**Universidade dos Açores**

Pós-Graduação em Programação, Robótica e Inteligência Artificial (PRIA)

Relatório de Integração de Aprendizagem Automática

Carro Robótico com Visão Computacional usando HuskyLens

Autores: João Pavão, Tiago Martins

Docente: Prof. Armando B. Mendes

Ponta Delgada, 23 de maio de 2025

# Resumo

Este relatório descreve o desenvolvimento de um projeto prático no âmbito da unidade curricular de Aprendizagem Automática, inserido na Pós-Graduação PRIA da Universidade dos Açores. O projeto consiste na construção de um carro robótico autónomo equipado com a câmara HuskyLens, capaz de realizar tarefas de reconhecimento visual e decisão em tempo real.

O relatório explora a ligação entre os componentes físicos do projeto (Arduino UNO, ponte H L298P, HuskyLens) e os conceitos de Machine Learning lecionados na UC, como classificação supervisionada, KNN, redes neurais e visão computacional. São ainda apresentadas perspetivas futuras, incluindo a expansão com Raspberry Pi e a preparação para eventos como o AzoresBot.

**Palavras-chave:** Aprendizagem Automática, Visão Computacional, HuskyLens, Arduino, Robótica Educacional, IA Embarcada.

# Índice

[Resumo 2](#_Toc199154019)

[Índice 3](#_Toc199154020)

[1. Introdução 4](#_Toc199154021)

[2. Análise Detalhada da HuskyLens 5](#_Toc199154022)

[2.1 Características Físicas e Hardware 5](#_Toc199154023)

[2.2 Processador Kendryte K210 5](#_Toc199154024)

[2.3 Modos de Operação da HuskyLens 6](#_Toc199154025)

[2.4 Machine Learning e Inteligência Artificial Embutida 7](#_Toc199154026)

[3. Relação com a Aprendizagem Automática 8](#_Toc199154027)

[3.1 Tipos de aprendizagem 8](#_Toc199154028)

[3.2 Comparativo com os modelos da UC 8](#_Toc199154029)

[3.3 Frameworks e Processamento 9](#_Toc199154030)

[4. Contribuição do Projeto para a Aprendizagem 12](#_Toc199154031)

[5. Plano de Interação do Robô com a HuskyLens 13](#_Toc199154032)

[6. Expansão do Projeto com Raspberry Pi 14](#_Toc199154033)

[7. Bibliografia e Recursos 16](#_Toc199154034)

[7.1 Bibliografia 16](#_Toc199154035)

[7.2 Webgrafia 17](#_Toc199154036)

[7.3 Prompts utilizados (ChatGPT/OpenAI) 18](#_Toc199154037)

## 1. Introdução

O presente relatório visa documentar a integração de conceitos de Aprendizagem Automática (ML) num projeto prático de robótica baseado na câmara AI HuskyLens. O projeto consiste na construção de um carro robótico com capacidades de visão computacional, utilizando como microcontrolador o Arduino, uma ponte H L298P, motores DC, rodas, estrutura em 3D e possibilidade de integração de um sensor ultrassónico HC-SR04.

A HuskyLens é uma câmara de visão computacional com modelos de IA pré-treinados e permite a interação direta com algoritmos de classificação e reconhecimento de objetos, rostos, cores e linhas, sem necessidade de programação complexa.

## 2. Análise Detalhada da HuskyLens

### 2.1 Características Físicas e Hardware

- Sensor de imagem: OV2640 (2.0 Megapixeis)

- Interface: UART / I2C / SPI

- Display integrado: ecrã LCD para visualização em tempo real

- Alimentação: 3.3V ou 5V

- Microcontrolador principal: Kendryte K210

### 2.2 Processador Kendryte K210

* Chip especializado em IA embarcada, com foco em visão computacional e reconhecimento de áudio.
* Dual-core RISC-V 64-bit, cada núcleo com FPU de precisão dupla (IEEE754).
  + O processador tem dois núcleos (dual-core), usando a arquitetura RISC-V, que é open source e muito eficiente.
  + Cada núcleo é 64 bits, o que significa que pode manipular grandes quantidades de dados e memória.
  + Inclui FPU (Floating Point Unit) com precisão dupla, de acordo com o padrão IEEE 754, o que permite realizar cálculos matemáticos com vírgulas decimais com alta precisão — essencial para algoritmos de IA, redes neurais e processamento de imagem.
* Inclui KPU (Unidade de Processamento Neural) dedicada para executar redes neuronais convolucionais (CNNs)
  + A KPU é um acelerador de IA dedicado, embutido no chip, que serve para executar redes neuronais, especialmente CNNs (Convolutional Neural Networks), que são usadas em visão computacional.
  + Ela tira a carga de trabalho dos núcleos principais do processador.
  + Permite que modelos de IA façam inferência (reconhecimento) de imagens de forma muito mais rápida e eficiente.
* Capaz de realizar inferência em tempo real com modelos quantizados de até 5.9 MB (ex: YOLOv2-tiny).
* 8 MB de memória SRAM interna, sendo 2 MB exclusivos para a IA.
* Baixo consumo de energia e operação em larga faixa de temperatura (-40 °C a 125 °C).
* Integra aceleradores de FFT, AES e SHA256, facilitando aplicações com segurança e análise de sinais.
  + FFT (Fast Fourier Transform): usado em análise de sinais, como áudio e frequências.
  + AES (Advanced Encryption Standard): padrão de criptografia para segurança de dados.
  + SHA256 (Secure Hash Algorithm): usado para gerar códigos únicos (hashes) para verificação de integridade dos dados.
  + Esses aceleradores permitem processar áudio, proteger dados e executar análises complexas com rapidez e eficiência.

### 2.3 Modos de Operação da HuskyLens

A HuskyLens é uma câmara de visão computacional com algoritmos de IA integrados e interface interativa. Ela suporta múltiplos modos de funcionamento autónomos, prontos para uso:

* **Reconhecimento Facial**
  + Deteta e reconhece rostos humanos com alta precisão. Permite registar múltiplas identidades.
* **Reconhecimento de Objetos**
  + Permite treinar objetos por exemplo visual, associando imagens captadas a etiquetas (rótulos) definidas pelo utilizador.
* **Deteção de Linhas (Line Following)**
  + Ideal para carros robóticos; identifica linhas curvas ou retas para seguimento autónomo.
* **Reconhecimento de Cores**
  + Deteta e classifica cores específicas aprendidas pelo utilizador com um simples toque.
* **Leitura de Códigos QR e de Barras**
  + Capaz de ler códigos QR e barras 1D (como EAN-13), permitindo comunicação via imagem.
* **Deteção de Objetos Pré-treinados (Object Tracking)**
  + Usa modelos integrados (ex: Tiny YOLO) para detetar categorias genéricas (pessoa, carro, etc).
* **Aprendizagem por Exemplos (Supervised Learning)**
  + O utilizador pode mostrar imagens diferentes para cada categoria e a HuskyLens aprende com base nessas amostras visuais.

### 2.4 Machine Learning e Inteligência Artificial Embutida

A HuskyLens integra algoritmos de Machine Learning supervisionado e Deep Learning, sem necessidade de programação externa:

* Aprendizagem direta na câmara, usando apenas o botão físico ou interface serial (I2C/UART).
* Permite treinar modelos diretamente a partir da imagem da câmara, com feedback visual imediato.
* O processo é baseado em classificação supervisionada: a câmara associa uma imagem captada a um rótulo fornecido pelo utilizador.
* O reconhecimento é feito em tempo real, com feedback visual (retângulo, ID, confiança) no ecrã embutido.
* Possui modelos pré-treinados otimizados, mas com capacidade de reaprendizagem contínua baseada em novos exemplos fornecidos.

## 3. Relação com a Aprendizagem Automática

### 3.1 Tipos de aprendizagem

A HuskyLens utiliza predominantemente aprendizagem supervisionada, um tipo de Machine Learning onde o sistema aprende a partir de exemplos rotulados fornecidos pelo utilizador.

* O utilizador apresenta à câmara imagens associadas a rótulos (por exemplo, “Garrafa”, “Pessoa A”), e a HuskyLens utiliza essas amostras para aprender a classificar novas imagens com base na similaridade.
* Este tipo de aprendizagem está em sintonia com os conceitos trabalhados na disciplina, como a classificação supervisionada, e modelos como o KNN (K-Nearest Neighbors).
* Embora a HuskyLens utilize redes neuronais embarcadas, o funcionamento é análogo a um classificador supervisionado: reconhece padrões em imagens com base nos exemplos anteriores.
* Cada classe é aprendida interativamente, e a HuskyLens armazena os vetores de características para comparar com futuras imagens captadas em tempo real.

### 3.2 Comparativo com os modelos da UC

Durante a unidade curricular foram estudados vários algoritmos de Machine Learning, cuja aplicação pode ser observada ou inspirada nas funcionalidades do projeto com a HuskyLens e o processador Kendryte K210:

**KNN (K-Nearest Neighbors)**

Reflete a lógica de reconhecimento baseado em exemplos usada pela HuskyLens. Quando o utilizador fornece imagens rotuladas, a câmara classifica novas imagens comparando-as com as mais semelhantes, como num sistema KNN.

**Redes Neuronais**

A HuskyLens executa inferência com redes neuronais convolucionais (CNNs) otimizadas no processador Kendryte K210, através da KPU (Kendryte Processing Unit). Este componente foi desenvolvido para realizar inferência em tempo real com modelos quantizados, sem necessidade de computação externa.

**Classificação e Regressão**

O projeto base atual utiliza classificação supervisionada, mas pode evoluir para regressão supervisionada, por exemplo:

* Estimar a distância de um objeto com sensores como o HC-SR04.
* Associar valores contínuos a variáveis visuais detetadas pela câmara.

**Clustering (Agrupamento Não Supervisionado)**

Embora não implementado diretamente, futuras versões do projeto poderão incluir técnicas de agrupamento de imagens captadas, permitindo identificar padrões emergentes sem rótulos prévios, útil em análise de comportamento ou deteção de anomalias.

### 3.3 Frameworks e Processamento

Embora a HuskyLens funcione de forma autónoma — executando os seus próprios modelos de Inteligência Artificial diretamente no dispositivo — é possível ampliar o projeto com o uso de frameworks de Machine Learning (ML) e Visão Computacional externos. Esta abordagem traz maior controlo, flexibilidade e permite experimentar algoritmos personalizados que vão além dos modelos pré-definidos da HuskyLens.

**TensorFlow**

O TensorFlow é uma das bibliotecas mais populares para criação e treino de redes neuronais profundas (Deep Learning).

* Permite que os alunos construam os seus próprios modelos de IA adaptados à tarefa desejada (ex.: reconhecimento de plantas locais, identificação de gestos).
* Depois de treinado, o modelo pode ser quantizado (otimizado para ocupar menos espaço e funcionar mais rápido) e convertido para rodar em dispositivos embarcados, como o Kendryte K210, através de ferramentas como o NNCase.
* Ideal para projetos mais avançados com Raspberry Pi, câmaras USB ou integração com a HuskyLens como sensor auxiliar.

**scikit-learn**

Biblioteca leve e poderosa para algoritmos clássicos de ML.

* Oferece métodos de classificação (ex.: KNN, SVM), regressão (valores contínuos) e clustering (agrupamento automático).
* Pode ser usada para analisar os dados recolhidos por sensores como:
* A câmara HuskyLens (deteção de objetos, rostos, etc.)
* Sensores adicionais (distância, luz, temperatura)
* Com esses dados, é possível criar classificadores externos, treinar modelos no computador e depois aplicar os resultados no robô.

Ideal para análise de dados científicos no contexto educacional, sem necessidade de redes neuronais complexas.

**OpenCV**

É a biblioteca padrão para trabalhar com imagens e vídeo em tempo real.

* Permite capturar, processar e analisar imagens em tempo real de webcams ou câmaras do robô.
* Funções principais:
  + Pré-processamento: cortar, redimensionar, converter para tons de cinza.
  + Deteção de bordas e contornos: útil para localizar objetos ou formas.
  + Extração de características: preparar dados visuais para serem usados por algoritmos de ML.

Ideal para criar pipelines completos de visão computacional personalizada, que podem depois ser comparados com a performance da HuskyLens.

A HuskyLens não permite o upload de modelos treinados com TensorFlow ou scikit-learn. Ela executa apenas os seus próprios modelos internos embarcados no processador Kendryte K210, já otimizados e pré-programados pelo fabricante (DFRobot).

Ou seja:

* Não se pode treinar um modelo em Python e enviar para a HuskyLens.
* Não se pode modificar os algoritmos internos da HuskyLens.
* Pode-se interagir com ela apenas via comandos seriais (I2C ou UART).

Mesmo que não se possa modificar a IA da HuskyLens, pode-se usá-la como um sensor inteligente e enviar os seus dados para outro sistema (como um computador ou Raspberry Pi), onde aí sim pode-se usar TensorFlow, scikit-learn ou OpenCV.

## 4. Contribuição do Projeto para a Aprendizagem

Este projeto proporcionou uma experiência de aprendizagem significativa ao integrar conceitos teóricos de Aprendizagem Automática com a implementação prática em robótica educacional. Destacam-se os seguintes contributos:

* Aplicação concreta da aprendizagem supervisionada:

Tivemos a oportunidade de explorar diretamente o funcionamento de um sistema de classificação supervisionada, treinando a HuskyLens com exemplos visuais associados a rótulos e observando os resultados em tempo real. Esta abordagem facilitou a compreensão dos conceitos de treino, generalização e erro de classificação.

* Integração de inteligência artificial embarcada em sistemas físicos:

A utilização da HuskyLens demonstrou como é possível aplicar IA embarcada em robôs autónomos, sem necessidade de computadores externos. Esta experiência revelou o potencial da computação de borda ou na periferia (edge computing) em aplicações educacionais e experimentais.

* Análise crítica das limitações dos modelos pré-treinados vs. modelos personalizados:

O projeto promoveu a reflexão sobre as vantagens e desvantagens dos modelos integrados da HuskyLens em comparação com frameworks externas como TensorFlow ou scikit-learn. Discutiu-se o equilíbrio entre facilidade de uso e flexibilidade, incentivando a escolha consciente da abordagem mais adequada para cada problema.

* Consolidação da ligação entre teoria e prática através de hardware educacional acessível:

Ao construir o robô com componentes acessíveis (Arduino, ponte H, HuskyLens), pudemos materializar conceitos aprendidos na UC de Aprendizagem Automática. A experiência reforçou a importância do "aprender fazendo", especialmente no ensino de IA, robótica e ciência computacional.

## 5. Plano de Interação do Robô com a HuskyLens

A proposta prática para a demonstração do robô equipado com a HuskyLens, discutida no âmbito da UC de Programação com o Professor Mathias Funck, contempla o desenvolvimento de um conjunto de tarefas interativas baseadas em visão computacional, onde o comportamento do robô é determinado pela interpretação da cena visual captada pela câmara.

As funcionalidades previstas incluem:

* Seguimento de linha através do modo line tracking da HuskyLens, permitindo que o robô percorra trajetos definidos com curvas ou ramificações.
* Paragem ou mudança de direção quando a câmara reconhece um objeto ou uma cor previamente treinada, possibilitando reações contextuais em ambientes controlados.
* Leitura de códigos QR ou etiquetas visuais, desencadeando respostas automáticas como a execução de comandos, mensagens ou alterações no percurso.

Estas interações serão implementadas utilizando estruturas condicionais programadas no Arduino, estabelecendo uma ponte entre a lógica de controlo do microcontrolador e as capacidades de reconhecimento visual da HuskyLens. A combinação destes elementos visa criar um sistema autónomo simples, mas eficaz, ilustrando o potencial da IA embarcada em contextos educativos e experimentais.

## 6. Expansão do Projeto com Raspberry Pi

Com base na experiência adquirida com a HuskyLens e o Arduino, poderá se desenvolver uma nova versão do projeto, assente numa arquitetura mais robusta e flexível, centrada no Raspberry Pi 5 como unidade de processamento principal. Esta evolução permitirá explorar conceitos mais avançados de robótica e inteligência artificial, como autonomia total, planeamento de trajetória e aprendizagem profunda.

O novo carro robótico será desenhado como uma plataforma modular e open source, com múltiplas funcionalidades:

* Execução de modelos personalizados de Machine Learning treinados com ferramentas como TensorFlow, Keras, YOLOv5/v8 e MediaPipe, permitindo o reconhecimento de objetos, deteção de movimento, rastreamento facial ou gestual, e segmentação de cenas.
* Utilização de câmaras Pi (CSI) e webcams USB com alta resolução, suportando visão estéreo, análise em profundidade ou deteção simultânea de múltiplos alvos.
* Navegação autónoma em ambientes estruturados e não estruturados (labirintos, trilhos, obstáculos dinâmicos), com base em algoritmos de mapeamento, localização e planeamento de rota.
* Fusão sensorial com sensores adicionais (ultrassónicos, LDR, IMU, GPS, sensores de cor ou gás), aumentando a fiabilidade e a tomada de decisão do sistema.
* Integração com o sistema ROS (Robot Operating System), permitindo:
  + Modularização do código com nós ROS independentes para visão, sensores, motores e lógica de decisão.
  + Utilização de simuladores 3D como Gazebo para testes prévios.
  + Comunicação eficiente entre os diferentes componentes através de tópicos e serviços.
  + Expansão futura com algoritmos de navegação avançada, como SLAM, path planning e controle adaptativo.
* Comparação entre abordagens embarcadas (HuskyLens) e programáveis (Raspberry Pi), refletindo sobre as vantagens, limitações e contextos ideais para cada tecnologia.

Este projeto representa uma evolução natural no percurso formativo, combinando os fundamentos da Aprendizagem Automática com práticas reais de robótica autónoma, visão computacional e sistemas distribuídos. A versão baseada no Raspberry Pi será programada integralmente por nós, oferecendo controlo total sobre o comportamento do robô e promovendo competências transversais em programação, eletrónica, IA e integração de sistemas.

Este sistema poderá ser apresentado em mostras científicas e competições educacionais, como o AzoresBot, demonstrando o potencial da aprendizagem baseada em projetos para o ensino de ciência e tecnologia.

## 7. Bibliografia e Recursos

### 7.1 Bibliografia

**Inteligência Artificial e Machine Learning**

1. Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
2. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O’Reilly Media.
3. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.
4. Liu, Y. (2024). *Python Machine Learning by Example: With Real-World Use Cases* (4th ed.). Packt Publishing.
5. Mzili, T. (2025). *Innovations in Optimization and Machine Learning*.
6. Ryan, M. (2025). *Machine Learning for Tabular Data: XGBoost, Deep Learning, and AI*.
7. Selvi, G. (2025). *IoT and Machine Learning for Smart Applications*.

**Visão Computacional com Python e OpenCV**

1. Chen, J. (2023). *Learn OpenCV with Python by Examples: Implement Computer Vision* (2nd ed.).
2. Chesterfield, G. (2024). *Advanced Image Processing with Python and OpenCV*.
3. Mohaideen, K. (2024). *Industrial Vision Systems with Raspberry Pi: Python and OpenCV*.
4. Nuti, G. (2024). *Neural Network Computer Vision with OpenCV 5*.
5. Beyeler, M. (2017). *Machine Learning for OpenCV: A Practical Introduction to the World of Machine Learning and Image Processing using OpenCV and Python*. Packt Publishing.
6. Laganière, R. (2011). *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing.

**Arduino, Sensores e Robótica Educativa**

1. Ziemann, V. (2023). *A Hands-On Course in Sensors Using the Arduino and Raspberry Pi* (2nd ed.).
2. Guillen, G. (2024). *Sensor Projects with Raspberry Pi: IoT and Digital Image Processing* (2nd ed.).

**Raspberry Pi e Robótica Avançada**

1. Cazemier, F. (2024). *The Comprehensive Guide to Build Raspberry Pi 5 Robotics Projects*.
2. STEM School. (2025). *Raspberry Pi 101: The Ultimate Beginner’s Guide*.
3. Li, C. (2024). *Record Weather Data with Arduino and Solar Power: Use Sensors Effectively*.

### 7.2 Webgrafia

- [Wiki Oficial da HuskyLens - DFRobot]

(<https://wiki.dfrobot.com/HUSKYLENS_V1.0_SKU_SEN0305_SEN0336>)

- [Datasheet Kendryte K210]

(<https://cdn.hackaday.io/files/1654127076987008/kendryte_datasheet_20181011163248_en.pdf>)

- [HuskyLens WIKI Document.pdf]

(<https://raw.githubusercontent.com/DFRobot/Wiki/master/SEN0305/res/HuskyLens%20WIKI%20Document.pdf>)

### 7.3 Prompts utilizados (ChatGPT/OpenAI)

- 'Cria uma estrutura de relatório para um projeto com a HuskyLens relacionado com Machine Learning.'

- 'Explica o funcionamento do processador Kendryte K210.'

- 'Compara a aprendizagem supervisionada da HuskyLens com KNN.'

- 'Sugere planos de interação entre um robô Arduino e a HuskyLens para demonstração num evento.'

- 'Cria uma versão estendida do relatório destacando IA embarcada e futuras aplicações com Raspberry Pi.'