ESCOLA POLITÉCNICA DA USP

Relatório

Experiência 2

PSI3422 - Laboratório de Sistemas Eletrônicos (2019)

Matheus Bordin Gomes - 9838028
Thomas Palmeira Ferraz - 9348985

A Experiência 2 da disciplina PSI3422 teve como objetivo desenvolver um carrinho controlado por um Raspberry Pi, utilizando visão computacional e aprendizagem de máquina. Esse relatório apresenta os principais detalhes do projeto desenvolvidos que não podem ser inferidos nas demonstrações feitas.

1. Comunicação TCP/IP

As funções feitas para a comunicação TCP/IP pelo grupo funcionaram nos programas de testes disponibilizados pela disciplina. Porém, começaram a apresentar erros na transmissão de imagens. Dessa forma, o grupo optou pela utilização do *header* com as funções de comunicação TCP/IP disponibilizado pelo professor. Também não utilizamos processamento paralelo na comunicação TCP/IP.

2. Template Matching

Para localizar a placa que o carrinho deve seguir, utilizou-se a técnica de *template matching*. Como o tamanho da placa vista pelo carrinho varia de acordo com a distância entre os dois, foi preciso fazer o *template matching* com diversos *templates* de tamanhos diferentes.

De acordo com o tamanho do menor *template*, maior a distância que o carrinho poderá achar a placa. E de acordo com o tamanho do maior *templates*, menor será a distância que o carrinho irá parar da placa. De acordo com as distâncias passadas pelo professor, o grupo decidiu usar nove *templates*, escalados em progressão geométrica, com razão 0.81 e começando com lado de 55 pixels. Não utilizou-se mais *templates*, pois quando eles estão muito pequenos, geram muito falsos positivos, fazendo com que o carrinho ande para direções erradas.

Como as placas possuem números escritos em seus interiores, foi preciso utilizar a técnica de *'don't care'* no interior dos *templates*. Como a imagem do *template* disponibilizada pelo professor tinha pixels com valores 1.0 apenas no interior da placa, utilizou-se as funções "somaAbsDois"e "dcReject"da biblioteca Cekeikon, para tornar o *template matching* invariante aos pixels no interior da placa.

O grupo optou por utilizar correlação cruzada (CV_TM_CCORR) para gerar invariância ao brilho. Além disso, o grupo também optou por utilizar a função "matchTemplateSame"da biblioteca Cekeikon, para facilitar comparar a saída do *template matching* com a imagem original e identificar o centro da placa.

3. Processamento Paralelo

Para acelerar a execução do *template matching*, que é executado nove vezes, o grupo optou por realizar o processamento paralelo nesse trecho do código. Para isso, foi utilizada a biblioteca OpenMP, por simplicidade na implementação.

No programa fase3, sem a utilização de processamento paralelo, obteve-se aproximadamente 14.2 frames por segundo. E com processamento paralelo, foi possível obter 21.1 frames por segundo. Já

no programa fase5, sem processamento paralelo, obteve-se 14.2 frames por segundo também. E com processamento paralelo, obteve-se 20.8 frames por segundo. Nota-se que esses testes foram feitos com o programa exibindo as imagens enquanto eram processadas. Ou seja, com o processamento paralelo, obteve-se um ganho significativo de, aproximadamente, sete frames por segundo.

4. Máquina de Estados Finita

A máquina de estados finita utilizada no sistema implementado é a apresentada na figura 1. Ela é baseada na apresentada na apostila "Aprendizagem de Máquina"da disciplina, porém com algumas diferenças. Nessa máquina de estados, o carrinho inicia no Estado 0, onde procura pela placa até que ela seja suficientemente grande para facilitar a leitura do dígito. Quando o carrinho chega a proximidade máxima da placa, o carrinho para e o sistema vai para o Estado 1. Nesse estado, é feita a temporização para garantir a parada do carrinho para que o frame que será utilizado para identificar o digito não fique tremido. Em sequência, o sistema vai para o Estado 2, no qual o carrinho tenta identificar o digito na placa encontrada. Caso não consiga, retorna para o Estado 0. Caso o digito seja identificado, o sistema vai para o Estado 3, onde é feita a temporização da execução da ação, a qual já inclui a parada do carrinho após a ação determinada pela placa. Não é tomado tanto cuidado nessa transição como é feito na transição do Estado 1 para o Estado 2, pois é menos problemático receber frames borrados nessa transição.

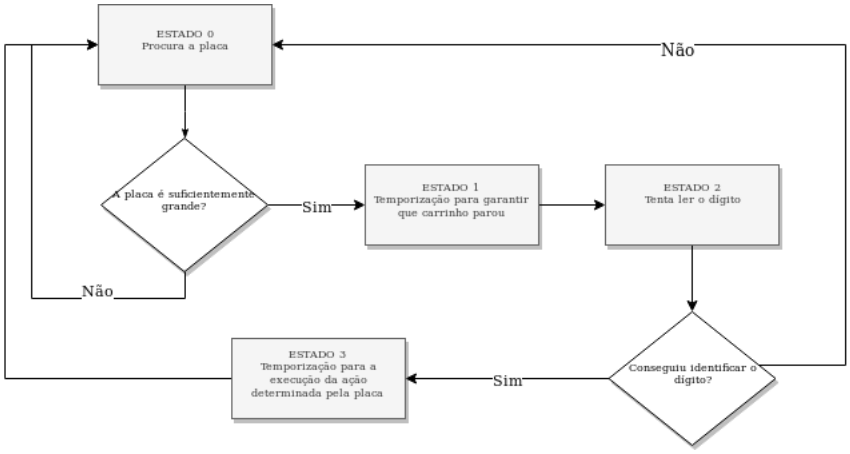

Figura 1. Máquina de estados finita utilizada no sistema.

5. Aprendizagem de Máquina

O algoritmo de aprendizagem de máquina escolhido pelo grupo para realizar o reconhecimento dos dígitos no interior das placas foi o *"Fast Approximate Nearest Neighbors"*, o qual já possui no suporte na biblioteca Cekeikon e apresentou bom desempenho. O treinamento do modelo é feito rapidamente e obteve-se uma taxa de erro de 2.99% de erro, no teste do algoritmo com as imagens para teste do MNIST. Além disso, o grupo obteve um tempo de treinamento de 376 ms e um tempo de predição de 59 ms no teste realizado com processamento paralelo na predição. Já no teste sem processamento paralelo, o tempo de predição foi de 179 ms.

6. Outros pontos relevantes

É interessante citar que a temporização das ações não foi feita com as funções "wait"e "sleep", visto que essas funções têm o ponto negativo de trava a execução do programa. No lugar das funções citadas, utilizou-se contagem de frames transmitidos/recebidos. A principal vantagem do método utilizado é que o programa continua a ser executado e a transmissão de imagens não é interrompida. Porém, como a transmissão de frames é dependente da velocidade da conexão TCP/IP, esse método não é tão preciso. Porém, a falta de precisão nos movimentos é compensada com o fato de que o

carrinho tem a capacidade de seguir a placa. Assim, basta que o carrinho consiga realizar o movimento com uma precisão mínima para visualizar a placa seguinte.

Referências

KIM, Hae Yong. Aprendizagem de Máquina Avançada. **Universidade de São Paulo**, São Paulo. Disponível em: http://www.lps.usp.br/hae/apostila/mle_avancada.pdf. Acesso em: 5 de dezembro de 2019.

KIM, Hae Yong. Template matching (casamento de máscara, ou casamento de modelo). **Universidade de São Paulo**, São Paulo. Disponível em: http://www.lps.usp.br/hae/apostila/tmatch.pdf. Acesso em: 5 de dezembro de 2019.

KIM, Hae Yong. Aprendizagem de Máquina. **Universidade de São Paulo**, São Paulo. Disponível em: http://www.lps.usp.br/hae/apostilaraspi/aprendizagem.pdf. Acesso em: 5 de dezembro de 2019.