Otmização de recursos de hardware de Estrutura Robótica para Aquisição e Classificação de Imagens (ERACI) por meio da distribuição de processos em rede

José Ricardo Ferreira Cardoso[1], Jones Mendonça de Souza[1], Carlos Eduardo Angeli Furlani[2], José Eduardo Pitelli Turco[2], Eduardo Aparecido Roberti[1]

[1]Instituto Federal de São Paulo – Campus Barretos (IFSP)
Barretos, SP.

[2]Universidade Estadual Paulista (Unesp)
Jaboticabal, SP.

{jrfercar, jonessouza, eduardoroberti}@ifsp.edu.br,
{eduardo.furlani, jose.turco}@unesp.br

Abstract. *The application of Digital Image Processing techniques, as well as the use of systems that use Artificial Intelligence, has increasingly gained the attention of researchers who seek its application in the most diverse media. In order to develop a robotic system capable of creating a computer vision system for analyzing an image and, basically detecting the presence of an object of interest, the developed project unified knowledge about these two areas of computer science with the robotics, culminated in the development of a robotic structure with free and modular tools.*

Resumo. *A aplicação de técnicas de Processamento de Imagens Digitais, bem como a utilização de sistemas que utilizam a Inteligência Artificial, tem ganhado cada vez mais a atenção de pesquisadores que buscam a sua aplicação nos mais diversos meios. Com o objetivo de desenvolver um sistema robótico capaz de criar um sistema de visão computacional capaz analisar uma imagem e, detectar basicamente a presença de um objeto de interesse, o projeto desenvolvido unificou conhecimentos sobre estas duas áreas da ciência da computação com a área de robótica, culminou no desenvolvimento de uma estrutura robótica com ferramentas gratuitas e modulares.*

1. Introdução

O desenvolvimento de sistemas capazes de aprender e tomar decisões estão cada vez mais presente na sociedade, sejam eles dentro de computadores de uso pessoal, dispositivos móveis celulares, veículos ou até mesmo estruturas robóticas (LUGER, 2013). Estes dispositivos munidos destes softwares tem ampliado sua presença para diferentes meios com o propósito de melhorar a eficiência em processos produtivos na indústria e na agricultura, dentre outros.

Neste contexto, há diversos estudos científicos que utilizam técnicas de Processamento Imagens Digitais (PID) e Inteligência Artificial (IA) para a captura, segmentação, extração de características e a classificação de uma imagem

(GONZALEZ; WOODS, 2010; LUGER, 2013). Segundo (KANG; OH, 2018) a forma com que as imagens são obtidas merece atenção especial, podendo a escolha implicar em um alto desempenho do sistema e sacrifício da conveniência do usuário, enquanto, outra pode trazer grande conveniência, com sacrifício do desempenho do sistema.

A preocupação com o rápido desenvolvimento de novas tecnologias, eliminando tarefas baseadas em tecnologia já existentes no mercado, tem levado a difusão e utilização de software, framework e hardware opensource. Neste sentido, uma única placa de computador de baixo custo como o Raspberry Pi ® (RPI), que permite utilizar bibliotecas de processamento de imagens, como o Open Source Computer Vision – OpenCV que pode ser utilizado para a construção de dispositivos robóticos que permitem o processamento e o reconhecimento de padrões em imagens (OSROOSH; KHOT; PETERS, 2018).

Considerando que, a tecnologia da informação e a eletrônica são meios pelos quais se busca alcançar esta eficiência, a produção de um sistema de visão computacional é justificado, pelo que se espera em ganhos na eficiência e redução de custos por meio do reconhecimento de padrões nos diferentes espaços de trabalho.

Tudo isto leva ao questionamento de se é possível criar um sistema robótico capaz de permitir a aquisição, armazenamento, e o reconhecimento de padrões em imagens, por meio de software que utiliza visão computacional para analisar uma imagem e, detectar basicamente a presença de objetos de interesse em área rural ou urbana, utilizando software e hardware open source voltado, basicamente, para o uso escolar (RASPBERRY PI FOUNDATION, 2020) e, com característica modular.

Para responder a este questionamento, no presente trabalho é apresentado os resultados obtidos com o desenvolvimento da primeira parte do projeto que tem como objetivo geral a implementação de uma estrutura robótica capaz de detectar a presença de objetos de interesse. Com este fim foram estipulados os seguintes objetivos específicos: a) adquirir e armazenar imagens em um banco de dados; b) permitir a geração de conhecimento computacional por meio de algoritmos de visão computacional e gerir este conhecimento; e c) realizar reconhecimento de padrões em imagem em tempo real.

2. Materiais e Métodos

2.1. Descrição e Área

A captura das imagens, bem como os testes do ERACI (Estrutura Robótica para Aquisição e Classificação de Imagens), foram realizados no município de Barretos, localizada no estado de São Paulo. O município encontra-se a uma altitude de 544,0 m, Latitude: 20° 33’ 33” Sul, Longitude: 48° 34’ 8” Oeste.

Para adequar o sistema a diferentes tipos de algoritmos da área de visão computacional, foram capturadas imagens nas mais diversas áreas por meio do Dispositivo Robótico Remoto (DRR). Ao todo o DRR capturou imagens em áreas urbanas e rurais dentro de uma área de 171.728,00 m^2.

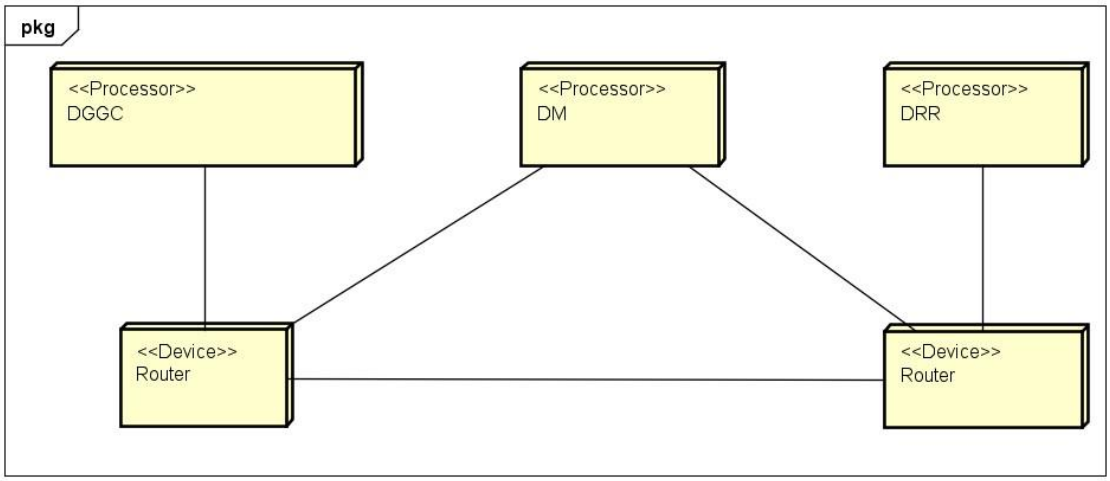

Figura 1 – Diagrama de instalação do sistema apresentando em cada um dos nós os dispositivos envolvidos em sua interação.

Fonte: Autoria Própria utilizando UML (SILVA & VIDEIRA, 2001)

2.2. Sistema

Com o desenvolvimento das redes de computadores e a sua consequente evolução, foi possível montar sistemas de computação compostos por grandes quantidades de computadores conectados por uma rede de alta velocidade chamados de sistemas distribuídos. Este tipo de sistema consiste em componentes autônomos, acessados por usuários que podem ser tanto pessoas quanto programas que se comportam como se estivessem lhe dando com um único sistema (TANENBAUM; STEEN, 2007). Para alcançar esta característica, o ERACI foi projetado foi projetado sob uma arquitetura de software e hardware modular, com suas partes separadas geograficamente e independentes, garantindo assim, basicamente a segurança, escalabilidade e transparência. Estas características permitem a troca de informações entre os dispositivos que o compõe, além de tornar possível a existência de vários especialistas conectados, por meio de Dispositivos Móveis (DM) com o Dispositivo Gerador e Gestor de Conhecimento (DGGC), para realizar a pré-classificação manual das imagens armazenadas no Banco de Dados (BD) (Figura 1).

No diagrama de instalação (Figura 1), o nó Dispositivos Robóticos Remotos (DRR) corresponde a estrutura que é levada ao local escolhido, para realizar a captura de imagens e a realização de testes. Ele está conectado a um nó chamado Router, que é um roteador TP-Link® modelo TL-MR 3420, capaz de permitir conexão com a Internet

por meio de outro roteador pela porta RJ-45 Wide Area Network (WAN) ou por meio de um modem 3G/4G conectado à sua porta Universal Serial Bus (USB). Quando o DRR está conectado ao roteador, é possível fazer uso de um dispositivo móvel conectado a este mesmo roteador, para realizar seu controle e monitoramento em tempo real por meio do aplicativo ERACI instalado nele.

Na outra ponta do diagrama (Figura 1), o nó Dispositivo Gerador e Gestor do Conhecimento (DGGC) tem softwares capazes de armazenar e gerir os dados de imagens, classes, usuário e classificações realizadas pelo usuário exercendo o papel de professor do subsistema de visão computacional. Além disso, O DGGC tem a capacidade de manter serviços HyperText Transfer Protocol (HTTP) e, gerar arquivos baseados em algoritmos de Machine Learning (ML), capazes de realizar o reconhecimento de padrões em imagens. O dispositivo representado por este nó também está conectado a um roteador TP-Link, porém o modelo é o WR-840N, que permite a conexão com a Internet apenas por meio de sua porta RJ-45 WAN.

Quando os dois dispositivos extremos, DRR e DGGC, estão em funcionamento e conectados aos seus respectivos roteadores e, estes por sua vez, estão conectados entre si por meio de rede cabeada ou pela Internet, eles podem trocar informações. Neste caso o DRR pode enviar as imagens capturadas ao DGGC que as processa, extrai as características e armazena as imagens e as características no BD. O DGGC envia os arquivos classificadores ao DRR para que este seja capaz de reconhecer os padrões nas imagens que estão sendo “vistas”.

O nó DM corresponde ao dispositivo móvel que tem o aplicativo ERACI instalado. Por meio deste aplicativo é possível gerir os dados armazenados no BD e realizar classificações das imagens nele armazenadas.

A forma como o sistema é organizado permite que possam existir várias instancias de DRR e DM e, a interação entre eles, ocorra por meio da rede de computadores, com a utilização dos protocolos HTTP, TCP/IP e UDP/IP. Dentro de uma arquitetura cliente/servidor, cada DRR e DM se comporta como um cliente conectado ao DGGC, que é capaz de receber inúmeras conexões de ambos os clientes. Assim, é possível que, quando em um ambiente de produção, possam existir vários DRR carregando imagens para um único servidor capaz de gerar modelos de ML cada vez melhores. Por conseguinte, é também possível que haja vários usuários classificando imagens a serem utilizadas pelos algoritmos de ML para a geração deste “conhecimento".

Cada um dos nós do diagrama (Figura 1) possui uma série de componentes de software, com as mais diversas funcionalidades incorporadas. Estas funcionalidades são divididas em subsistemas do DRR, subsistemas do DGGC e subsistemas do aplicativo para DM. Todos estes subsistemas foram implementados utilizando basicamente 3 plataformas, sendo elas: Java, Python e Android.

No DGGC é implementado a classe PDI para a extração de características da imagem e, assim, ser possível a classificação em uma etapa posterior. Esta classe é composta dos mais diversos métodos, necessários para a obtenção da imagem armazenada no BD e, sua conversão em RGB (Red, Green and Blue), conversão em HSV (Hue, Saturation and Value), tons de cinza e, diversos métodos para realizar o pré-processamento e segmentação das imagens, baseados na biblioteca OpenCV (OPENCV, 2020).

Em relação a extração das características das imagens, foi implementada uma estrutura de métodos que permitem a extração de características, consideradas fundamentais para o reconhecimento de padrões (BARELLI, 2018) como: Aspecto, Dimensional e Inercial. O primeiro método extrai a moda, média e desvio padrão referente a composição das cores de uma imagem. O segundo extrai características referente à dimensão da imagem segmentada por cor. O último extrai os 24 momentos da imagem, bem como os 7 momentos invariantes de HU (HU et al., 2012).

Com base nestas características, também foi criado um método para mostrar ao requisitante uma imagem assinalada. Para isso, primeiro é realizada a segmentação por cor, utilizando o espaço de cores HSV, baseando-se nos valores da moda, dentro de um limiar dimensionado pelo desvio padrão da imagem convertida em tons de cinza. Depois disso, é realizada a detecção de bordas por meio de Canny e, assim, por meio do centro de massa e da média dimensional obtida da imagem segmentada, é desenhado um círculo e inserido uma cruz no ponto que corresponde ao centro de massa da imagem. O tamanho do círculo e da cruz são proporcionais a área segmentada.

A geração dos classificadores acontece por meio do módulo de software responsável pela IA que, quando inicializado, realiza o carregamento dos dados cadastrados e pré-classificados pelos usuários. Depois disso é realizado o treinamento de diversos classificadores, baseados nos algoritmos: K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Logistic Regression (RL), Naïve Bayes (NB), Decision Tree (DT), Random Forest (RF) e Multilayer Perceptron (MLP) abstraídos pelo módulo scikitlearn (SCIKIT-LEARN, 2020).

Estes classificadores são gerados continuamente, sempre que ocorre um novo treinamento, programado para acontecer a cada 50 novas pré-classificações realizadas pelo(s) usuário(s). A função de previsão do módulo de IA, decide, dentre os diversos algoritmos, qual aquele que apresentou a melhor acurácia durante os testes realizados na etapa de treinamento. Primeiro, ele verifica qual o melhor classificador para prever a superclasse e, estima a superclasse que melhor representa a imagem. Depois, realiza o mesmo procedimento para verificar a subclasse desta imagem. Como resultado da função de previsão, é retornado uma tupla contendo a, superclasse, classe e um valor para acurácia do processo. Este valor de acurácia do processo é calculado por meio do valor médio resultante da soma dos valores de acurácia do algoritmo previsor da superclasse, subclasse e do resultado da função predict_proba() (SCIKIT-LEARN, 2020) do classificador para a subclasse estimada. Assim, cada um dos classificadores receberam como entrada os valores referentes aos campos apresentados na (Tabela 1) e como saída valores relativos aos agrupamentos característicos de área urbana (veículo, pessoas, edificações, árvore …), área rural (solo, agricultura, erva daninha, árvore, pessoa, veículo …) e especificamente, canavial (solo, erva daninha, cana-de-açúcar, veículo, pessoa …). Estes classificadores foram estruturados da seguinte forma:

K-Nearest Neighbors (KNN): Este classificador basicamente identifica um grupo de objetos K no conjunto de treinamento que estão mais próximos do objeto de teste e, atribui um rótulo baseado na classe mais dominante desta vizinhança.Para verificar o melhor valor para o atributo K, a função responsável pela geração do modelo analisa qual a melhor vizinhança que resulte na melhor acurácia do classificador (AWAD; KHANNA, 2015). Este valor foi estipulado para ser um dos números inteiros impares

dentro de um intervalo que vai de 1 até quantidade de classes possíveis para um novo objeto.

Suport Vector Machines (SVM): Esse algoritmo corresponde a métodos de aprendizagem supervisionada que analisam e reconhecem padrões. Seu funcionamento consiste em uma tarefa de aprendizagem de duas classes. Assim, ele constrói um modelo ou uma função de classificação que atribui novas observações a uma das duas classes de um hiperplano, o que faz dele um classificador linear binário não probabilístico. Um modelo SVM mapeia as observações como pontos no espaço, de modo que elas são classificadas em uma partição separada que é dividida pela maior distância até o ponto de dados de observação mais próximo de qualquer classe. Desta forma ele prevê que uma nova observação pertence a uma classe baseada em qual lado da partição eles caem. Por sua vez, os vetores de suporte são os pontos de dados mais próximos do hiperplano que divide as classes (AWAD; KHANNA, 2015).

Regressão Logística (RL): Este é um modelo probabilístico de classificação estatística que prevê a probabilidade da ocorrência de um evento. Ela modela a relação entre uma variável dependente categórica X e um desfecho categórico dicotômico ou característica Y (AWAD; KHANNA, 2015). Para o ERACI as classificações se dão pelo método um contra todos de forma cascateada.

Naive Bayes (NB): Ele é um classificador probabilístico que aplica a teoria de Bayes com uma forte suposição de independência, de tal forma, que a presença de uma característica individual não esteja relacionada a presença de outra (LUGER, 2013). Na prática este algoritmo analisa uma base de dados histórica e gera uma tabela de probabilidade. Esta tabela de probabilidade será consultada pelo algoritmo sempre que for necessário a classificação de um novo objeto (AWAD; KHANNA, 2015).

Decision Tree (DT): é um processo passo a passo para decidir a categoria que algo pertence. Para resolver um problema deste tipo poderia ser utilizado um fluxograma de perguntas em que a resposta fosse dada de forma encadeada até que se chegue a uma determinada categoria, a qual o objeto provavelmente pertença (HARTSHORN, 2016). No ERACI, o classificador foi parametrizado para utilizar a entropia para estabelecer os nós de cada uma das árvores.

Random Forest (RF): é uma abordagem de aprendizagem em conjunto para realizar classificações. Basicamente ela faz uso de uma coleção de árvores de decisão (DT) correlacionadas para isto. Assim, em vez de trabalhar apenas com uma única solução baseada na saída de uma árvore profunda, a floresta randômica agrega a produção de um número de árvores rasas, formando uma camada adicional para agrupamento (HARTSHORN, 2016). O ERACI realiza a verificação de acurácia de diferentes quantidades de árvores, podendo estas variar entre 1 e 40, além de utilizar a função de entropia para estabelecer os nós de cada uma das árvores.

Multilayer Perceptron (MLP): Este é um algoritmo baseado em uma rede de neurônios simples, que mapeia conjuntos de dados de entrada em um conjunto de saídas. Um MLP compreende várias camadas de nós totalmente conectados por um grafo direcionado, no qual cada nó, com exceção do nó de entrada, é um neurônio com uma função de ativação não linear. O componente fundamental de um MLP é o neurônio. Assim, em um MLP, um par de neurônios é conectado em duas camadas adjacentes, usando bordas ponderadas (LUGER, 2013). A melhor configuração para a MLP a ser utilizada pelo ERACI foi conseguida por meio da utilização da metodologia

de tentativa e erro. Ela mostrou que para se chegar ao melhor resultado classificatório era conveniente utilizar 1000 iterações, com taxa de aprendizagem de 0,000010 e 4 camadas ocultas com 400 neurônios cada uma. Os outros parâmetros da rede foram mantidos com seus valores padrão (SCIKIT-LEARN, 2020).

Ao final do treinamento de cada um dos classificadores, cada um deles é armazenado em arquivo para ser utilizado posteriormente pelo próprio dispositivo ou ser transferido por meio da web para um DRR.

Com o propósito de testar estes classificadores, foi criado um algoritmo que faz uso da validação cruzada utilizando StratifiedKFold() do (SCIKIT-LEARN, 2020) com seu parâmetro n_splits definido para ser igual a quantidade de classes existentes durante o momento. Os parametros shuffle e randomstate foram definidos como True e 0, respectivamente.

Para verificar se os algoritmos estão realmente aprendendo à medida que os dados préclassificados são inseridos, foi criado um processo no módulo de IA que injeta no classificador, de forma cíclica e incremental, 50 dados por vez e, extrai valores referentes a acurácia sob esta determinada quantidade de dados. Esta quantidade foi estipulada ao acaso, e visa permitir a geração de dados que descrevem a evolução, ou não, da acurácia dos algoritmos.

Tabela 1. Campos e respectivos tipos de dados utilizados para o treinamento e teste dos algoritmos.

Número	Campo	Tipo	Variação
1	Modavermelho	Inteiro	0 – 255
2	Modaverde	Inteiro	0 – 255
3	Modaazul	Inteiro	0 – 255
4	Mediavermelho	Float	0 – 255
5	Mediaverde	Float	0 – 255
6	Mediaazul	Float	0 – 255
7	Mediacinza	Float	0 – 255
8	Desviopadraovermelho	Float	0 – 255
9	Desviopadraoverde	Float	0 – 255
10	Desviopadraoazul	Float	0 – 255
11	Desviopadraocinza	Float	0 – 255
12	hu1	Float	∞
13	hu2	Float	∞
14	hu3	Float	∞
15	hu4	Float	∞

16	hu5	Float	∞
17	hu6	Float	∞
18	hu7	Float	∞

3. Resultados e Discussão

3.1. Hardware do DRR

O primeiro item desenvolvido foi o DRR por meio da configuração do Raspberry Pi. Para que o hardware não fique exposto e vulnerável a qualquer avaria no que tange, principalmente, à desconexão acidental de cabos, foi confeccionado uma estrutura plástica para envolve-la e deixar acessível (Figura 2a e Figura 2b). : os botões, a estrutura pan-tilt, câmera, refletor infravermelho, power Bank, terminal HDMI e, os terminais para alimentação do RPI e para carregamento da bateria selada de 12 V. Além disso, foram criados orifícios para permitir o acesso ao cartão de memória, terminais RJ45 e USB. Estes orifícios são fechados com placas plásticas parafusadas na carenagem do robô.

Para realizar a aqusição das imagens nas localidades o DRR foi acoplado a um veículo automotor, de forma que ele fique prensado na porta de seu lado esquerdo (lado do motorista) em sua janela lateral. Assim, as partes que correspondem ao refletor infravermelho e a câmera ficam do lado de fora do veículo e o restante, como botões e o LED (Light-Emitting Diode) indicador de que ele está vendo, permaneçam no lado interno do veículo (Figura 2c). A distância, entre o solo, em local plano, e a câmera do dispositivo permaneceu a aproximadamente 1,35 m, com a variação de até 0,15 m podendo ocorrer devido às irregularidades existente no terreno no ato da captura da imagem.

Figura 2 – Robô com a estrutura plástica para proteção: a e b) superfícies laterais da carcaça; c) acoplamento do dispositivo ao veículo automotor.

Fonte: Autoria Própria

3.2. Hardware do DGGC

O DGGC foi implementado em um computador Raspberry Pi modelo 3 B+ e, é configurado para armazenar os dados necessários à geração do conhecimento e aqueles necessários à gestão desse conhecimento. Para que seja assegurada a execução do sistema sem interrupção por falta de energia, ele utiliza um módulo HAT, capaz de gerir o fornecimento energético e garantir uma autonomia de funcionamento de até 3 h, sem alimentação pela tomada de energia elétrica, utilizando bateria de lítio com a especificação: 3,7 V; 3800 mAh. Esta estratégia foi utilizada para mitigar a possibilidade de corrupção dos dados armazenados no cartão de memória do RPI. Com o objetivo de auxiliar na redução da temperatura em ambos os dispositivos (RPI e HAT), foi conectado um cooler aos pinos 5 V e Ground do RPI.

3.3. Software do DRR

Dentre os softwares desenvolvidos, encontra-se o Subsistema de Inicialização do Robô, que é executado como um serviço do Sistema Operacional (SO) Raspbian. Basicamente ele é formado por apenas um script Python e, é responsável por carregar os outros subsistemas necessários para o funcionamento do dispositivo. Ele faz uso de um script para configuração que tem sua referência passada na forma de objeto para os subsistemas de arquivamento, monitoramento e de comunicação e, armazena e disponibiliza dados como: IP ou URL do servidor, portas UDP e TCP que o DGGC está "ouvindo", portas UDP e TCP que o DRR "escuta", senha de segurança para validação de conexão em rede com clientes e servidores, intervalo de tempo em que o dispositivo capturará e armazenara as imagens visualizadas e o threshould utilizado para considerar que ocorreu mudança em uma imagem visualizada pelo robô.

O Subsistema de Controle possui métodos para desempacotamento de instruções e a execução de ações diretamente nos módulos pan-tilt e ponte-H. Ele também é responsável por executar as tarefas de desligamento e reinicialização do sistema e, em conjunto com o subsistema de visão computacional, acionar a captura de imagens e vídeos. Seu funcionamento é baseado na troca de mensagens codificadas, que possibilitam a movimentação do robô, capturar imagens e movimentar a estrutura pan/tilt.

O Subsistema de Monitoramento, quando é iniciado, verifica em qual situação ocorreu o último desligamento por meio da análise de arquivo interno que armazena dados como: tempo que o sistema esteve em funcionamento, o tempo em que esteve funcionando sob alerta de baixa tensão, a forma que se deu o desligamento (abrupto ou não), o tempo máximo que ele funcionou sem interrupção abrupta e a data, hora e minuto que se deu o último armazenamento de dados. Essas informações são utilizadas para a inicialização das variáveis necessárias para a proteção do sistema, no que se refere ao tempo de disponibilidade da bateria. A cada 60 s, são executados os métodos destinados a detecção de alerta de baixa tensão e verificação do tempo de disponibilidade de carga da bateria. Ambos os métodos atualizam o arquivo para monitoramento da bateria e verificam se a quantidade de tempo, com e sem alertas de baixa tensão, são indícios de que o sistema deve ser desligado ou reinicializado para sua autoproteção. Outra tarefa realizada após a inicialização deste subsistema, é a verificação periódica do uso da CPU (Central Process Unit) e, obtenção dos dados de latitude e longitude em que o dispositivo se encontra. Para ser possível realizar todas

estas tarefas, este subsistema utiliza a biblioteca psutil e o módulo de hardware receptor de GPS modelo NEO-6M GPS, respectivamente.

O Subsistema de Arquivamento verifica periodicamente, em intervalos de 60 s, se há novos arquivos armazenados nos diretórios de imagem e de vídeo. Em caso afirmativo, ele tenta realizar uma conexão com o subsistema de serviço web no DGGC, e assim que a conexão for efetivada, envia estes arquivos para ele. Depois disso, remove os arquivos do DRR. Após finalizar o envio dos dados, no formato json, por meio do protocolo HTTP para o serviço que é disponibilizado no DGGC, ele entra no modo de espera, até que novos arquivos sejam detectados.

O Subsistema de Comunicação, após iniciado, entra em um laço de repetição que verifica se existe uma conexão disponível com o servidor. Quando a conexão com o servidor ocorre, este subsistema envia uma senha local para o servidor e aguarda o recebimento da senha do servidor para validação. Quando a senha proveniente do servidor é validada, são criados objetos responsáveis pelo seu monitoramento, multiplexação dos comandos recebidos do subsistema supervisório e, envio de imagens em tempo real utilizando recursos de computação concorrente por meio de threads. Quando a senha recebida pelo robô não é validada, a conexão é desfeita e são realizadas novas tentativas de conexão até que os requisitos para validação da conexão sejam alcançados. Além disso, ele também tem o comportamento de servidor TCP/IP e UDP/IP, que esperam por uma conexão de clientes por meio de sockets. Ele basicamente aguarda por uma conexão proveniente de um cliente e, ao receber esta solicitação, recebe um valor de chave a ser validada pela paridade com a chave armazenada no módulo de configuração. Após esta validação ocorre a transmissão das imagens e dos dados de monitoramento para os dispositivos solicitantes, bem como, a habilitação do controle do DRR por parte destes, bastando para isso que eles enviem os comandos codificados no formato exigido pelo sistema de controle.

3.4. Softwares do DGGC

O Subsistema de Inicialização do DGGC é composto basicamente por um script python responsável pela inicialização dos outros subsistemas. Ele é executado como serviço do SO e é estruturado dentro de um computador RPI dedicado para o armazenamento dos dados de imagens, características, usuários, classes e classificações por meio do Sistema Gerenciador de Banco de Dados (SGBD), bem como para a execução dos subsistemas de visão computacional, gerador de IA e de serviços web. Sua estrutura é simples, consistindo apenas na inicialização do Subsistema de Visão Computacional e de Serviço web.

Após a execução do subsistema de inicialização do DGGC, o Subsistema de Serviço web é iniciado com a instanciação do serviço. Este por sua vez disponibiliza serviços pelos métodos HTTP: GET, POST, PUT e DELETE. Um aspecto relevante a ser mencionado aqui, é que todas as requisições exigem a autenticação com senha e, ao final da execução dos métodos é retornado para o requisitante uma lista no formato json com os registros do Banco de Dados requisitados.

O subsistema de Gerenciamento do Banco de Dados é iniciado como um serviço do SO. Assim o SGBD PostgreSQL® começa a “escutar"pela porta 5432. Junto com ele também é iniciado um módulo que disponibiliza os serviços necessários para a gestão

de usuários, classes, classificações e imagens pelo subsistema mobile por meio da porta 5005. Para isso ele utiliza um módulo Data Access Object – DAO. Este módulo DAO utiliza o sqlalquemy para realizar o Object Relational Mapping – (ORM) que é basicamente a conversão de registro de banco de dados em objeto de um sistema orientado a objetos e vice e versa (ELMASRI; NAVATE, 2018; SIAHAAN, 2019; SIANIPAR, 2019; SQLALCHEMY, 2020).

Foi implementado também, neste dispositivo, um Subsistema Supervisório Desktop para permitir que o usuário tenha acesso ao DRR e, assim, acompanhe as imagens provenientes de sua câmera, os dados de uso da CPU, as coordenadas de posicionamento global, além de possibilitar o movimento do DRR e sua pan/tilt. A interação do usuário com este subsistema acontece por intermédio de uma interface visual que permite, em tempo real, acompanhar as imagens transmitidas pelo robô, visualizar as suas coordenadas geográficas, saber o nível de utilização da CPU, movimentar o robô, posicionar a câmera, capturar e armazenar as imagens nas estações de trabalho onde o supervisório está instalado ou no próprio robô. É importante salientar que todas as tarefas relacionadas ao controle do robô, também podem ser realizadas por intermédio de um Joystick conectado pela USB ou por Bluetooth.

3.5. Software do DM

Também foi desenvolvido um aplicativo para permitir a gestão de dados, pré-classificação de imagens e a supervisão do DRR. Este aplicativo foi nomeado como ERACI. Para minimizar os riscos referente a utilização indevida dos dados de imagens, foram criados módulo para criação e gestão de usuários e classes. Este aplicativo recebe uma imagem assinalada do subsistema de serviços web e, a partir daí, permite que o usuário exclua a imagem ou realize a classificação da mesma.

O Subsistema Superviório Local funciona de forma independente do subsistema de gestão de dados, embora possa ser acessado pelo mesmo aplicativo. Ele foi confeccionado para permitir que o usuário possa visualizar o que o DRR está “vendo” e como está classificando. Ela também disponibiliza botões na tela para que o usuário possa posicionar a câmera, conforme a necessidade do momento.

3.6. Aquisição e a Pré-classificação das Imagens

As imagens são as fontes para a geração de modelos de ML utilizados para realizar o reconhecimento de padrões e classificação das imagens. Assim, a qualidade deste modelo está diretamente relacionada a quantidade e variabilidade destas imagens (OPENCV, 2020), que devem conter o objeto a ser classificado, chamados aqui de imagem positiva e também imagens que não contém o objeto a ser classificado, chamado aqui de imagem negativa. Assim, entre o período de 4 de setembro a 24 de janeiro de 2020 foram adquiridos um total de 36.979 imagens positivas e negativas.

Para este projeto foram testados vários métodos e sobreposição de métodos. Contudo, concluiu-se que para a realização dos testes com o ERACI deveria ser utilizado uma forma que abrangesse pelo menos o histograma da imagem por meio da média, moda e desvio padrão de cada uma das camadas RGB e, da imagem em tons de cinza. Estes dados fornecem informações da imagem como um todo e não apenas dos

objetos de interesse na imagem. Por esta razão também foi pensado em uma forma que simule digitalmente o processo em que o ser humano foca em um objeto específico da imagem. Para isso foi utilizado a segmentação por cor utilizando o esquema HSV.

Baseado na hipótese de que quando se está focado em um objeto da imagem, as cores que compõe esta parte da imagem tendem a prevalecer em relação às outras. Para o ERACI este foco pode ser obtido por meio da distância entre o objeto e o observador e, por meio do ângulo de visão deste observador em relação ao objeto observado. Para se chegar a esta forma de obtenção de características da imagem, foram feitos alguns testes que levaram em conta a relevância das características para a realização dos testes de classificação sobre o ERACI e, o tempo gasto entre a extração das características e a classificação. Levando em conta que a sobreposição de vários métodos para pré-processamento exige mais trabalho da CPU, optou-se por utilizar a sobreposição de apenas dois métodos para a segmentação das imagens, ficando assim a sobreposição dos métodos para uniformização da imagem e para a detecção de bordas utilizando Canny. Após a segmentação da imagem por cor foi possível obter os 7 momentos invariantes de HU (HU et al., 2012).

A pré-classificação das imagem, acontece pelo usuário conectado ao ERACI por meio do aplicativo eraci devidamente instalado no DM. Este aplicativo, quando conectado ao DGGC realiza uma chamada ao subsistema de serviço web que envia ao requisitante uma imagem assinalada, obtida de forma aleatória. Baseado nisso, foi possível classificar 1.579 imagens entre os dias 20 de dezembro de 2019 e 25 de fevereiro de 2020.

3.7. Geração de Inteligência Artificial

Foram estabelecidas por meio do aplicativo eraci as possíveis classes em que uma imagem poderia pertencer. As classes foram agrupadas em 3 grupos denominados superclasses: Área Rural e Canavial possuem uma gama de 8 classes possíveis e, área urbana comporta 7 classes.

De forma geral, se utilizássemos para realizar a classificação das imagens uma metodologia baseada na escolha ao acaso, pode-se dizer que teríamos de (100/7 ~ 14) a (100/8 ~ 12%) de chances de realizar uma classificação correta. Com base nestes valores, concluiu-se que para se certificar que o algoritmo está reconhecendo padrões e, tomando decisão baseado nestas, ele deverá ter, no mínimo uma acurácia maior que 14% para área urbana e maior que 12% para área rural e canavial.

Com o propósito de realizar estes testes, foi criado um algoritmo que faz uso da validação cruzada utilizando StratifiedKFold() do (SCIKIT-LEARN, 2020) com seu parâmetro n_splits definido como 10, sobre as imagens previamente classificadas no BD.

Para verificar se os algoritmos estão realmente aprendendo à medida que os dados préclassificados são inseridos, foi criado um processo no módulo de IA que injeta no classificador, de forma cíclica e incremental, 50 dados por vez e, extrai valores referentes a acurácia sob esta determinada quantidade de dados. Esta quantidade foi estipulada ao acaso, e visa permitir a geração de dados que descrevem a evolução, ou não, da acurácia dos algoritmos. Com os dados gerados por este processo foi possível

gerar uma série de gráficos que mostram a relação existente entre a quantidade de dados pré-classificados e a acurácia dos algoritmos classificadores.

Após a análise dos dados resultantes da estimação da acurácia, em termos de quantidade de dados utilizados para o treinamento dos classificadores, verificou-se que no início, quando existiam poucos dados para treinamento, a acurácia dos classificadores pareciam melhores. Porém, com um olhar mais criterioso sobre esta informação, foi observado que este fato ocorre devido à pouca variabilidade de itens categóricos. Conforme esta quantidade vai aumentando, maior é a dificuldade em se classificar, resultando assim em uma acurácia menor entregue pelos algoritmos. Esta análise sobre os gráficos, também mostrou que todos eles apresentam um crescimento constante no valor da acurácia, a partir de determinada quantidade de dados, o que nos leva a acreditar que, ambos os algoritmos, estão sujeitos a um limiar, onde, a partir daí, o algoritmo se estabiliza e se mantem “aprendendo"e melhorando o seu desempenho. Por esta razão, foi dada maior importância em se descobrir com que quantidade de dados pré-classificados os algoritmos começam a apresentar boa acurácia de forma crescente e estável. Com este propósito calculou-se o mínimo global da função de acurácia projetado sobre a função de quantidade. A ele foi dado o nome de Quantidade Mínima para Regularidade (QMR), que pode ser observado nas Tabelas .

Como observado nas (Figuras 3 e 4) o QMR se dá conjuntamente com a menor acurácia entregue pelo respectivo algoritmo ou mínimo global e, se mantem em crescimento constante, podendo ocorrer oscilações, em que os mínimos de cada período, não são menores que o QMR. Quando o QMR é analisado em termos da classificação de superclasses, nota-se que o classificador MLP atinge este ponto com 1.250 dados (Figura 5). Desta forma o algoritmo entrega uma acurácia aproximada de 74% com um desvio padrão de aproximadamente 0,084.

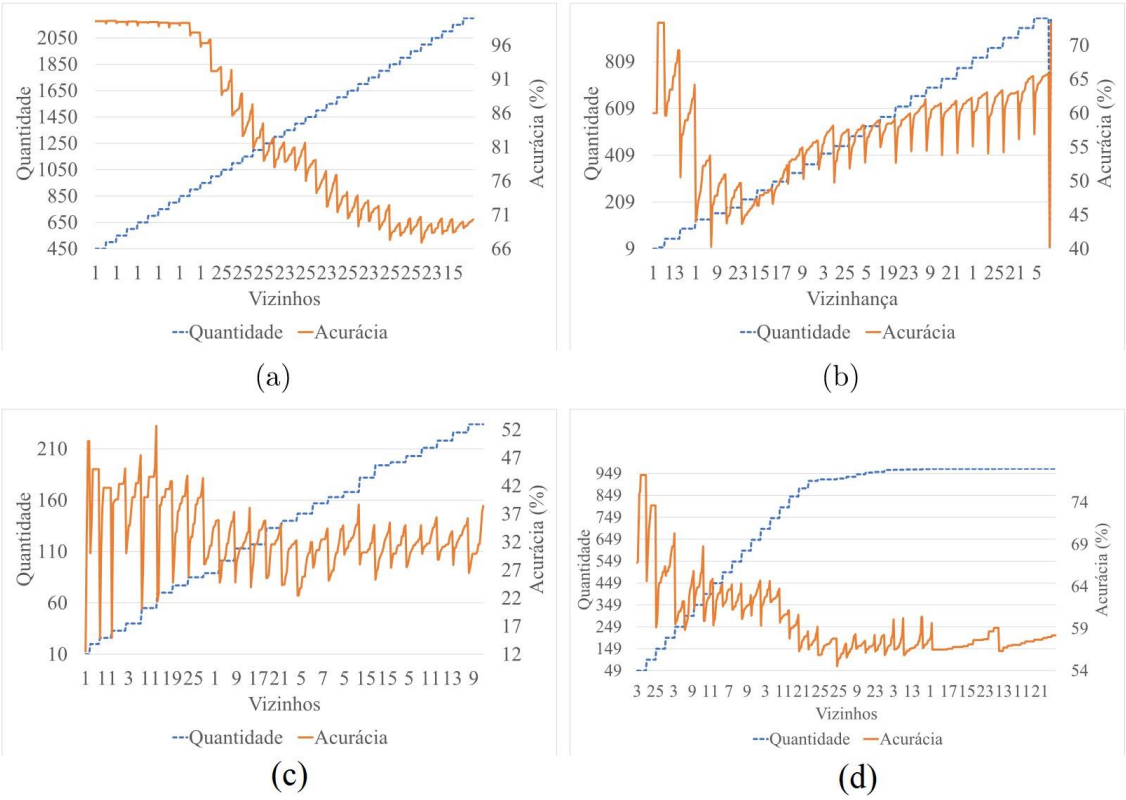

Figura 3 – Relação existente entre quantidade de dados e acurácia do algoritmo KNN. (a) Superclasse; (b) Classe Canavial; (c) Área Rural; (d) Área Urbana.

Tabela 2 – QMR relativo aos classificadores para a classificação de superclasses.

Classificador	QMR	Acurácia	Desvio Padrão
MLP	1250	74,0803511	0,084252265
SVM	1600	62,99735556	0,036370276

Tabela 3 – QMR relativo aos classificadores para a classificação da subclasse canavial.

Classificador	QMR	Acurácia	Desvio Padrão
MLP	185	42,6344086	0,108922396
SVM	185	42,16845878	0,067352286

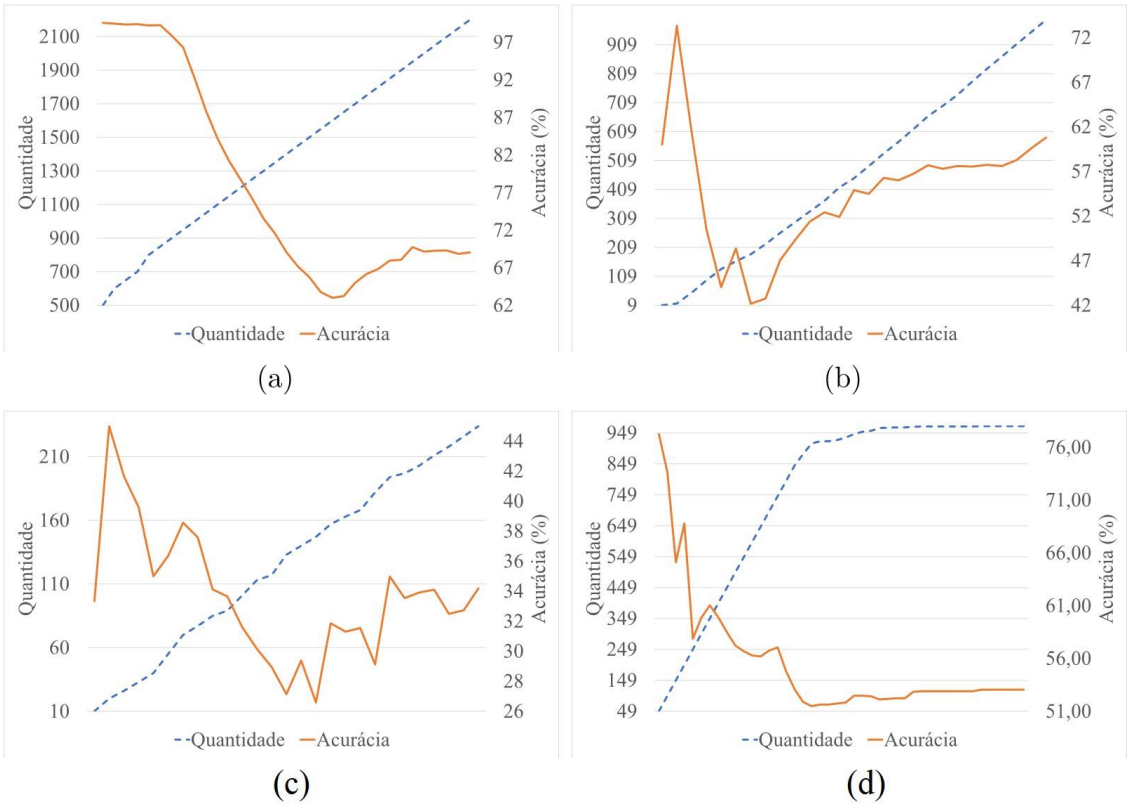

Figura 4 – Relação existente entre quantidade de dados e acurácia do algoritmo SVM. (a) Superclasse; (b) Classe Canavial; (c) Área Rural; (d) Área Urbana.

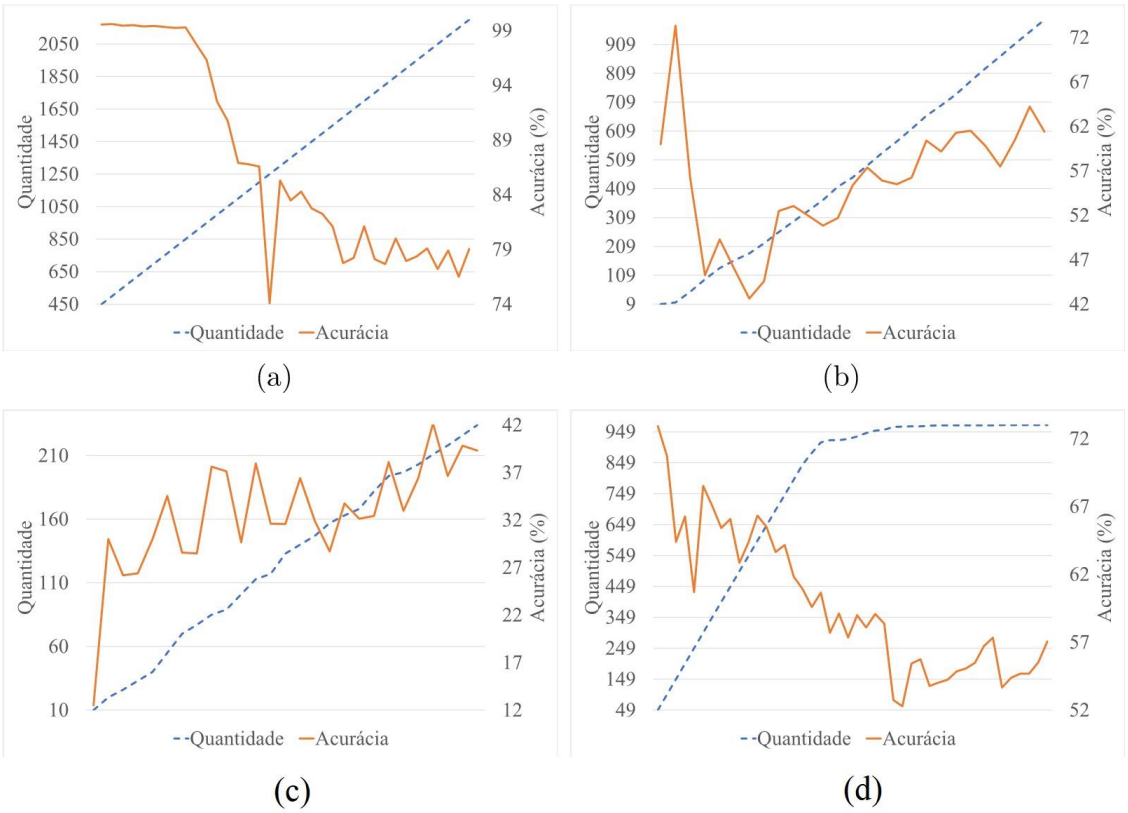

Figura 5 – Relação existente entre quantidade de dados e acurácia do algoritmo MLP. (a) Superclasse; (b) Classe Canavial; (c) Área Rural; (d) Área Urbana.

Uma das razões a serem levantadas aqui sobre a oscilação da acurácia, durante todo o processo de injeção de dados para o treinamento, é o fato de ocorrer desbalanceamento em relação a quantidade de categorias injetadas em cada processo ((AWAD; KHANNA, 2015)). Este fato propicia que os algoritmos se tornem muito bons em reconhecer padrões referentes a uma determinada categoria e, ruim em outras que tenham menos amostras.

Por meio da análise de todos estes dados, é possível dizer que os métodos utilizados para realizar a classificação das imagens tem uma tendência a ter uma melhor acurácia, a partir do ponto QMR. Assim, a acurácia entregue pelos algoritmos tendem a aumentar proporcionalmente ao aumentando da quantidade de registros pré-classificados.

Devido a variação da acurácia entregue pelo algoritmo, pode-se dizer que a estratégia de se verificar em tempo de execução, qual é o algoritmo que entrega a melhor acurácia para a classificação é válida, pois, desta forma, é utilizado sempre o classificador que melhor se adequa a quantidade de dados pré-classificados utilizados para treinamento.

3.8. Tempo de Processamento

Para analisar o tempo gasto pelo DRR para realizar todo os processos, até o reconhecimento do padrão na imagem em tempo real, foi embutida no subsistema de

visão computacional um método. Com os dados resultantes foi gerado um gráfico que permite uma analise destes valores (Figura 6).

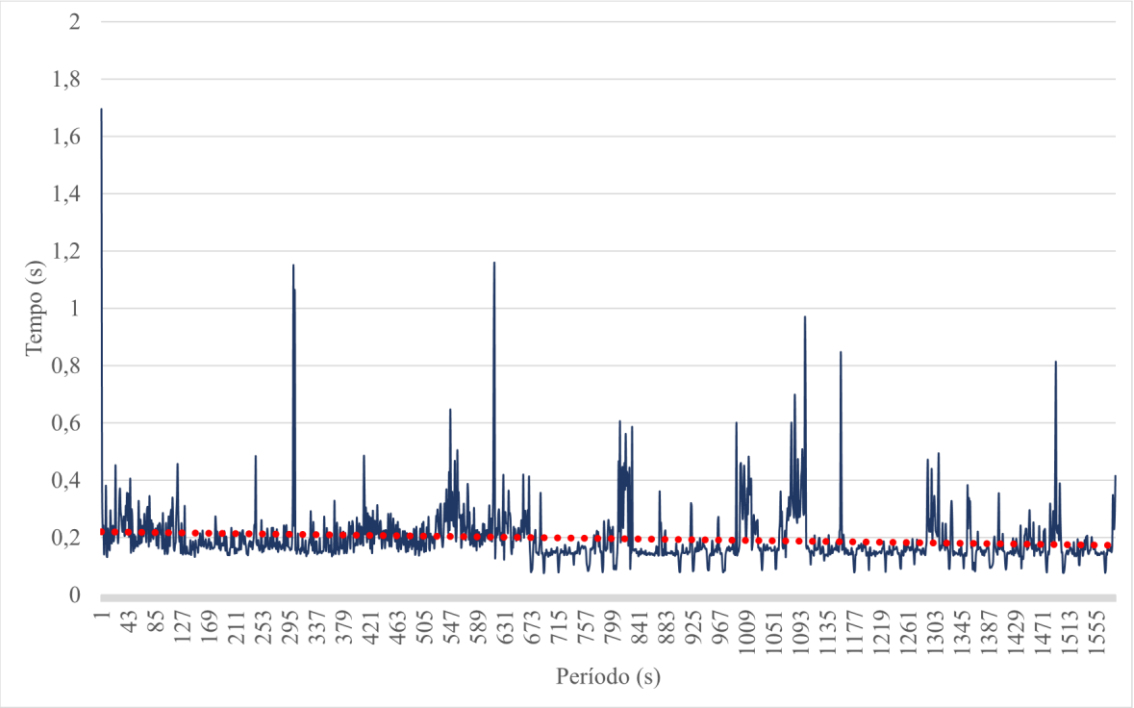

Figura 6 – Tempo de processamento necessário para realizar todos os procedimentos para extração de características e reconhecimento de padrões na imagem capturada em tempo real pelo DRR em um período de 1.585 s.

Uma análise sobre os dados que serviram de base para a geração do gráfico (Figura 6) mostrou que o tempo gasto para o processamento dos dados, desde a captura da imagem até o reconhecimento dos padrões na imagem, tem variação de aproximadamente 0,075651407 e 1,69562912 s, o que dá ao dispositivo um tempo médio para processamento de 0,196309677 s. O desvio padrão resultante destes dados é da conta de 0,099244419 s.

A linha de tendência do gráfico, apresentado pela linha tracejada em vermelho, mostra uma tendência em manter este tempo perto do valor mínimo, na grande maioria das vezes, o que assegura que o sistema pode ser utilizado para realizar classificações em tempo real quando a velocidade do veículo condutor do DRR é controlada pelo próprio DRR, que deverá permitir que a classificação de uma nova imagem seja realizada somente após a finalização de uma classificação anterior, e assim sucessivamente.

4. Conclusão

O ERACI foi confeccionado com recursos Open Source, com softwares disponibilizados gratuitamente e hardwares baseado em computador de baixo custo voltado primordialmente para o ensino de informática, algo bem vindo quando se trata em reproduzí-lo por empresas recém formadas.

Seu desenvolvimento mostrou que o computador Rapberry Pi pode ter seu uso expandido para o desenvolvimento de projetos mais robustos, servindo, não apenas como uma placa de prototipação mas também de unidade de processamento de sistemas

robóticos voltados para setores produtivos, sendo assim, uma alternativa computacional de baixo custo para este fim. Sendo assim, o projeto em si conseguiu realizar a aquisição e armazenamento das imagens capturadas em um banco de dados de forma automatizada e inteligente, pois, é capaz de detectar quando existe uma possível conexão entre o DRR que captura a imagem e o DGGC que a armazena em Banco de Dados.

Outro aspecto desejado para o ERACI foi que ele tivesse a capacidade de gerar conhecimento por meio de algoritmos de PDI e IA, além de ser capaz de gerir este conhecimento. Estes objetivos foram atingidos por meio do módulo DGGC, que realiza de forma cíclica e automática a extração das características e a geração de conhecimento. Além disso, ele disponibiliza serviços web para a gestão e compartilhamento do conhecimento gerado.

O reconhecimento de padrões em imagens foi alcançado por meio do DRR que é capaz de capturar imagens, enviar as imagens para o DGGC, obter os classificadores por meio de acesso a serviço publicado pelo DGGC e em seguida, realizar o reconhecimento dos padrões nas imagens em tempo real, com uma variação aceitável, porém, ainda não ideal. Além disso, a capacidade de realizar a escolha de qual classificador utilizar em tempo de execução se mostrou bem vinda, pois ela leva em conta que, um classificador pode apresentar melhor acurácia que outro diante da quantidade de amostras e sua variabilidade, existentes para treinamento em determinado momento.

Por fim, é possível concluir que o ERACI atingiu os objetivos propostos inicialmente e, que este sistema pode ser melhorado por meio da aplicação de algoritmos mais robustos para o reconhecimento de padrões e, também por meio de implementação de uma estrutura robótica capaz de possibilitar a locomoção, controlada por ele próprio, da estrutura no campo. Estas características desejadas, podem ser implementadas em projetos futuros.

Referências

AWAD, M.; KHANNA, R. Efficient Learning Machines: Theories, concepts, and applications for engineers and system designers. Berkeley, CA, USA: Open Access, 2015.

BARELLI, F. Introdução à Visão Computacional: Uma abordagem prática com Python e OpenCV. Casa do Código, 2018. ISBN 9788594188588. Disponível em: <https: //books.google.com.br/books?id=CA5ZDwAAQBAJ>.

ELMASRI, R.; NAVATE, S. B. Sistemas de Banco de Dados. 7. ed. São Paulo, SP, Brasil: Pearson Education do Brasil, 2018.

GONZALEZ, R. C.; WOODS, R. E. Processamento Digital de Imagens. 3. ed. São Paulo, SP, Brasil: Pearson Education do Brasil, 2010.

HARTSHORN, S. Machine Learning with Random Forests and Decision Trees: A visual guide for beginners. [S.l.]: Fairly Nerdy, 2016.

HU, R. et al. Multiscale distance matrix for fast plant leaf recognition. IEEE Transactions on Image Processing, v. 21, n. 11, p. 4667–4672, 2012.

KANG, E.; OH, I. Weak constraint leaf image recognition based on convolutional neural network. In: 2018 International Conference on Electronics, Information, and Communication (ICEIC), 2018. p. 1–4.

LUGER, G. Inteligência Artificial. 6. ed. São Paulo, SP, Brasil: Pearson Education do Brasil, 2013

OPENCV. OpenCV. 2020. Acessado em: 2020-04-09. Disponível em: <https://opencv.org/>.

OSROOSH, Y.; KHOT, L. R.; PETERS, R. T. Economical thermal-rgb imaging system for monitoring agricultural crops. Computers and Electronics in Agriculture, v. 147, p. 34 – 43, 2018. ISSN 0168-1699. Disponível em: <http://www.sciencedirect.com/science/article/pii/ S0168169917310360.

RASPBERRY PI FOUNDATION. Raspberry Pi Documentation. 2020. Acessado em: 2020-04-09. Disponível em: <https://www.raspberrypi.org/documentation/>.

SCIKIT-LEARN. Machine Learning in Python. 2020. Acessado em: 2020-04-09. Disponível em: <https://scikit-learn.org/stable/>.

SIAHAAN, V.; SIANIPAR, R. OpenCV-Python with PostgreSQL for Absolute Beginners: A Hands-On, Practical Database-Driven Applications. SPARTA PUBLISHING, 2019. Disponível em: <https://books.google.com.br/books?id=6o6vDwAAQBAJ>.

SILVA, A. da; VIDEIRA, C. UML, Metodologias e Ferramentas CASE: Linguagem de Modelação UML, Metodologias e Ferramentas CASE na concepção e desenvolvimento de sistemas de informação. ISBN 978-989-615-061-7. Lisboa, Porto: Centro Atlântico, 2008.

SQLALCHEMY. The Python SQL Toolkit and Object Relational Mapper. 2020. Acessado em: 2020-04-09. Disponível em: <https://www.sqlalchemy.org/>.

TANENBAUM, A. S.; STEEN, M. van. Sistemas distribuídos: princípios e paradigmas. São Paulo, SP, Brasil: Pearson Prentice Hall, 2007.