, e $\phi(\mathbf{x})$ representa uma transformação em $\mathbf{x}$. Consequentemente, pode ser demonstrado que a função de decisão do modelo pode ser escrita como

$$\sum_{i=1}^{n} c_i y_i k(x, x_i) - b \tag{19}$$

onde $k(\mathbf{x}, \mathbf{x}_i) = \langle \phi(\mathbf{x}), \phi(\mathbf{x}_i) \rangle$ é a chamada função de *kernel*. Como em uma SVM a fronteira de decisão depende estritamente dos vetores de suporte, c_i será não nulo apenas para estas amostras, portanto o modelo pode realizar predições em um espaço não linear apenas computando a função de *kernel* entre uma nova entrada e os vetores de suporte encontrados durante o treinamento.

Um exemplo de função de *kernel* comumente utilizada em SVMs é a *Radial Basis Function* (RBF), que pode ser descrita pela expressão

$$k(\mathbf{x}, \mathbf{x}_i) = e^{-\gamma \cdot ||\mathbf{x} - \mathbf{x}_i||^2} \tag{20}$$

onde γ é um hiper-parâmetro a ser escolhido antes de treinar o modelo. Esta função pode ser interpretada como uma métrica de similaridade, considerando que seu valor é 1 quando $x = x_i$ mas tende a zero conforme $||\mathbf{x} - \mathbf{x}_i||^2$ aumenta.

2.2.2.3 ANN

Inspiradas no sistema nervoso de animais, as redes neurais artificiais, ou *Artificial Neural Networks* (ANN)s, procuram modelar relações complexas a partir da combinação de várias funções mais simples, chamadas de neurônios. Diversas variações de redes neurais existem, mas na sua forma mais básica, do tipo *feedforward*, cada neurônio pode ser descrito como uma combinação linear de valores de entrada, somados de um valor constante (chamado *bias*) e aplicados a uma função de ativação,

$$y = \mathrm{a}(\mathbf{w}^{\mathsf{T}}\mathbf{x} + b), \tag{21}$$

onde y é o valor de saída, $\mathbf{x}$ é um vetor de entrada, $\mathbf{w}$ um vetor de pesos, b é *bias* e $a(x)$ é a chamada função de ativação. Diversas funções podem ser usadas como função de ativação, mas uma das mais comuns é a função *Rectified Linear Unit* (ReLU),

$$\mathrm{ReLU}(x) = \max(0, x) = \begin{cases} x, x \geq 0 \\ 0, x < 0 \end{cases} . \tag{22}$$

Os neurônios de uma rede *feedforward* são organizados em camadas, onde os valores de saída dos neurônios de uma camada são os valores de entrada dos neurônios da camada seguinte, os valores de entrada da ANN são os valores de entrada da primeira camada (camada de entrada) e os valores de saída da ANN são os valores de saída da última camada (camada de saída).

Em problemas de classificação tipicamente são colocados tantos neurônios na camada de saída quanto há classes no problema, sendo aplicada nos resultados dessa camada a função *softmax*,

$$\mathrm{softmax}(\mathbf{y})_i = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}}. \tag{23}$$

O efeito dessa função é padronizar o resultado de forma que os valores de saída estejam todos entre 0 e 1 e que a soma deles seja 1, tornando-os análogos a probabilidades. A resposta correta atribuída a cada ponto do conjunto de treinamento, então, é aquela onde o neurônio da camada de saída correspondente à classe correta tem como saída o valor 1, enquanto os outros neurônios dessa camada têm valor 0. Importante entretanto notar que estes valores não são de fato probabilidades, apenas ajudam a indicar a indecisão do modelo.

Assim como em outros métodos, o treinamento de uma ANN é um problema de otimização matemática que busca minimizar uma função de erro, que em problemas de classificação costuma ser a entropia cruzada,

$$\mathrm{E}(\mathbf{y}) = -\sum_{i=1}^{K} t_i \cdot \ln(y_i), \tag{24}$$

sendo t_i o valor verdadeiro (0 ou 1) para a classe i deste ponto do conjunto de treinamento e $\mathbf{y}$ é o vetor de saída após passar pela função *softmax*, já que os valores precisam estar entre 0 e 1 para a entropia cruzada resultar em um valor positivo. Esta minimização é feita de forma iterativa, alterando os valores dos pesos e dos *biases* dos neurônios por meio do algoritmo do gradiente descendente,

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \eta \nabla_{\mathbf{w}} \mathrm{E}, \tag{25}$$

$$b_{i+1} = b_i - \eta \frac{\partial \mathrm{E}}{\partial b}, \tag{26}$$

onde η é a chamada taxa de aprendizado e $\nabla_{\mathbf{w}}\mathrm{E}$ o gradiente da função de erro em relação aos pesos.

Embora redes neurais possam possuir milhares de neurônios, os gradientes podem ser computados de forma eficiente utilizando o algoritmo de retropropagação. Neste algoritmo, em uma rede com L camadas, é calculada a derivada da função de erro em relação às ativações dos neurônios das camadas de saída como valor intermediário δ^L. Aproveitando-se da propriedade da derivada conhecida como regra da cadeia,

$$\frac{dy}{dx} = \frac{dy}{dz}\frac{dz}{dx}, \tag{27}$$

o valor obtido para δ^L é utilizado como base para calcular os gradientes da camada L e as derivadas das ativações da camada $L-1$, obtendo outro valor intermediário δ^{L-1}, repetindo este processo até a primeira camada.

As equações (21) e (25) podem ser reescritas na forma matricial para representar as mesmas operações quando realizadas sobre todos os neurônios de uma camada. Isto é importante pois embora o processo de treino de uma ANN possa ser

computacionalmente custoso mesmo com a retro-propagação, existem diversas bibliotecas de código aberto disponíveis para computar seus resultados utilizando GPUs, que são mais eficientes que CPUs na computação de operações matriciais. Desta forma mesmo redes complexas com grande número de neurônios são viáveis de se treinar e capazes de realizar classificações de forma rápida.

A capacidade das ANNs de aprender relações complexas viabiliza a determinação de modelos bastante eficazes, porém também exige cuidados por torná-las propensas a problemas de *overfitting*. Algumas técnicas de regularização comumente utilizadas para reduzir este problema são *dropout* (que a cada iteração do treinamento anula a ativação de parte dos neurônios da rede), penalização de pesos L1 ou L2 (que adicionam à função de custo um termo em função dos pesos da rede multiplicado por um parâmetro de regularização) e restrição de pesos.

Uma variação de ANN que se demonstra muito efetiva em aplicações de visão computacional é *Convolutional Neural Network* (CNN), que além das camadas mencionadas anteriormente também utiliza camadas convolucionais. Nestas camadas, ao invés de aplicar uma função de ativação sobre uma combinação linear dos resultados da camada anterior, é aplicada uma convolução sobre elas

$$o_i^L = (\mathbf{o}^{L-1} * \mathbf{k}^L)_i = \sum_{j=1}^{k_s} o_{n-j+c_k}^{L-1} \cdot k_j \ , \tag{28}$$

onde $\mathbf{o}^L$ representa o vetor de saída da camada L, $\mathbf{k}$ o vetor do *kernel* de convolução, k_s seu tamanho e c_k o índice do elemento no centro do *kernel* (assumindo k_s ímpar). Os mesmos conceitos de utilização da regra da cadeia para atualizar os pesos de uma rede *feedforward* via retropropagação se aplicam também aos valores do *kernel* de uma camada convolucional. A formulação de (28) representa uma convolução com *kernel* unidimensional, porém o mesmo conceito pode ser generalizado para um número qualquer de dimensões.

As camadas convolucionais conseguem capturar uma relação entre as variáveis preditivas que são ignoradas pelas combinações lineares das camadas de uma rede *feedforward* comum. Por exemplo, um *kernel* bidimensional em um problema de classificação de imagem, por ser aplicado sobre pixeis próximos entre si, é capaz de capturar o contexto espacial de seus valores e assim identificar a presença de formas geométricas em área específicas de imagem.

3 DESENVOLVIMENTO DO MODELO DE CLASSIFICAÇÃO

3.1 FERRAMENTAS UTILIZADAS

3.1.1 Câmera Hiper espectral

A câmera hiper espectral utilizada no sistema é a Specim FX17, que mede a refletância em 224 bandas do espectro NIR, mais especificamente entre 900 e 1700 nanômetros, coletando até 640 *frames* de 640 pixels por segundo. Uma observação a ser feita é que embora o termo "*frames* por segundo" comumente seja utilizado para se referir à taxa de quadros de vídeos, no contexto da câmera hiper espectral, *frame* é o termo utilizado para se referir a uma linha da imagem, pois a câmera opera realizando a leitura linha a linha, concatenado-as para formar a imagem. Em outras palavras, neste contexto a captura de 640 *frames* de 640 pixels equivale à captura de uma imagem com resolução de 640x640 pixels.

Para o processamento dos dados hiper espectrais em tempo real, entretanto, não é necessário utilizar a imagem completa. A fabricante da câmera disponibiliza um *Software Development Toolkit* (SDK), incluindo uma *Application Programming Interface* (API) em C, que possibilita o processamento das leituras de cada *frame* assim que adquirido. Ou seja, supondo por exemplo que a câmera seja configurada para operar a 500 *frames* por segundo, uma linha de 640 pixels será gerada a cada $\frac{1}{500}$ segundos, ou 2 milissegundos. Ao invés de utilizar imagens de 640x500 pixels geradas a cada segundo como entrada do modelo de classificação, pode-se utilizar os *frames* de 640x1 pixels gerados a cada 2 milissegundos.

3.1.2 Software

As análises exploratórias sobre os dados e o treinamento dos modelos de classificação foram primeiramente realizados utilizando a linguagem de programação Python, amplamente utilizada no ramo de ciência de dados, devido à disponibilidade de bibliotecas de código aberto bem consolidadas e com uma diversidade de recursos voltados para aplicações de estatística, calculo numérico, inteligência artificial e visão computacional, como NumPy, Pandas, Scipy, OpenCV, Scikit-learn e PyTorch. Outro fator positivo é a ferramenta Jupyter Notebook, que permite a execução de código em partes de forma interativa e é extremamente conveniente para a análise de dados e para prototipagem e validação de modelos.

Embora seja uma linguagem de alto nível com foco na legibilidade e facilidade de uso, sendo relativamente lenta quando comparada a diversas outras linguagens, o Python possui interface com a linguagem de baixo nível C permitindo que partes críticas do código em termos de performance sejam implementadas e otimizadas em sub-rotinas desta linguagem, para então serem referenciadas pelo código de alto nível.

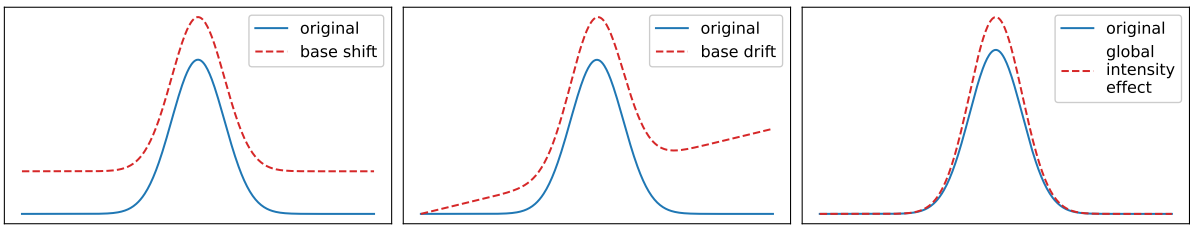
Tal integração é recurso amplamente utilizado de forma direta ou indireta (incluindo outras bibliotecas que fazem isso de forma direta como dependência) por todas as bibliotecas citadas.

Mesmo com a utilização de sub-rotinas em C, o uso do Python ainda implica em um *overhead* significativo que pode tornar sua utilização inviável para aplicações que exigem execução rápida dos modelos. Tal questão é abordada na seção 4.2.

3.2 PRÉ-PROCESSAMENTO

Devido a efeitos de dispersão luminosa na leitura de amostras sólidas, é comum que dados espectrais apresentem erros sistemáticos que devem ser corrigidos por meio de uma etapa de pré-processamento dos dados (PASQUINI, 2003). Os efeitos dos três tipos de erro sistemático comumente encontrados, *base shift*, *base drift* e *global intensity effect*, são apresentados na figura 6.

Figura 6 – Efeito dos erros sistemáticos *base shift*, *base drift* e *global intensity effect*

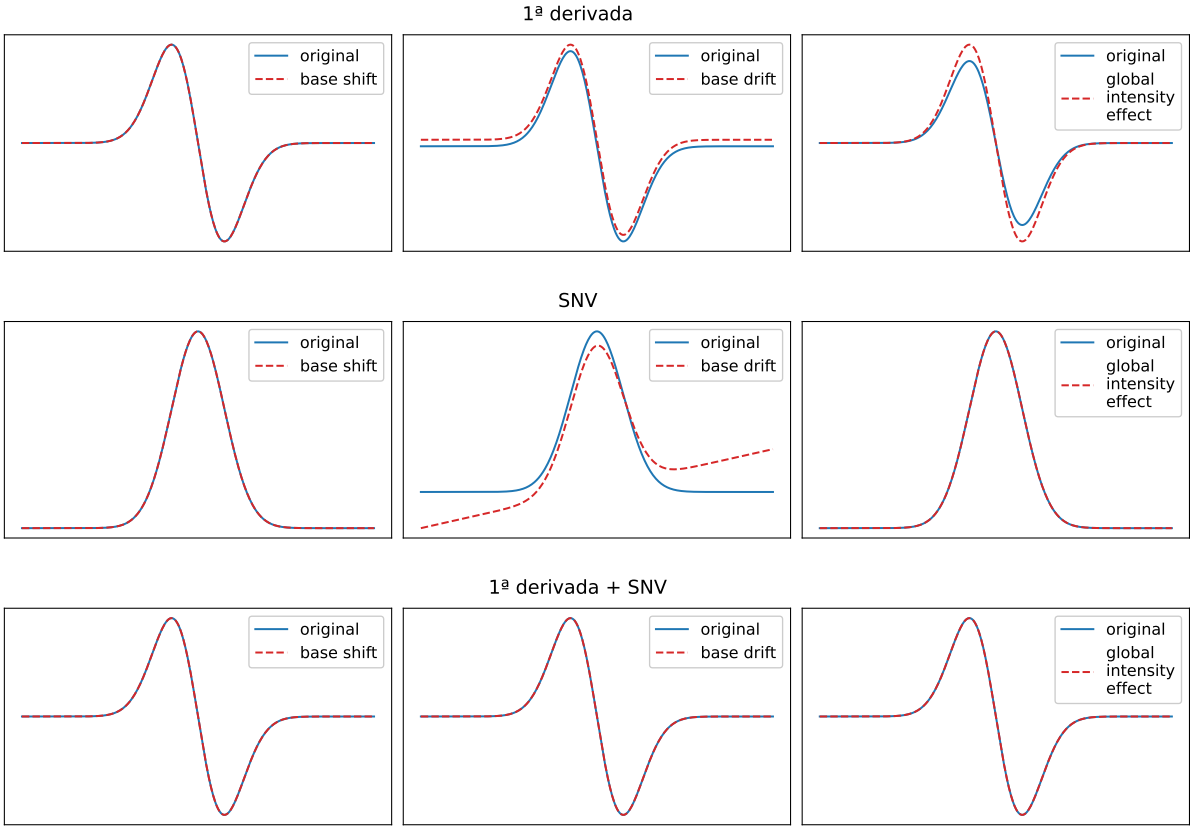


Fonte: Autoria própria

A correção destes erros é feita por meio de uma diferenciação numérica do espectro, realizada através da convolução das linhas de $\mathbf{X}$ por um Filtro de Savitzky-Golay (FSG), combinada com uma transformação *Standard Normal Variate* (SNV), que equivale a padronizar as linhas de $\mathbf{X}$ de forma que $\bar{\mathrm{x}} = 0$ e $\sigma = 1$. Note que em estatística o termo “padronizar os dados” geralmente se refere à padronização das variáveis, portanto das colunas de $\mathbf{X}$, diferentemente da SNV, que aplica esta padronização sobre as amostras, portanto sobre as linhas. A figura 7 mostra como a primeira derivada corrige *base shift* e converte *base drift* em *base shift*, enquanto o SNV corrige *base shift* e *global intensity effect*, de forma que a combinação de ambos corrige esses 3 erros sistemáticos.

Já a figura 8 demonstra o efeito do pré processamento com primeira derivada e SNV quando aplicados a espectros de uma fruta mensurados com um espectrômetro NIR portátil. O gráfico exibe 10 espectros, e apesar de todos eles terem sido obtidos do mesmo fruto, nota-se que há grande variação entre eles, devido aos erros apontados anteriormente. Após o pré-processamento tais variações são corrigidas, tornando os

Figura 7 – Efeito de um FSG de primeira derivada e transformação SNV sobre erros sistemáticos



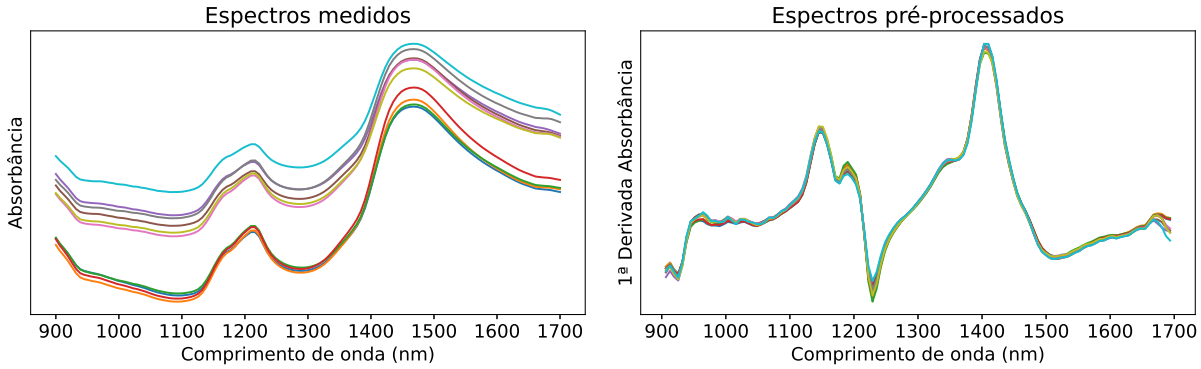
Fonte: Autoria própria

espectros bastante similares, com apenas leves diferenças que são normais devido ao fato de que o fruto não é necessariamente homogêneo, podendo haver variações no teor de açúcar, umidade, etc, nos diferentes pontos do alimento avaliados pelo espectrômetro. Também pode ser observado, comparando a figura 8 com a figura 2, que após o pré-processamento fica nítido como os picos de absorbância coincidem com as regiões dos sobretons de moléculas com ligações C-H e O-H, que é o que se espera de uma fruta devido à predominante presença de água e carboidratos na sua composição.

3.3 TREINAMENTO DOS MODELOS

Para o treinamento dos modelos foram coletadas amostras de 8 categorias de matérias-primas utilizadas no processo produtivo e de 10 categorias de contaminantes, sendo estas amostras então usadas para obter imagens hiper espectrais cujos espectros formarão o conjunto de dados para treinamento dos modelos. Para manter a maior fidelidade possível à situação na qual o modelo terá que operar na prática, tais

Figura 8 – Exemplo do efeito do pré-processamento sobre 10 espectros obtidos de um mesmo fruto



Fonte: Autoria própria

imagens foram obtidas utilizando a mesma câmera, iluminação e esteira que viriam a ser utilizadas no sistema concluído.

Cada imagem foi gerada com apenas um tipo de matéria-prima ou contaminante, sendo então selecionados de forma manual quais são os píxeis que representam matéria-prima, contaminantes e esteira. As imagens de dimensões 640 x *frames* x 224 foram então redimensionadas em matrizes com número de linhas igual ao número de píxeis total da imagem e 224 colunas, sendo que a combinação destas matrizes forma o conjunto de treinamento do modelo, representado por uma matriz cujo número de linhas é o total de píxeis de todas as imagens somadas e o número de colunas é 224.

A organização do conjunto de dados desta forma acaba resultando na perda da informação espacial da imagem, sendo impossível, portanto, que o modelo tire conclusões sobre o objeto na esteira com base na sua forma ou tamanho. Embora em aplicações de visão computacional típicas, utilizando imagens com os 3 canais de cor *Red Green Blue* (RGB), o contexto espacial de cada píxel possa ser crucial para que conclusões sejam tomadas sobre a imagem, no caso de uma imagem hiper espectral a riqueza de informações que pode ser obtida a partir das 224 bandas coletadas já é suficiente para retirar a informação desejada por esse sistema (se há algo que deve ser retirado da esteira ou não). O modelo, portanto, fará a classificação píxel a píxel, com base estritamente no espectro NIR, podendo ser considerado um modelo de segmentação semântica.

Após sua construção, a matriz do conjunto de treinamento apresenta um grande desequilíbrio na quantidade de píxeis entre as diferentes classes, especialmente em relação à esteira (por ser a única classe que aparece em todas as imagens), mas também entre alguns contaminantes e matéria-prima, devido à disponibilidade menor de objetos daquela classe durante a captura das imagens. Para evitar que esse

desbalanceamento induza um viés no modelo de classificação ou na redução de dimensionalidade é realizada uma amostragem estratificada para que cada classe tenha o mesmo número de píxeis no conjunto de treinamento. A classe menos representada é um dos contaminantes com pouco mais de 2800 píxeis, portanto, como o objetivo do projeto é separar esteira, alimentos e contaminantes, foram selecionados 2800 píxeis de cada um dos 10 contaminantes, somados de 28 000 de esteira e 28 000 de alimentos igualmente distribuídos entre os 8 tipos coletados. Consequentemente, após este processo a matriz $\mathbf{X}$ a ser usada no treinamento do modelo tem ao todo 84 000 espectros.

Embora existam outras estratégias para lidar com esse tipo de desbalanceamento sem remover dados, no caso deste problema uma simples amostragem foi considerada satisfatória. Considerando que cada píxel representa uma amostra do conjunto de dados e que não se observa divergência muito grande entre os diferentes espectros obtidos de um mesmo material, o número de amostras total do conjunto é bastante elevado e com alta redundância, de forma que se espera que o modelo de classificação consiga generalizar uma fronteira de decisão para a classificação a partir de um subconjunto destes dados.

Aplicando então sobre $\mathbf{X}$ os métodos de pré-processamento descritos na seção 3.2, em seguida foi utilizada a técnica de validação cruzada *k-fold* com $k = 5$, para treinar e avaliar o desempenho dos diferentes modelos a fim de determinar valores adequados para os diferentes hiper-parâmetros de cada um. Em um primeiro momento este processo foi feito sem utilizar técnicas de redução de dimensionalidade, pois deseja-se utilizar os modelos treinados sem elas para se ter uma base de comparação para depois avaliar o impacto que estas técnicas apresentam sobre a acurácia do modelo.

Além dos hiper-parâmetros dos modelos também se avaliou o desempenho para diferentes números de pontos na derivação por FSG usada no pré-processamento. Como este filtro tem a característica de combinar derivação com *smoothing*, um filtro com número de pontos muito baixo pode resultar em um sinal muito ruidoso, enquanto com um número de pontos muito alto pode haver atenuação excessiva e perda de poder preditivo dos modelos.

Os hiper-parâmetros escolhidos e a acurácia total obtida para os modelos KNN, SVM linear, SVM não linear usando *kernel* do tipo RBF, ANN simples *feedforward* e CNN com convoluções unidimensionais, constam na tabela 1, enquanto para o FSG foi escolhida uma janela de 11 pontos para a derivação.

Nota-se, portanto que os modelos SVM com *kernel* RBF, ANN e CNN apresentam alta acurácia, especialmente considerando que a porcentagem apresentada se refere à classificação píxel a píxel, pois mesmo que o classificador não identifique todos os píxeis do contaminante, a identificação de apenas uma parte deles pode ser

Tabela 1 – Acurácia dos modelos sem redução de dimensionalidade

Modelo	**Parâmetros**	**Acurácia média na validação cruzada**
KNN	$k = 7$	0.900
SVM - linear	$k_r = 1$	0.889
SVM - RBF	$k_r = 10$, $\gamma = 0.01$	0.967
ANN	2 camadas ReLU (200, 100) , $L2 = 0.1$	0.967
CNN	3 camadas conv. e *max pooling*, 1 camada ReLU , $L2 = 0.01$	0.982

Fonte: Autoria própria

suficiente para que o sistema acione os bicos de ar no momento correto para a ejeção.

Dividindo $\mathbf{X}$ pela metade, treinando um modelo SVM - RBF com metade dos dados (utilizando os mesmos hiper-parâmetros demonstrados na tabela 1) e usando a outra como conjunto de validação, obtém-se a matriz de confusão da figura 9, demonstrando que este modelo classifica corretamente quase todos os píxeis de contaminantes. Tal padrão se observa também na matriz de confusão dos outros modelos, onde os contaminantes são a classe com maior acurácia e o erro mais frequente é a atribuição das classes alimento ou contaminante a píxeis da esteira. Tais erros ocorrem principalmente na borda dos objetos, onde a câmera acaba captando uma mistura de luz refletida por eles e pela esteira e também onde há maior chance de erro na atribuição da classe de referência durante a construção do conjunto de treinamento, porém no contexto desse sistema são de pouca relevância, considerando que o objetivo final é a ativação dos bicos quando houver presença de contaminantes.

Repetindo este processo utilizando os métodos de redução de dimensionalidade PCA, LDA e PLS, os resultados podem ser observados na tabela 3, de onde foi excluído o modelo CNN, pois não há sentido em aplicar convoluções sobre as variáveis latentes, por serem ortogonais e não possuírem uma sequência natural entre si da mesma forma como as bandas dos espectros possuem.

Como o número de dimensões máximo que pode ser extraído via LDA e PLS é restrito pelo número de classes, a formulação deste problema como tendo apenas 3 classes (esteira, contaminante e alimento) restringiria a quantidade de discriminantes e variáveis latentes a 2 e 3, respectivamente, o que implicaria em uma perda muito grande de informação para este problema de classificação. Para contornar esta situação foram aplicados estes métodos usando como referência as subclasses, ou seja, o alimento ou contaminante específico de cada espectro como referência, permitindo a seleção de até 18 discriminantes com a LDA e 19 variáveis latentes com a PLS.

Nota-se aqui que o argumento empírico realizado na subseção 2.2.1, de que métodos de redução de dimensionalidade costumam ser eficazes em dados obtidos via espectroscopia, se sustenta no caso desta aplicação, sendo que de forma geral

Figura 9 – Matriz de confusão obtida a partir da validação do modelo SVM com *kernel* RBF

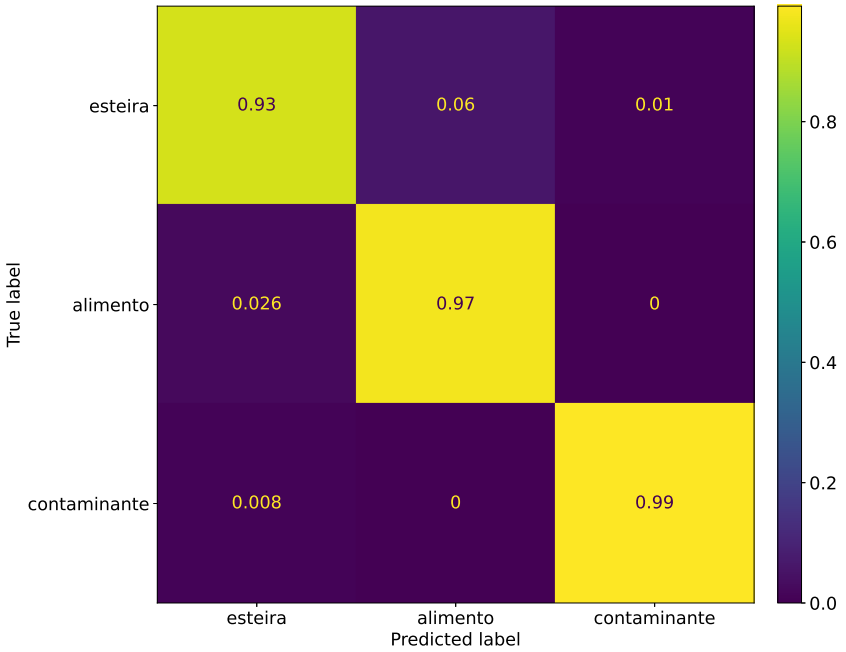


Tabela 2 – Acurácia dos modelos com redução de dimensionalidade

Modelo	**Parâmetros**	**Redução de dimensionalidade**	**Acurácia média na validação cruzada**
KNN	k = 5	PCA, 10 componentes	0.956
	k = 5	LDA, 16 discriminantes	0.952
	k = 5	PLS, 16 componentes	0.952
SVM - linear	k_r = 5	PCA, 20 componentes	0.879
	k_r = 1	LDA, 10 discriminantes	0.88
	k_r = 1	PLS, 12 componentes	0.889

Fonte: Autoria própria

os modelos apresentaram acurácia semelhante mesmo com redução expressiva na dimensionalidade do problema. O modelo KNN em particular apresentou um resultado interessante, onde sua acurácia aumentou mais de 5%, demonstrando como os métodos aplicados, além de apresentarem a vantagem de reduzir o volume de dados a serem processados com uma simples multiplicação matricial (reduzindo consideravelmente o tempo de treinamento e execução dos modelos), também podem contribuir para um aumento no percentual de classificações corretas. Isto provavelmente ocorre neste caso devido a um efeito regularizador da redução de dimensionalidade, que pode atenuar a influência que a variabilidade dos sinais em bandas como pouco poder preditivo tem sobre a fronteira de decisão do classificador.

A figura 10 apresenta 3 imagens, representando uma imagem hiper-espectral "crua", na qual foram escolhidas 3 bandas para representar os canais de cor RGB

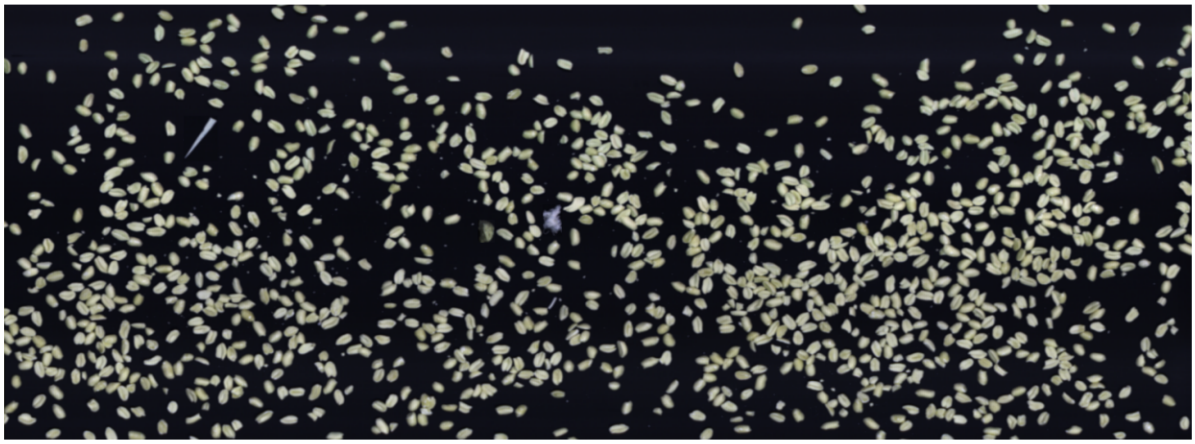
Tabela 3 – Acurácia dos modelos com redução de dimensionalidade

Modelo	**Parâmetros**	**Redução de dimensionalidade**	**Acurácia média na validação cruzada**
SVM - RBF	k_r = 10 , γ = 0.05	PCA 15 componentes	0.967
	k_r = 10 , γ = 0.01	LDA 18 discriminantes	0.966
	k_r = 5 , γ = 0.05	PLS, 14 componentes	0.964
ANN	2 camadas ReLU (150, 50) , $L2$ = 0.1	PCA, 30 componentes	0.97
	2 camadas ReLU (100, 50) , $L2$ = 0.01	LDA, 16 discriminantes	0.965
	2 camadas ReLU (100,50) , $L2$ = 0.01	PLS, 16 componentes	0.966

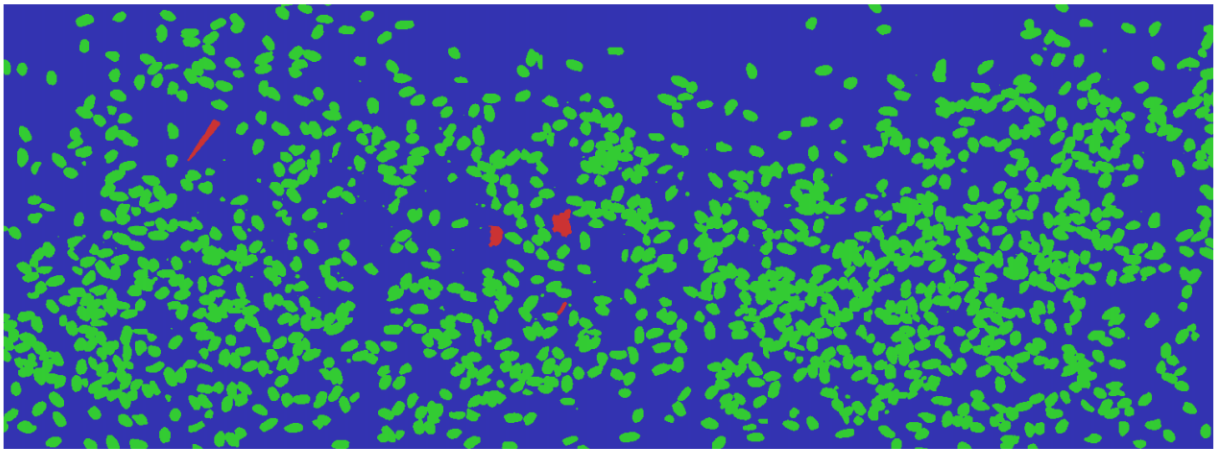
Fonte: Autoria própria

apenas para fins ilustrativos, o valor de referência atribuído a seus pixeis, e o resultado da segmentação utilizando um modelo SVM em combinação com PCA. Conforme mencionado anteriormente, os erros de classificação ocorrem principalmente na borda dos objetos, de forma que todos os modelos identificaram os contaminantes e alimentos corretamente, porém o contorno de cada objeto varia entre eles. Em termos práticos para o objetivo deste projeto, nenhum objeto passou pela câmera sem ser detectado em nenhuma das imagens de teste geradas, porém alguns falsos positivos foram observados em partes da esteira onde não haviam objetos e alguns farelos ou pedaços pequenos que se desprenderam de alimentos foram ocasionalmente identificados erroneamente como contaminantes. A especificação do projeto determina entretanto que objetos a partir de 4 milímetros quadrados devem ser detectados e descartados, de forma que objetos com menos de 9 pixeis podem ser ignorados.

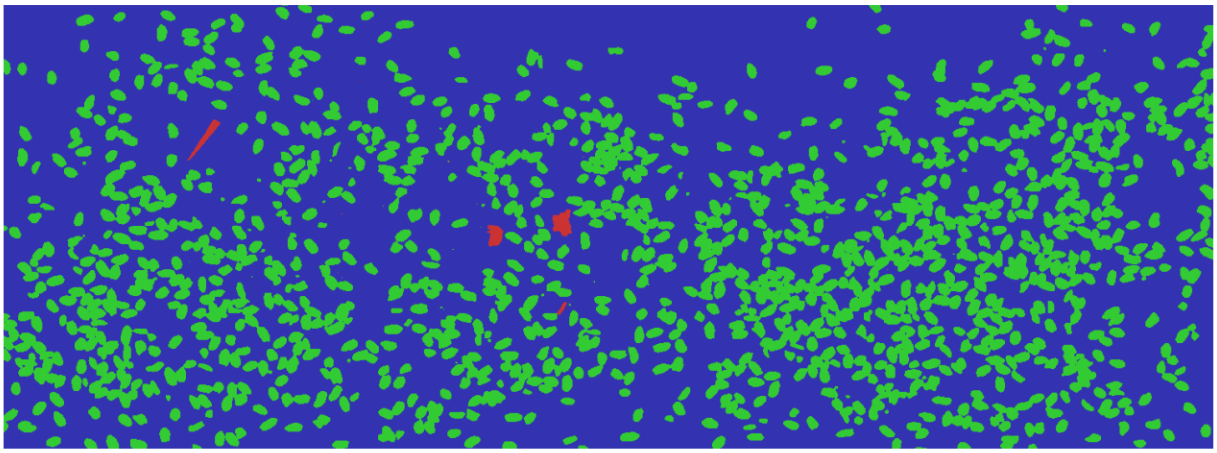
Figura 10 – Exemplo de segmentação de uma imagem hiper-espectral, onde vermelho indica um contaminante, verde as matérias primas e azul a esteira



(a) Representação de imagem hiper-espectral utilizando 3 das 224 bandas para representar os canais RGB



(b) A mesma imagem com pixeis manualmente rotulados em esteira, matéria prima e contaminante



(c) A mesma imagem, desta vez rotulada/segmentada por um modelo utilizando PCA + SVM com kernel RBF

Fonte: Autoria própria

4 DESENVOLVIMENTO DOS *SOFTWARES*

No sistema completo há 3 componentes de software com diferentes atribuições. Um deles é executado em um microcontrolador e interage com as componentes físicas, lendo sensores (com exceção da câmera hiper-espectral) e acionando atuadores. Já os outros dois são executados no computador principal, um responsável pelo processamento dos dados hiper-espectrais, fazendo utilização das chamadas da API da fabricante da câmera para obtenção dos dados em tempo real e execução dos modelos de classificação, enquanto o outro é responsável (dentre outras coisas) pela interface gráfica a ser exibida no painel e por coordenar a comunicação entre estes diferentes componentes e com o sistema supervisor central da fábrica, sendo que este último será referido como software do painel.

Nas seções seguintes deste capítulo são abordadas em mais detalhes as tarefas de cada componente e como elas são executadas.

4.1 SOFTWARE DO PAINEL

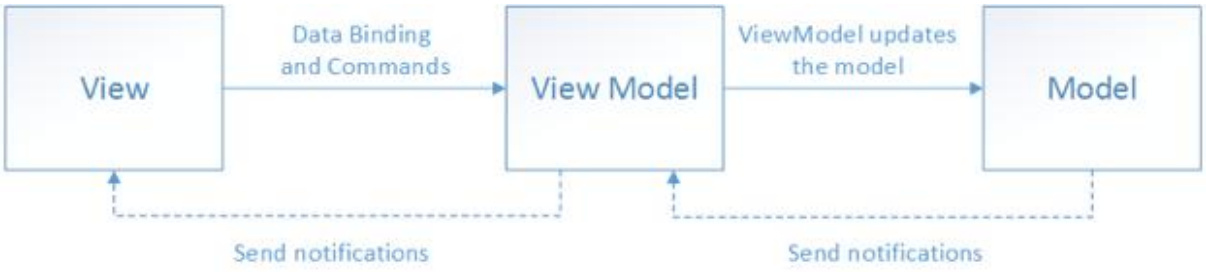
Apesar de referido aqui como *software* do painel, as atribuições deste componente vão além de apenas a exibição de uma tela. Outras atribuições incluem:

- Coordenar as ações entre o programa do microcontrolador e de processamento de dados
- Comunicação com o sistema supervisório da fábrica, informando estado de operação e parando o sistema caso ordenado
- Implementação de um sistema de *login*, com usuários do tipo administrador e operador, para restringir as ações que podem ser executadas por usuários do tipo operador
- Registro do histórico de detecções
- Registro de *logs* da operação
- Exibir tela para exibição de gráficos e informações sobre o histórico de detecções
- Exibir tela de configuração de diversos parâmetros do *software*, do sistema, da câmera e dos modelos de classificação
- Exibir telas de calibração da câmera e captura de amostras

Para o desenvolvimento deste componente foi utilizado como base outro *software*, já utilizado em outros tipos de processos, por conter parte das funcionalidades

desejadas neste projeto, além de já apresentar diversas telas com o estilo visual definido pela empresa. Este programa utiliza a linguagem de programação C# e se baseia no padrão de código *Model View ViewModel* (MVVM) utilizando a *framework Windows Presentation Foundation* (WPF). Nesta arquitetura, a camada de apresentação é formada por *views* definidas em *Extensible Application Markup Language* (XAML), uma linguagem declarativa baseada em *Extensible Markup Language* (XML), que são atualizadas por eventos gerados na *viewmodel*, que por sua vez contém a lógica da aplicação, enquanto os *models* representam a camada de dados a serem manipulados pela *viewmodel*, conforme indicado no diagrama da figura 11.

Figura 11 – Diagrama representando o padrão de código MVVM

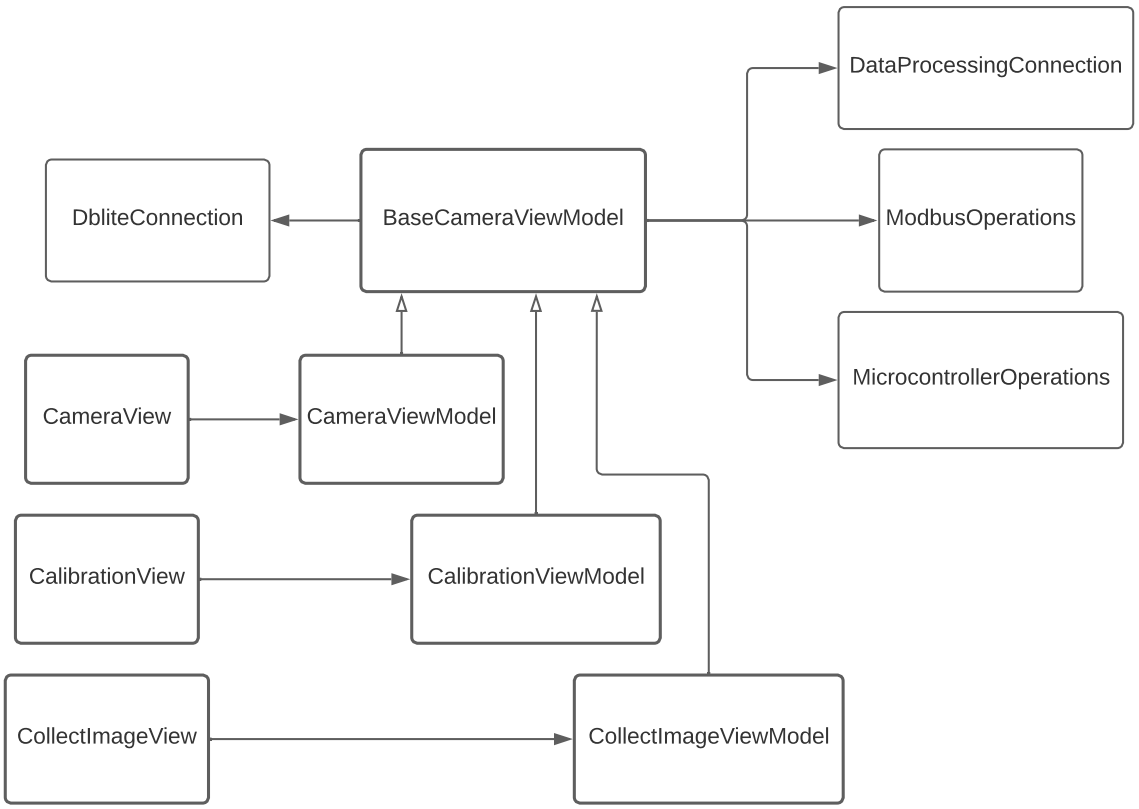


Fonte: (MICROSOFT, 2022)

Um exemplo dessa estrutura no contexto deste programa pode ser vista no diagrama da figura 12, que indica a organização das classes envolvidas no funcionamento das telas de operação da câmera. A classe CameraView define a tela de operação do sistema, para acompanhamento do estado atual e realizar ações de início e parada, enquanto CalibrationView é usada no processo de calibração da camera e CollectImageView é uma tela criada para tornar mais conveniente a coleta de amostras a serem utilizadas no treinamento do modelo. Cada *view* referencia uma *viewmodel*, que por sua vez desconhece a *view*, sendo as alteração do estado da tela causadas por eventos disparados pela *viewmodel* que são captados pela *view*. Para evitar duplicação de código estas *viewmodels* herdam da classe abstrata *baseviewmodel* diversos métodos comuns entre elas. As classes DataProcessingConnection, ModbusOperations e MicrocontrollerOperations são responsáveis pela comunicação com o processo que faz a comunicação com a câmera e processamento de dados, com o sistema supervisório da fábrica e com o microcontrolador, respectivamente, e são independentes das *viewmodels*. Por fim a classe DbliteConnection é uma classe estática que apenas agrega métodos *Create Read Update Delete* (CRUD) referentes às configurações, histórico de detecções e *logs* que são armazenadas em banco de dados local. Outras classes, incluindo outras *views* e *viewModels*, também existem no programa, mas para manter o foco nas partes essenciais do sistema e que tiveram maior atuação do autor deste

trabalho, não serão abordadas em detalhes neste documento.

Figura 12 – Diagrama representando as classes envolvidas nas telas de operação da câmera

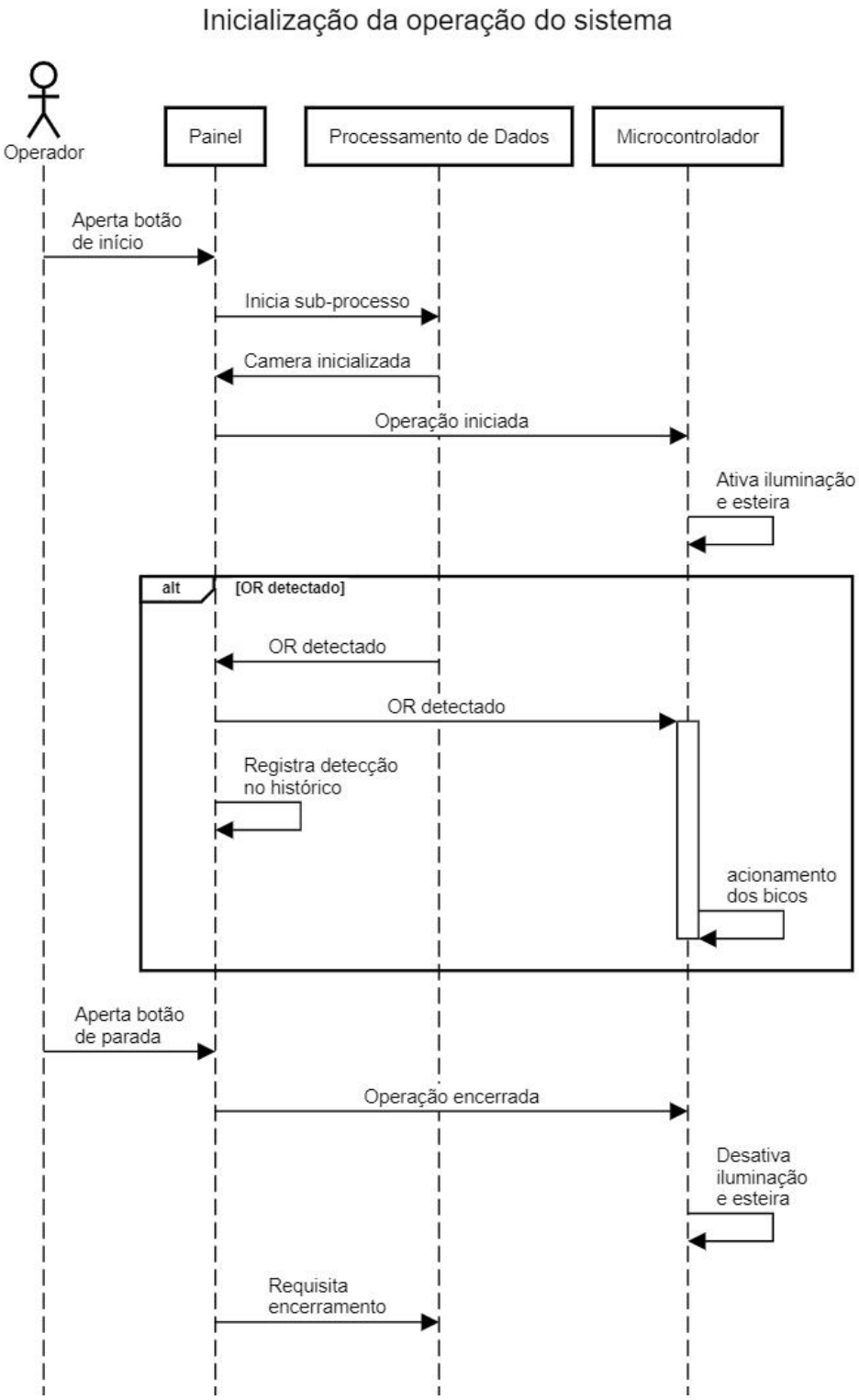


Fonte: Autoria própria

O diagrama de sequência da figura 13 demonstra a interação entre os diferentes componentes quando o operador requisita o início da operação, assumindo que o sistema não se encontre em uma condição inválida. Embora não representado no diagrama, o sistema está regularmente checando diversas condições como temperatura da iluminação, temperatura da câmera, botão de emergência pressionado ou não, silo de entrada do sistema vazio entre outros, de forma que qualquer situação anômala resulta em paralisação do sistema, exibição de um alerta com mensagem descritiva na tela e acionamento de indicadores luminosos na parte física. Logo após o operador apertar o botão de início também é verificado se há arquivos válidos de calibração e modelo de classificação selecionados, tal que a operação não chega a ser iniciada caso esta verificação falhe.

A forma como este programa se comunica com as outras componentes varia em cada caso. A comunicação com o software de processamento de dados ocorre via *socket Transmission Control Protocol* (TCP) endereçado ao *localhost*, enquanto com

Figura 13 – Diagrama de sequência representando a operação da câmera



Fonte: Autoria própria

o microcontrolador e sistema supervisório ocorre via comunicação serial, no caso do sistema supervisório utilizando o protocolo *Modbus* (no qual o software do painel atua como escravo) e no caso do microcontrolador utilizando um formato de mensagem customizado, porém também no modelo mestre/escravo, sendo o microcontrolador neste caso o escravo.

4.2 SOFTWARE DE PROCESSAMENTO DOS DADOS

Conforme mencionado anteriormente na sub-seção 3.1.2, embora a linguagem de programação Python tenha sido usada para treinar e validar diferentes modelos e métodos de redução de dimensionalidade, encontrando valores de seus hiperparâmetros que fossem adequados para o problema e comparando o poder preditivo de cada um deles, o uso desta linguagem implica em um *overhead* significativo. Considerando que o modelo deve realizar as classificações de cada *frame* em menos de 2 ms e que a API da câmera é fornecida pelo fabricante na linguagem C, foi optado por utilizar a linguagem C++ para o desenvolvimento deste programa. Além do recurso de interoperabilidade que permite a utilização direta de bibliotecas em C, a linguagem C++ também apresenta um número maior de recursos e uma biblioteca padrão mais abrangente, mas mantendo a capacidade de gerar programas muito eficientes.

A função essencial deste programa é receber os dados da câmera pela API do fabricante, aplicar o pré-processamento, reduzir a dimensionalidade e executar o algoritmo de classificação sobre eles, para informar o *software* do painel quando um OR é detectado e salvar imagens do instante em que isto ocorre. Em paralelo, também é monitorada a taxa de *frames* coletados por segundo, para conferir se está havendo flutuação excessiva em relação à taxa desejada, e temperatura da câmera, que possui limite operacional estabelecido pelo fabricante. Por fim, outra tarefa executada por este programa é o processo de calibração e captura de amostras, que embora do ponto de vista operacional sejam duas tarefas distintas, para este *software* são executadas da mesma maneira, capturando um número predeterminado de *frames* e os consolidando em uma imagem hiper-espectral.

O código é escrito de forma procedural, seguindo um fluxo simples. Conforme indicado no diagrama de sequência (figura 13), quando o operador requisita o início da operação, o *software* do painel inicia este programa, sendo repassadas diversas configurações escolhidas na tela do painel como argumentos de linha de comando, incluindo opções referentes ao funcionamento da câmera (por exemplo, a taxa de *frames* por segundo desejada), ao funcionamento dos modelos (caminho do arquivo do modelo) e as ações a serem tomadas sobre o resultado da classificação (quais classes devem ser rejeitas, em que local salvar as imagens das detecções, etc). A primeira etapa do código, então, envolve a conversão das *strings* recebidas como argumentos para seus respectivos tipos, a leitura de arquivos relevantes cujos caminhos foram

recebidos e a atribuição de valores a certas variáveis globais de acordo com as configurações. Em seguida, a *thread* principal do programa entra em *loop* infinito, na qual ela dorme e periodicamente checa o atual valor da temperatura e taxa de *frames* por segundo, enquanto outras *threads* lidam com o processamento dos dados da câmera.

O programa, consequentemente, se encerra quando o *software* do painel requisitar seu encerramento, tipicamente devido a uma requisição de parada por parte do operador, mas que também pode ocorrer caso algum problema (como sobreaquecimento da câmera ou da iluminação) seja detectado. No caso em que é executado para calibração da câmera ou coleta de amostra, o programa também se encerra após coletar um certo número de *frames* configurável e salvar a imagem resultante da combinação destes *frames*.

Embora os métodos de redução de dimensionalidade descritos na sub-seção 2.2.1 envolvam a utilização de algoritmos mais complexos para encontrar os autovetores das matrizes referentes a cada método, após encontrados estes vetores, sua aplicação se reduz a uma simples multiplicação matricial pela matriz de mudança de base, de forma que sua implementação é trivial, além de haver bibliotecas bem otimizadas para tal tarefa. A convolução discreta do FSG e a padronização dos dados por SNV também são de simples implementação. A execução dos modelos utiliza bibliotecas de código aberto, porém neste caso, foi necessário escrever código para converter os modelos treinados e salvos pelos códigos e bibliotecas em Python para a estrutura utilizada pelas bibliotecas usadas no código em C++.

Quando um pixel é classificado como pertencente a um contaminante, o código aguarda *frames* seguintes para averiguar se pixeis vizinhos também recebem tal classificação, contando assim a quantidade de pixeis de contaminantes contíguos. Caso ultrapassem o limiar estabelecido no capítulo 3 de 10 pixeis, são contabilizados como detecção de um OR, levando o programa a repassar tal informação ao *software* do painel via *socket* TCP e salvando uma imagem segmentada dos últimos *frames* a passarem pela câmera.

4.3 SOFTWARE DO MICROCONTROLADOR

O código do microcontrolador, escrito na linguagem C, deve executar as seguintes tarefas:

- Monitorar a velocidade da esteira, para após a detecção de um OR, calcular e acompanhar a sua posição
- Acionar os bicos de ar para descartar os ORs ao deixarem a esteira
- Fazer a leitura de diferentes sensores e botões presentes no sistema

- Ligar o acionamento do sistema de iluminação e dos motores da esteira quando o sistema estiver operando, ou desligá-los caso haja algum problema (sobre-aquecimento das lâmpadas, botão de emergência pressionado, etc)
- Comunicar-se com o computador principal, recebendo dados das detecções de ORs e ordens do sistema supervisor central e enviando o estado atual dos botões físicos do sistema e de alguns sensores

Para o monitoramento da velocidade por microcontrolador é acoplado a um *encoder* de posição instalado no eixo do motor da esteira, cuja posição é amostrada a cada 20 ms para calcular a frequência de rotação do eixo do motor e consequentemente a velocidade linear atual da esteira, a qual espera-se que se mantenha próxima de 35 cm/s, equivalente a cerca de 1.85 rotações por segundo. Quando a informação de que um OR foi detectado pela câmera é recebida, sua posição atual é estimada e guardada pelo microcontrolador, sendo atualizada logo após cada atualização da velocidade.

A cada amostragem o estado atual dos sensores e botões também são verificados, sendo então checado se alguma ação deve ser feita com base na leitura (por exemplo, desligar a iluminação e esteira caso algum sobre-aquecimento seja detectado). Para a comunicação, o microcontrolador continuamente confere se está recebendo alguma mensagem do computador principal, apenas enviando alguma resposta quando requisitado, sem nunca iniciar a troca de mensagens por conta própria.

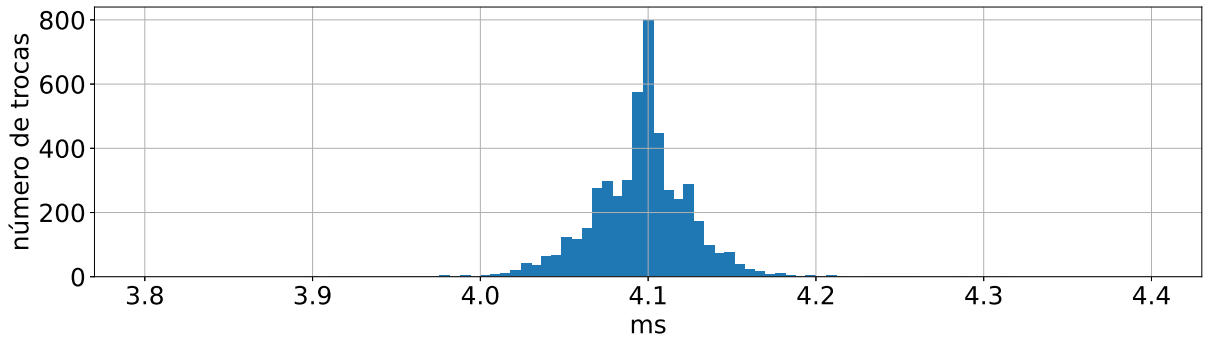
Como há um atraso entre o momento em que um OR passa pelo ponto de leitura da câmera e o momento em que a informação sobre este evento chega ao microcontrolador, junto à mensagem informando a detecção é enviada a quantidade de milissegundos entre a obtenção do *frame* e o início do envio da mensagem, que é somado ao tempo estimado para transmitir a mensagem pelo cabo de comunicação serial. Assim, a posição do objeto pode ser calculada a partir destes dados de acordo com a expressão

$$d_{ofe} = d_{cfe} - v \cdot (t_{proc} + t_{trans}) \,, \tag{29}$$

onde d_{ofe} é a distância do objeto até o fim da esteira, d_{cfe} do ponto de leitura da câmera até o fim da esteira, v a velocidade do objeto, t_{proc} o tempo de processamento e t_{trans} o tempo de transmissão da mensagem. O valor de d_{cfe} é uma constante definida no projeto da parte física do sistema, v e t_{proc} podem ser mensurados de forma relativamente precisa durante a operação, enquanto t_{trans} é difícil de determinar durante a operação, considerando que o computador e o microcontrolador operam em frequências diferentes e não possuem um *timer* sincronizado, mas pode ser estimado a partir da taxa de transmissão da comunicação serial.

As mensagens a serem enviadas pelo computador principal possuem ao todo 9 bytes: 1 indicando o tipo de mensagem, 1 com *flags* indicando o estado de operação, 4 indicando quais bicos devem ser ativados, 1 indicando a quantidade de milissegundos de t_{proc} e 2 de *Cyclic Redundancy Check* (CRC). Como cada byte precisa de 10 bits para ser transmitido, devido à utilização de um *start* bit e um *stop* bit, com a taxa de transmissão configurada para 57600 bits por segundo isto equivale a 1,56 ms para transmitir uma mensagem. Tal valor é apenas um limite inferior, entretanto, e ainda deve ser considerado o potencial *overhead* causado por chamadas do sistema operacional do computador, *drivers* dos dispositivos envolvidos e a própria lógica dos programas do microcontrolador e computador que lidam com essa troca de mensagens. Embora espera-se que estes fatores tenham um impacto pequeno, para confirmar esta hipótese foi criada um versão do código que mensura o tempo de envio e resposta de 5000 mensagens, considerando resposta também de 9 bytes, cujos resultados são exibidos no histograma da figura 14. O limite inferior no caso seria $1,56 \cdot 2 = 3,12$ ms, mas devido ao *overhead* mencionado anteriormente e uma certa variabilidade, observa-se na prática um valor em torno de 4,1 ms. Portanto, 2,05 ms se demonstra uma estimativa mais adequada para t_{trans}.

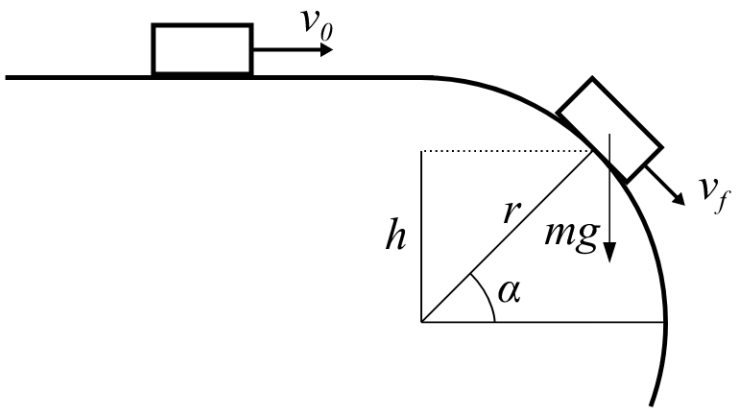
Figura 14 – Histograma representando o tempo decorrido em 5000 trocas de mensagens entre microcontrolador e computador, considerando envio e resposta com 9 bytes cada



Fonte: Autoria própria

Quando o OR chega ao fim da esteira, ainda deve ser considerado o tempo que o objeto leva para cair até o ponto onde o ar pressurizado ejetado pelos bicos irá atingi-lo. Para isso, primeiramente deve-se determinar a velocidade v_f no instante que o objeto deixa a esteira, o ângulo α em relação a horizontal no qual isso ocorre e a altura h em relação ao centro da esteira, conforme ilustrado na figura 15, onde v_0 representa a velocidade na esteira conforme mensurado pela amostragem do *encoder*, g a aceleração da gravidade, r o raio do fim da esteira (assumindo-se que tem o formato de um semicírculo) e m a massa do OR.

Figura 15 – Diagrama representando a saída de um objeto da esteira



Fonte: Autoria própria

Considerando θ como o ângulo em relação ao topo da esteira em um instante t (ou seja, quando o bloco entra em queda livre θ será o ângulo complementar de α), a variação de v pode ser descrita por uma equação diferencial, onde a aceleração é causada pela componente da força da gravidade tangencial ao semicírculo subtraída da força de atrito, sendo que a força normal equivale à componente da gravidade no sentido do centro do semicírculo subtraída da força centrífuga,

$$m \cdot \dot{v}(t) = m \cdot g \cdot sen(\theta(t)) - \mu_c(m \cdot g \cdot cos(\theta(t)) - \frac{m \cdot v(t)^2}{r}) \,, \tag{30}$$

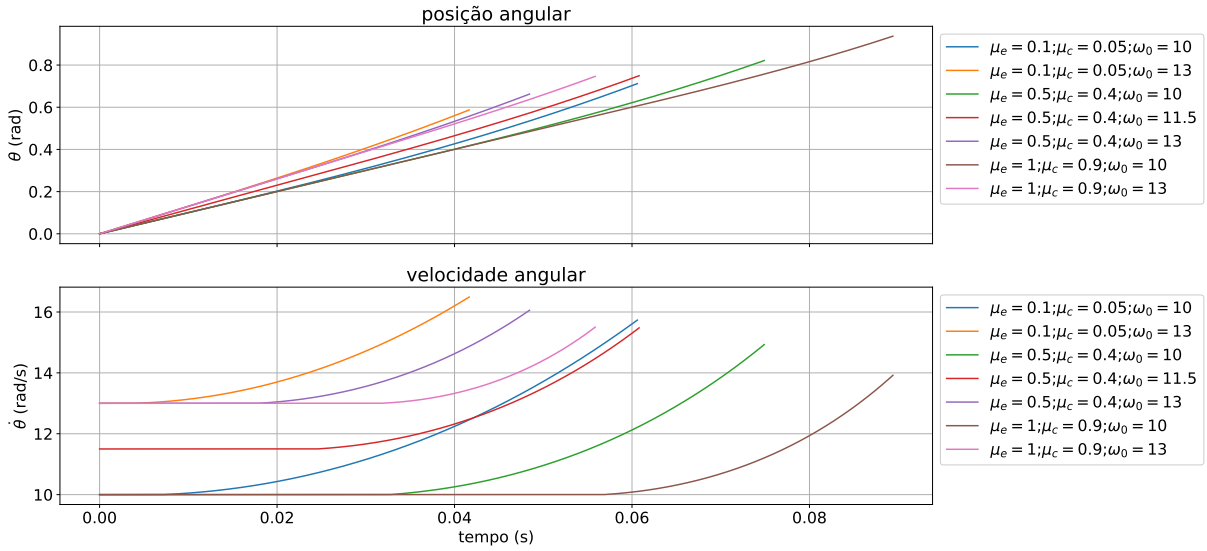
onde μ_c é o coeficiente de atrito cinético entre o OR e a esteira. Considerando que θ também varia com o tempo, substitui-se a velocidade v pela velocidade angular ω usando a relação $v = \omega \cdot r$,

$$\ddot{\theta}(t) = \frac{1}{r}[g \cdot sen(\theta(t)) - \mu_c(g \cdot cos(\theta(t)) - \dot{\theta}(t)^2 \cdot r)] \,. \tag{31}$$

Tal formulação é aplicável, entretanto, quando há deslizamento sobre a esteira, ou seja, a componente tangencial da força da gravidade é maior que a força de atrito estático, mas para valores pequenos de θ isto não é verdade, considerando que $sen(0) = 0$. Na prática, assumindo que ao chegar no fim da esteira o objeto tem a mesma velocidade da mesma, estando parado em relação a ela, a força de atrito é proporcional ao coeficiente de atrito estático μ_e e $\ddot{\theta} = 0$, o que implica em $\dot{\theta}$ constante e igual a $v_0 \cdot r$. Conforme θ aumenta, a componente da força da gravidade tangencial ao semicírculo aumenta em magnitude, enquanto a componente no sentido do centro diminui, consequentemente reduzindo também a força normal e a força de atrito. Apenas quando $g \cdot sen(\theta) > \mu_e(g \cdot cos(\theta) - \dot{\theta}^2 \cdot r)$ que o deslizamento passa a ocorrer e (31) se torna aplicável.

Tal equação é problemática entretanto, considerando que além de ser difícil de solucionar analiticamente, os valores do parâmetro μ tanto para o atrito estático quanto cinético de cada OR são desconhecidos. Para evitar a necessidade de calcular soluções numéricas para esta equação diferencial em tempo real no microcontrolador a cada nova detecção de OR, e considerando que tipicamente $\mu_e > \mu_c$, a solução desta equação é computada numericamente de maneira prévia assumindo $\mu_e = 0.5$ e $\mu_c = 0.4$, com condições iniciais $\theta(0) = 0$ e $\dot{\theta}(0) = \frac{0.35}{r}$. Para lidar com possíveis ORs com valores de μ significativamente maiores ou menores que os assumidos, as equações também são solucionadas para valores de μ_e entre 0.1 e 1, μ_c entre 0.05 e 0.9 e ω_0 entre 10 e 13, sendo estes resultados considerados na determinação da quantidade de tempo que o bico permanece ativado, com o propósito de escolher este intervalo de forma que mesmo sendo ativado um pouco antes ou depois do tempo ideal, o bico ainda assim descarte o objeto.

Figura 16 – Resultado do cálculo numérico da velocidade e posição angular



Fonte: Autoria própria

A partir da solução numérica deseja-se encontrar o ponto no qual o objeto se desprende da esteira e entra em queda livre, o que ocorre quando a componente da força da gravidade no sentido do centro do semicírculo passa a ser menor ou igual à força centrífuga, resultando em uma força normal nula, ou seja, quando $\dot{\theta}(t)^2 \cdot r \geq g \cdot cos(\theta(t))$. Sendo t_f o menor valor de t que satisfaz esta condição, então $v_f = \dot{\theta}(t_f) \cdot r$, $\alpha = \frac{\pi}{2} - \theta(t_f)$ e $h = r \cdot sen(\alpha)$. Os valores obtidos numericamente para $\theta(t)$ e $\dot{\theta}(t)$ de $t = 0$ até $t = t_f$, na faixa de valores de μ e ω_0 estipulados anteriormente, pode ser vista na figura 16.

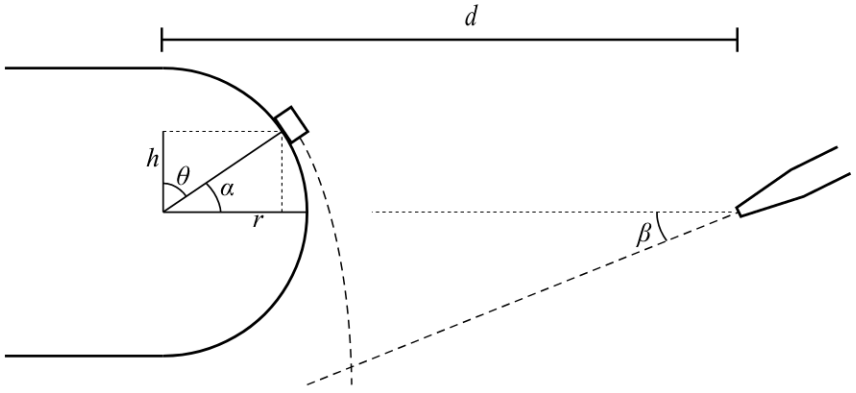
Tendo estes parâmetros é necessário, então, calcular a trajetória do objeto em queda livre, conforme representado na figura 17. Desprezando a resistência do ar

e considerando a força gravitacional como a única agindo sobre o objeto, pode-se descrever a posição do objeto a partir de coordenadas x_{or} e y_{or}

$$\begin{cases} x_{or}(t) = v_f \cdot sen(\alpha) \cdot t + x(0) \\ y_{or}(t) = -\frac{g \cdot t^2}{2} - v_f \cdot cos(\alpha) \cdot t + y(0) \end{cases} , \tag{32}$$

onde, adotando o centro do semicírculo como origem deste sistema de coordenadas, deduz-se que $x(0) = r \cdot cos(\alpha)$ e $y(0) = h$, enquanto os sinais negativos na expressão de $y(t)$ se devem à adoção de valores negativos para vetores na direção vertical com sentido ao chão.

Figura 17 – Diagrama representando a trajetória do objeto em queda livre e o ar emitido pelos bicos



Fonte: Autoria própria

Embora o ar expelido pelos bicos possa apresentar escoamento turbulento e não laminar, devido à sua proximidade da esteira, tais efeitos são desprezados, assim como o tempo de deslocamento do ar até o ponto que atinge o objeto, e aproxima-se a trajetória do ar expelido como uma simples reta, que neste sistema de coordenadas pode ser descrita como

$$y = tan(\beta) \cdot x + h_b - d \cdot tan(\beta) \, . \tag{33}$$

Desta forma, substituindo y e x em (33) por $x_{or}(t)$ e $y_{or}(t)$

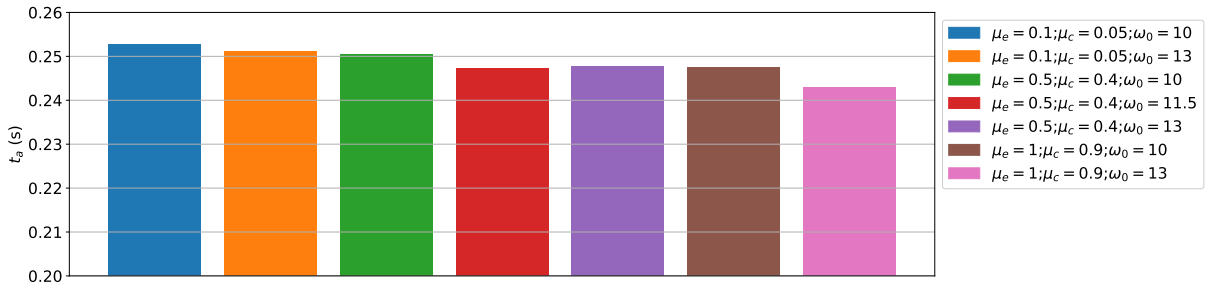
$$-\frac{g \cdot t^2}{2} - v_f \cdot cos(\alpha) \cdot t + y(0) = tan(\beta) \cdot (v_f \cdot sen(\alpha) \cdot t + x(0)) + h_b - d \cdot tan(\beta) \, , \tag{34}$$

o valor de t que satisfaz essa equação somado ao valor de t_f encontrado a partir da solução numérica de (31) resulta na quantidade de tempo t_a, contabilizada a partir da chegada do objeto no fim da esteira, a partir do qual o bico deve estar acionado.

Tomando os valores encontrados nas soluções da equação (31) encontradas numericamente para o momento em que o objeto entra em queda livre, os inserindo

na equação (34), resolvendo-a para encontrar t e finalmente somando este valor a t_f, os valores de t_a encontrados são exibidos na figura 18. Observa-se que na faixa de valores testados para μ_e, μ_c e ω_0, a variação do valor calculado para t_a não passa de 11 ms

Figura 18 – Tempos para a ativação dos bicos após objeto atingir fim da esteira, para diferentes valores de μ_e, μ_c e ω_0



Fonte: Autoria própria

Os bicos são mantidos acionados por um intervalo t_{ia}, sendo a ativação iniciada em $t_a - \frac{t_{ia}}{2}$ e encerrada em $t_a + \frac{t_{ia}}{2}$. Para a definição de t_{ia}, além dos 11 ms segundos de variação de t_a, podem ser considerados também os 20 ms de amostragem e até 1 ms de variabilidade na transmissão da mensagem do computador principal para o microcontrolador, somando 32 ms. Porém junto a estes fatores, também existem outras fontes de incerteza como erros de medição, pequenos desvios das medidas reais do sistema em relação ao projeto, influência de fatores desconsiderados nas equações (como resistência do ar, escoamento turbulento, tempo de deslocamento do ar pressionado do bico até o objeto e tempo de resposta dos diferentes atuadores e sensores envolvidos), além de incertezas em relação ao tempo de execução do *software*, especialmente o do painel, por ser escrito em uma linguagem gerenciada por coletor de lixo como C#. Embora o descarte de matéria-prima em condições aceitáveis devido a um acionamento prolongado dos bicos seja uma situação indesejável, o descarte de ORs é essencial e a passagem de um OR para etapas seguintes do processo produtivo devido a um tempo de acionamento muito curto é uma situação ainda pior. Portanto, para evitar que algum efeito não considerado implique em falha na rejeição do OR foi adotado t_{ia} de 250 ms.

5 CONCLUSÃO

No momento em que este documento é escrito o sistema ainda não havia sido integrado à linha de produção, devido a ajustes na parte física do sistema requisitados pelos responsáveis pela limpeza e manutenção dos equipamentos na fábrica e que fogem do escopo desta monografia, de forma que não se pode afirmar se na prática os objetivos citados na seção 1.3 foram atingidos. Os testes realizados até o momento se demonstraram promissores, sendo que em seu estado atual o sistema foi capaz de identificar e expulsar os objetos indesejados inseridos na esteira com alta acurácia. Em relação a estabilidade do *software* também não foi detectado nenhum problema, exceto alguns *bugs* visuais no painel, que foram levantados como pontos de melhoria para futuras versões mas que não afetam o funcionamento do processo.

Dito isso, algumas limitações já podem ser levantadas. Embora o espectro NIR seja bastante efetivo para identificar moléculas encontradas nos alimentos e contaminantes de interesse deste projeto, existem também materiais que não apresentam sobretons bem definidos nessa região do espectro luminoso, de forma que deve ser considerada a possibilidade de aparecerem na esteira contaminantes difíceis de observar no modelo de câmera hiper espectral utilizada. Também deve ser considerado que, assim como qualquer modelo estatístico, os modelos desenvolvidos para este projeto podem não ser confiáveis caso algum contaminante com perfil espectral muito diferente dos utilizados no treinamento apareça na esteira. A utilização de algum modelo bayesiano para estimar a incerteza de uma predição e rejeitar também aqueles objetos sobre os quais o sistema possui pouca confiança pode ser uma possível melhoria como forma de amenizar esta limitação.

Outro ponto de melhoria seria mover algumas atribuições do *software* do painel para o do processamento de dados, mais especificamente a comunicação com o microcontrolador, por envolver ações críticas em termos de tempo de execução. A reutilização de código desenvolvido anteriormente para outro projeto com requisitos temporais mais relaxados em relação a este contribuiu para a redução do tempo de desenvolvimento do sistema e cumprimento dos prazos de entrega, porém futuramente essa realocação de tarefas pode ser benéfica. Como no C++ não há gerenciamento automático de memória por um coletor de lixo como há no C#, o tempo de execução do código passa a ser mais previsível, o que permitiria o uso de um tempo de ativação menor dos bicos de ar e consequentemente menor desperdício de matéria-prima devido à sua ativação prolongada.

Neste projeto o foco principal foi a remoção de contaminantes sólidos, porém o trabalho realizado nele pode servir de base para futuramente readequar este sistema ou criar outros similares para além de remover contaminantes também realizar avaliações quantitativas ou qualitativas dos alimentos passando pela esteira e remover

aqueles que estiverem em um patamar considerado inadequado. Indo além do controle de qualidade de matéria-prima, também já há planos por parte da empresa de utilização de um sistema similar para avaliação não da matéria-prima mas do produto final, para identificar itens que por alguma falha no processo apresentam algum defeito, que também pode utilizar este sistema como base.

Nestes casos, outras limitações também devem ser consideradas. A classificação pixel a pixel para esse tipo de aplicação pode não ser adequada, sendo que modificar o processo de classificação para considerar o contexto espacial dos píxeis aqui ignorados e realizar as classificações em combinação com algum método de identificação de objetos seria o ideal. Esta abordagem também poderia melhorar a capacidade do sistema de identificar ORs pequenos e desconsiderar pixeis atribuídos à classe errada sem a utilização de um limiar fixo como foi feito neste projeto.

Dependendo do tipo de processo, outros tipos de atuadores podem ser mais adequados que a ação pneumática por bicos de ar, o que na versão atual do *software* desenvolvido não é suportado, devendo ser consideradas então possíveis adequações para sua utilização com outro tipo de atuação caso este sistema seja utilizado como base.

REFERÊNCIAS

AJANKI, Antti. **Example of k-nearest neighbour classification**. Disponível em: https://commons.wikimedia.org/wiki/File:KnnClassification.svg. Acesso em: 1 out. 2022.

BADARÓ, Amanda Teixeira; GARCIA-MARTIN, Juan Francisco; CARMEN LÓPEZ-BARRERA, María del; BARBIN, Douglas Fernandes; ALVAREZ-MATEOS, Paloma. Determination of pectin content in orange peels by near infrared hyperspectral imaging. **Food Chemistry**, v. 323, p. 126861, 2020. ISSN 0308-8146. DOI: https://doi.org/10.1016/j.foodchem.2020.126861.

DUDA, Richard O.; HART, Peter E.; STORK, David G. **Pattern Classification**. 2. ed. [*S.l.*]: Wiley, 2001. ISBN 978-0-471-05669-0.

HARRIS, Daniel C.; BERTOLUCCI, Michael D. **Symmetry and Spectroscopy**: An Introduction to Vibrational and Electronic Spectroscopy. [*S.l.*]: Dover Publications, 1978.

HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. **Elements of Statistical Learning 2nd ed.** [*S.l.*: *s.n.*], 2009. v. 27.

JOLLIFFE, Ian T. A Note on the Use of Principal Components in Regression. **Journal of the Royal Statistical Society. Series C (Applied Statistics)**, [Wiley, Royal Statistical Society], v. 31, n. 3, p. 300–303, 1982. ISSN 00359254, 14679876. DOI: https://doi.org/10.2307/2348005.

LARHMAM. **Maximum-margin hyperplane and margin for an SVM trained on two classes**. Disponível em: https://commons.wikimedia.org/wiki/File:SVM_margin.png. Acesso em: 15 out. 2022.

METROHM. NIR Spectroscopy - A guide to near-infrared spectroscopic analysis. **Metrohm Monograph**, 2013.

MICROSOFT. **The Model-View-ViewModel Pattern**. Disponível em: https://learn.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm. Acesso em: 20 nov. 2022.

MOHAMED, E.S.; SALEH, A.M.; BELAL, A.B.; GAD, Abd_Allah. Application of near-infrared reflectance for quantitative assessment of soil properties. **The Egyptian Journal of Remote Sensing and Space Science**, v. 21, n. 1, p. 1–14, 2018. ISSN 1110-9823. DOI: https://doi.org/10.1016/j.ejrs.2017.02.001.

OLIVEIRA, M.M.; CRUZ-TIRADO, J.P.; ROQUE, J.V.; TEÓFILO, R.F.; BARBIN, D.F. Portable near-infrared spectroscopy for rapid authentication of adulterated paprika powder. **Journal of Food Composition and Analysis**, v. 87, p. 103403, 2020. ISSN 0889-1575. DOI: https://doi.org/10.1016/j.jfca.2019.103403.

PASQUINI, Celio. Near infrared spectroscopy: Fundamentals, practical aspects and analytical applications. **Journal of the Brazilian Chemical Society**, v. 14, 2 2003. ISSN 01035053. DOI: 10.1590/S0103-50532003000200006.

PRIETO, Nuria; PAWLUCZYK, Olga; DUGAN, Michael Edward Russell; AALHUS, Jennifer Lynn. A Review of the Principles and Applications of Near-Infrared Spectroscopy to Characterize Meat, Fat, and Meat Products. **Applied Spectroscopy**, v. 71, n. 7, p. 1403–1426, 2017. PMID: 28534672. DOI: https://doi.org/10.1177/0003702817709299.

QUELAL-VÁSCONEZ, Maribel Alexandra; LERMA-GARCÍA, María Jesús; PÉREZ-ESTEVE, Édgar; ARNAU-BONACHERA, Alberto; BARAT, José Manuel; TALENS, Pau. Changes in methylxanthines and flavanols during cocoa powder processing and their quantification by near-infrared spectroscopy. **LWT**, v. 117, p. 108598, 2020. ISSN 0023-6438. DOI: https://doi.org/10.1016/j.lwt.2019.108598.

SUPPORT-VECTOR Networks. **Machine Learning**, v. 20, 3 1995. ISSN 15730565. DOI: 10.1023/A:1022627411411.

THARWAT, Alaa; GABER, Tarek; IBRAHIM, Abdelhameed; HASSANIEN, Aboul Ella. Linear discriminant analysis: A detailed tutorial. **AI Commun.**, v. 30, p. 169–190, 2017.

WEGELIN, Ja. A survey of Partial Least Squares (PLS) methods, with emphasis on the two-block case. **Technical Report 371, Department of Statistics, University of Washington, Seattle, 2000.**, v. 371, 2000.

ZHU, Shichao; CHEN, Honghui; WANG, Mengmeng; GUO, Xuemei; LEI, Yu; JIN, Gang. Plastic solid waste identification system based on near infrared

spectroscopy in combination with support vector machine. **Advanced Industrial and Engineering Polymer Research**, v. 2, n. 2, p. 77–81, 2019. ISSN 2542-5048. DOI: https://doi.org/10.1016/j.aiepr.2019.04.001.