Trabalho 1 - Introdução à Visão Computacional

Leonardo Miranda e Guilherme Scherer
5 de Maio de 2025

Objetivo

O trabalho 1 (T1) da disciplina tem como objetivo desenvolver um método que utilize algoritmos de processamento de imagem que resolva um problema da escolha do grupo de alunos.

Enunciado

O grupo (1 ou 2 pessoas) deve desenvolver uma metodologia que trabalhe no domínio de imagens (podendo ser imagens paradas ou sequências destas em vídeos) e que atendam os seguintes critérios:

- A aplicação deve usar OpenCV;
- Pode ser escrita em qualquer linguagem;
- Deve usar features das imagens;
- **A apresentação do trabalho será na aula do dia 05/05/2025;**
- O problema a ser resolvido terá uma POC, que será desenvolvida para prova de conceito;
- O aluno deve apresentar o pipeline do trabalho desenvolvido;
- O dataset pode ser público ou construído pelo aluno;
- Os resultados devem ser avaliados e não há necessidade de comparar com trabalhos da literatura;
- O trabalho deverá ser apresentado oralmente seguindo a metodologia científica, e ainda um artigo deverá ser produzido, com os detalhes do trabalho. Usar formato SBC com duas colunas.

Critérios de Avaliação

- Trabalho individual ou em duplas;
- Vedado o uso de redes neurais profundas. (Única arquitetura permitida é a MLP);
- O grupo apresentará o pipeline de visão computacional construído, justificando a escolha de todas as técnicas usadas;

- A apresentação deverá contemplar uma demo do programa, com imagem capturada durante a apresentação ou previamente;
- O código desenvolvido e o artigo produzido serão entregues em sala no Moodle, na mesma data da apresentação.

Introdução:

Durante a aula de visão computacional, fomos apresentados a vários exemplos práticos de como utilizar esta área da tecnologia ao nosso favor, e como podemos utilizar a visão computacional pra melhorar o desempenho de nossos algoritmos, fazendo com que eles entendam o nosso mundo através de imagens.

Primeiramente, antes de sequer iniciarmos os passos deste trabalho, pensamos em conceitos e em áreas de aplicação. O primeiro que nos veio a mente foi a detecção facial e reconhecimento de rostos, mas logo descartamos a ideia, por ser algo complexo e que talvez precise de redes neurais. Em seguida, pensamos sobre a deteção de placas de veículos ao entrar no estacionamento, para realizar o pagamento sem precisar do ticket, por exemplo.

Mesmo a ideia sendo interessante, nós realizamos o nosso trabalho se focando principalmente na detecção de vagas de estacionamento disponíveis, que ajudam muito na hora de encontrar lugares para estacionar. Esta funcionalidade está presente em shoppings como Iguatemi e Bourbon, no estacionamento do aeroporto e até mesmo em alguns estabelecimentos corporativos.

Em resumo, nos interessamos muito na ideia de realizar um sistema como esse e entender como a visão computacional pode ser útil no nosso dia a dia e, quem sabe, estudar formas de aplicar isto no estacionamento da própria universidade.

Desenvolvimento:

Iniciamos o desenvolvimento do nosso projeto montando um pipeline de ações que o processo vai realizar. Com isso, poderemos ver o que pode ser solucionado com visão computacional e o que vai ser necessário o *input* do usuário (ou de uma rede neural) para ser executado.

Nosso projeto iniciou com a captura de algumas imagens de estacionamento com a câmera posicionada de forma isométrica, de forma que possam ser identificadas de forma mais clara e objetiva quais vagas estavam ocupadas e quais não estavam. A imagem que utilizamos como base foi a seguinte:

Utilizamos esta imagem pois ela possui uma visão isométrica muito boa e, além disso, ela proporciona uma ótima visão de carros, tendo vagas regulares, carros de diversas cores e alguns objetos, como árvores, que podem nos ajudar a determinar a melhor forma de aplicar a visão computacional.

Ademais, utilizamos mais 3 imagens além destas para complementar o entendimento de entropia e conseguirmos criar um limiar correto de entropia, onde o valor fixo faria sentido. Inclusive, nós adquirimos uma imagem do estacionamento da PUCRS para comparar e trazer essa aplicação mais próxima da nossa realidade.

Aplicamos o nosso pipeline e criamos um Jupyter notebook, contendo todos os métodos necessários para rodar o processo. O nosso pipeline é composto de 5 passos:

1. Escolha da imagem pelo usuário;
    1.1. O usuário pode realizar o download de uma imagem qualquer de estacionamento e colocar na mesma pasta do jupyter;
    1.2. Após isso, ao executar o código, será necessário que o usuário especifique o nome do arquivo a ser lido, com a extensão junto (Ex: vagas1.png);

2. Marcação das vagas de estacionamento;
    2.1. Como não estamos utilizando uma MLP ou LLM, precisamos que o usuário marque manualmente as vagas determinadas. Cada vaga é representada por um quadrado ou retângulo, destacado pelo usuário;
    2.2. OBS: Em caso de vagas oblíquas ou numa visão não isométricas, sugerimos que seja marcado o mais próximo da vaga possível;

3. Marcação das coordenadas das vagas em formato de lista e cálculo da entropia;
    3.1. Utilizamos um sistema de lista para guardar as informações das vagas, basicamente mantendo o sistema organizado;
    3.2. Após isso, transformamos a vaga para a escala cinza e calculamos a entropia de cada área (vaga) e printa o valor na tela;

4. Comparação da entropia da vaga com o valor limite da entropia definido;
    4.1. Calculamos a entropia de duas formas diferentes, primeiramente, realizamos um teste com 4 imagens diferentes e chegamos no valor 6.25 de entropia seria

o ideal. Claro que, com imagens diferentes com iluminação diferente, pode não ser tão eficiente;

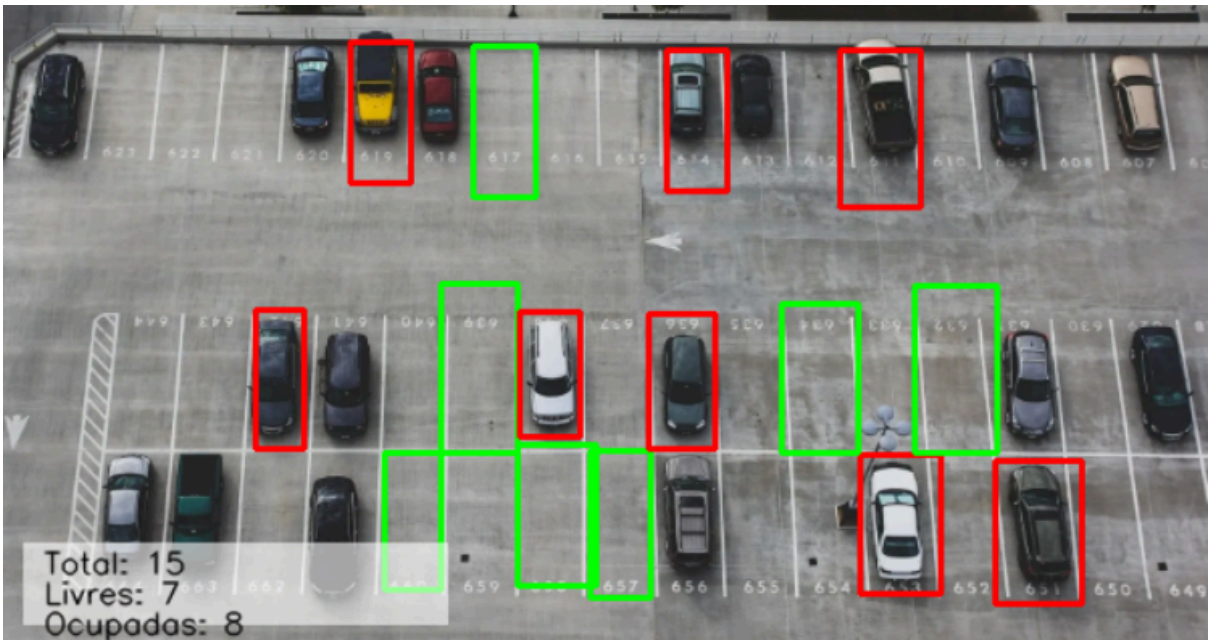
4.2. Também calculamos o limite utilizando a média da entropia, que performou muito bem, mas, o grande problema desta forma de calcular o limite da entropia é que, caso todas as vagas estejam ocupadas, a média sempre vai excluir as vagas com menos entropia, mesmo que elas estejam ocupadas;

4.3. Após a comparação ser feita, seja pela média ou seja pelo valor fixo calculado das imagens, o algoritmo destaca as vagas novamente e exibe na tela as vagas ocupadas (em vermelho) e as vagas livres (em verde);

5. Exibe relatórios e legendas;

5.1. Por fim, o algoritmo cria um relatório de vagas disponíveis para estacionar e as vagas ocupadas, assim como as vagas totais;

5.2. Também é gerada uma imagem com as vagas ocupadas e livres destacadas em suas respectivas cores, complementada com uma legenda. Segue um exemplo de imagem gerada;



Resultados e Conclusão:

A partir do nosso trabalho e da nossa realização através dos códigos, podemos concluir que é possível utilizar os métodos da visão computacional para identificar vagas disponíveis em um estacionamento.

Levando para uma visão mais corporativa, poderíamos incorporar câmeras em cada vaga, ou tirar uma foto do estacionamento a cada 5 minutos, por exemplo, e utilizar este algoritmo para detectar a quantidade de vagas disponíveis e informar ao usuário.

Referências:

[1] *Docxygen*. OpenCV Documentation. **docxygen**, 2025. Disponível em: https://docs.opencv.org/4.x/index.html. Acesso em: 05 maio 2025.

[2] *Dev Ideias*. Sistema Contador de Vagas de Estacionamento: Opencv Python. **YouTube**, 2022. Disponível em: https://www.youtube.com/watch?v=k5nZkYGQOx0&t=197s. Acesso em: 03 maio 2025.

[3] Link do GitHub do Grupo com os códigos utilizados: https://github.com/leomiranda16/t1-IVC.