Projeto: Carrinho com Visão Computacional e Aprendizagem de Máquina

Disciplina: PSI3422 - Laboratório de Sistemas Eletrônicos

Nome: Igor Costa Doliveira 11391446
Lucas Kairuz Martins 11805377

1. INTRODUÇÃO

No segundo experimento, será desenvolvido um carrinho elétrico controlado remotamente por uma placa embarcada. A placa embarcada estará conectada a um computador via Wi-Fi, adquirirá imagens de uma câmera e irá compactá-las e transmitirá para o computador. O computador, usando processamento de imagens e aprendizagem de máquina, reconhecerá as imagens e controlará o movimento do carrinho de acordo com as instruções, será utilizado uma máquina de estados finita para o controle de todas as opções de movimentação do carrinho.

Este projeto aborda os seguintes aspectos de eletrônica e sistemas computacionais:

- Comunicação via Wifi: O carrinho se comunica com o computador por meio de uma rede Wi-Fi.
- Placa embarcada Raspberry: A placa embarcada é um dispositivo que combina hardware e software para executar tarefas específicas.
- Processamento de imagens: O computador usa processamento de imagens para reconhecer as imagens enviadas pela placa embarcada.
- Aprendizagem de máquina: O computador pode usar aprendizagem de máquina para melhorar o reconhecimento de imagens.
- Processamento paralelo: O computador pode usar processamento paralelo para acelerar o reconhecimento de imagens.
- Máquina de estados finita: O computador pode usar uma máquina de estados finita para controlar o movimento do carrinho.
- Programação orientada a objetos: O software do computador pode ser desenvolvido usando programação orientada a objetos.

2. ESPECIFICAÇÕES

Para o desenvolvimento do projeto foi criada a arquitetura servidor-cliente TCP/IP via roteador wifi, o raspberry funcionará como o servidor e será criado o programa servidor6.cpp nele com funções básicas de um servidor, como trocar informações com os cliente, entre eles comandos em forma de strings e imagens da

câmera. Já o computador será o cliente e será criado o programa cliente6.cpp que receberá os comandos do servidor e mandará avisos.

Já para que os dispositivos se comuniquem será necessário conectar ambos no mesmo wifi e reconhecer o endereço IP do Raspberry. Foi utilizado uma máquina virtual no nosso notebook pessoal no qual instalamos a versão Linux da disciplina com todas as bibliotecas necessárias. Para fazer upload e editar arquivos nos dispositivos foi utilizado o FTP do **Filezilla**, já que utilizamos este aplicativo em outras disciplinas, para realizar os comandos manuais do carrinho foi utilizado o **SSH** no próprio prompt de comando.

Foi criado o programa ***projeto.hpp*** no qual colocamos algumas funções em comum do servidor e do cliente. Nele utilizamos a linguagem de programação C++ com orientação de objetos, foi criada a classe mãe DEVICE e as classes filhas SERVER e CLIENT, na classe mãe tem as funções básicas dos dois dispositivos e estão ilustradas a seguir:

```
static void *get_in_addr(struct sockaddr *sa);
virtual void sendBytes(int nBytesToSend, BYTE *buf)=0;
virtual void receiveBytes(int nBytesToReceive, BYTE *buf)=0;
void sendUint(uint32_t m);
void receiveUint(uint32_t& m);
void sendVb(const vector<BYTE>& vb);
void receiveVb(vector<BYTE>& vb);
void sendImg(const Mat_<COR>& img);
void receiveImg(Mat_<COR>& img);
void sendImgComp(const Mat_<COR>& img);
void receiveImgComp(Mat_<COR> &img);
```

Na última versão da disciplina nós utilizamos principalmente as funções *…ImgComp* para mandar os frames das imagens e as funções *…Uint* para mandar strings de comando, por exemplo o estado atual da máquina de estados. Já nas classes filhas SERVER e CLIENT foram criadas as funções abaixo:

```
void waitConnection(); (apenas no SERVER)
void sendBytes(int nBytesToSend, BYTE *buf);
void receiveBytes(int nBytesToReceive, BYTE *buf);
```

No programa ***cliente6.cpp*** nós criamos o modelo para detectar os números utilizando como treinamento a biblioteca pública MNIST, foi utilizado o método de **Template Matching** que simplesmente desliza a imagem do modelo sobre a imagem de entrada (câmera) e compara o modelo e o patch da imagem de entrada sob a imagem do modelo. Ele retorna uma imagem em tons de cinza, onde cada pixel indica o quanto a vizinhança desse pixel corresponde ao modelo. Ao encontrar um número com alta probabilidade de acerto o programa manda a informação para o servidor.

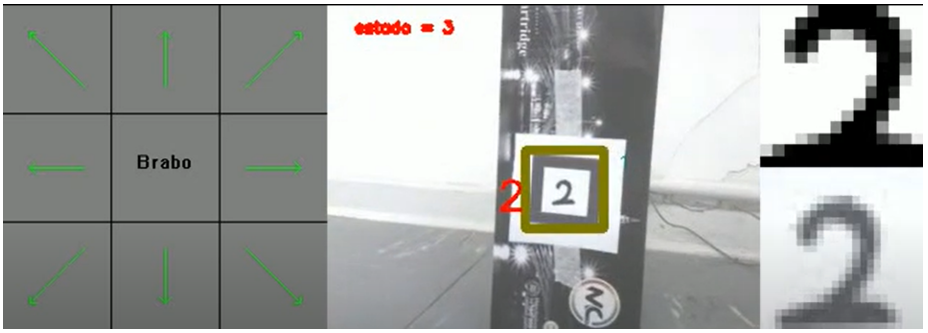
Por último, no programa **servidor6.cpp** estão todas as configurações dos periféricos do raspberry, como a velocidade e direção do motor, as configurações das fotos da câmera e a máquina de estados feita com switch case. Abaixo está um trecho do código. Todos os códigos estão comentados e serão enviados com o relatório.

```
// maquina de estados
switch (estado)
{
// segue o cartao
case 0:
    softPwmWrite(0, 100); softPwmWrite(1, 100);
    softPwmWrite(2, 100); softPwmWrite(3, 100);
    if (escalaCard >= 2)
        motores(x, card, escalaCard);
    else if (card == 1 && escalaCard < 2){
        estado = 1;
        t1=timePoint();
        softPwmWrite(0, 100); softPwmWrite(1, 100);
        softPwmWrite(2, 100); softPwmWrite(3, 100);
    }
    break;
// parada
case 1:
```

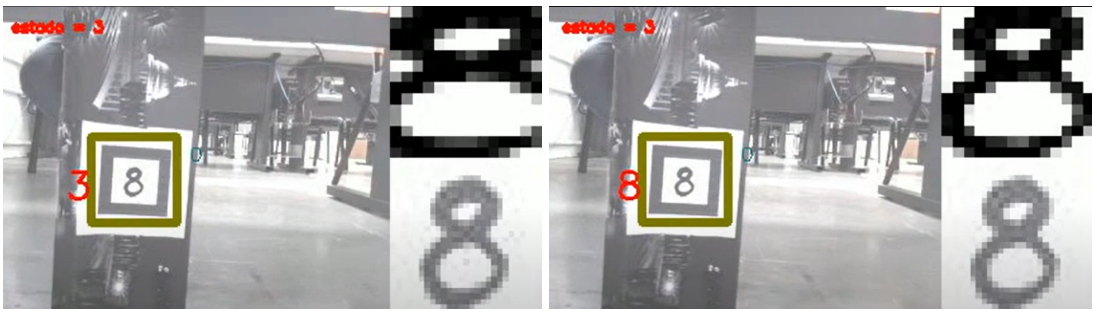
3. RESULTADOS

Para testar o sistema embarcado, primeiramente foram feitos alguns cards com números e o grupo percebeu que as imagens não estavam com uma boa qualidade e diminuía muito a eficiência do modelo, com isso nós ajustamos por tentativa e erro os parâmetros de contraste e brilho até que a porcentagem de acertos do modelo ficasse alta.

Dessa maneira, ao testar no circuito em sala de aula não tivemos grandes dificuldades. Na primeira tentativa o carrinho já havia percorrido 80% do caminho corretamente, porém ainda estava errando em alguns momentos por conta do tempo que deixamos ele em linha reta e com o número 8, estava confundindo com o 3. Abaixo está ilustrada uma imagem da visão do Raspberry acertando o número 2 durante o percurso.



Já na imagem abaixo é possível visualizar um momento que o modelo confunde o número 8 com o número 3 e outro momento que ele acerta a predição. É possível perceber que o número 3 foi escolhido por ele por conta de que o 8 foi recortado com zoom fazendo com que sua borda esquerda ficasse cortada, diferente da foto da direita.



4. CONCLUSÃO

É possível concluir que o sistema funcionou conforme o esperado. Nesta disciplina foram desenvolvidos todos os elementos de um sistema embarcado, a manufatura do hardware, a programação do software, utilização do sistema operacional Linux, os mecanismos de comunicação entre dois dispositivos usando protocolo Wifi, ferramentas de Machine Learning e a simulação e testes dos dispositivos. Além disso, aprendemos a importância de um código limpo e enxuto para projetos mais complexos, utilizando a orientação de objetos nos facilitou muito na codificação das funções.

É possível fazer melhorias em algumas partes deste projeto, como treinar um modelo de imagens melhor do que o utilizado (Template Matching), talvez criar uma rede neural usando convoluções, será necessário apenas adaptar um banco de dados com fotos mais realistas, uma dica seria pedir para cada aluno tirar 20 fotos dos próprios cards e depois juntar todas as imagens obtendo um banco de dados com mais de 500 imagens. Além disso, é possível utilizar algum método de controle como o PID para regular os parâmetros de zoom, brilho e contraste de acordo com a distância do card e velocidade do carrinho que talvez ajude na qualidade da imagem tirada. Por último, tivemos alguns problemas com a bateria que alimenta o raspberry, como melhoria gostaríamos de desenvolver um programa para monitorar a porcentagem de bateria restante do carrinho em tempo real.