**NOME:** Matheus Guimaraes Liporace

**MATRICULA:** 202302417949

### **Introdução:**

A Computação Visual é uma das bases da tecnologia moderna. Ela envolve tudo que diz respeito a gerar, manipular e interpretar imagens e cenas visuais com o uso de computadores. Dentro desse campo amplo, existem várias áreas com objetivos bem distintos, apesar de muitas vezes se complementarem.

Este trabalho apresenta uma visão prática e comparativa de quatro áreas fundamentais da Computação Visual: Síntese de Imagens (ou Computação Gráfica), Processamento de Imagens, Visão Computacional e Visualização Computacional. Para cada uma delas, selecionamos um projeto de código aberto disponível no GitHub que serve como exemplo claro de aplicação da área.

### **1. Síntese de Imagens (Computação Gráfica)**

A Computação Gráfica é voltada para a criação de imagens a partir de descrições matemáticas e modelos 3D. Ou seja, não parte de uma imagem pronta, mas sim de dados como coordenadas de vértices, informações sobre luz, câmera e texturas.

**Objetivo principal:** Gerar imagens sintéticas a partir de modelos abstratos.

**Projeto analisado: ssloy/tinyrenderer**

Esse projeto é praticamente um curso de renderização 3D em C++. Ele mostra passo a passo como criar um renderizador do zero, desde as linhas básicas até sombras e texturas.

O programa usa arquivos .obj (modelos 3D) e imagens de textura para gerar imagens 2D renderizadas. Toda a imagem é gerada via código, sem nenhuma entrada visual prévia. É um ótimo exemplo de como a descrição matemática de uma cena pode se transformar em uma imagem completa.

A execução é simples: você compila os arquivos C++ e o programa gera uma imagem como resultado.

### **2. Processamento de Imagens**

Aqui, o foco está em modificar imagens que já existem. Você entra com uma imagem, aplica um algoritmo e sai com uma nova imagem modificada — por exemplo, com mais contraste, em preto e branco ou desfocada.

**Objetivo principal:** Realçar ou transformar imagens existentes.

**Projeto analisado: danielgatis/rembg**

Este é um projeto extremamente popular em Python que remove o fundo de imagens. É um exemplo perfeito e moderno de processamento de imagens, onde a tarefa é segmentar o objeto principal e criar uma nova imagem com fundo transparente.

O programa utiliza um modelo de inteligência artificial pré-treinado para analisar a imagem de entrada e identificar quais pixels pertencem ao objeto principal e quais pertencem ao fundo. Com base nessa análise, ele gera uma nova imagem.

A saída é uma imagem no formato PNG com o fundo removido, pronta para ser usada em outras edições. O fluxo é claro: imagem com fundo entra, algoritmo de segmentação roda, imagem sem fundo sai.

Para rodar, basta instalar a biblioteca com pip e usar um comando simples no terminal, como rembg i input.jpg output.png.

### **3. Visão Computacional**

Diferente do Processamento de Imagens, a Visão Computacional vai além da manipulação da imagem — ela tenta "entender" o que está na imagem. É como ensinar a máquina a enxergar e identificar objetos, pessoas, movimentos etc.

**Objetivo principal:** Extrair informações relevantes a partir de imagens ou vídeos.

**Projeto analisado: ultralytics/yolov5**

Esse repositório traz a implementação de um dos modelos mais famosos para detecção de objetos: o YOLOv5. Ele usa redes neurais profundas para detectar objetos em tempo real com ótima precisão.

A entrada pode ser uma imagem, um vídeo ou a webcam. O modelo retorna as posições e categorias dos objetos detectados, junto com a confiança de cada detecção. Ou seja, o foco aqui não é gerar uma imagem nova, mas sim gerar dados estruturados sobre o que foi encontrado na imagem.

O desenho das caixas é apenas uma forma visual de representar essas informações — o que importa mesmo são os dados que o modelo extrai.

Rodar o projeto é simples: clone o repositório, instale as dependências com pip, e execute o script detect.py com a fonte desejada.

### **4. Visualização Computacional**

Por fim, temos a Visualização Computacional, que transforma dados abstratos — como planilhas ou arquivos CSV — em representações gráficas que facilitam a análise e a interpretação por humanos.

**Objetivo principal:** Representar visualmente dados complexos ou abstratos.

**Projeto analisado: Galeria Oficial da Biblioteca Seaborn**

Seaborn é uma das bibliotecas Python mais importantes para visualização de dados estatísticos. O seu repositório oficial e, por extensão, sua galeria de exemplos, são a representação canônica desta área. A galeria não é apenas um projeto, mas uma coleção de dezenas de exemplos de alto nível que demonstram o princípio fundamental da Visualização Computacional.

O fluxo em cada exemplo da galeria é o mesmo:

1. **Entrada:** Carrega-se um conjunto de dados abstratos, como o famoso dataset "titanic" ou "penguins", que são tabelas com linhas e colunas.
2. **Processo:** Uma única função de alto nível da Seaborn (ex: sns.scatterplot, sns.histplot, sns.heatmap) é chamada. Essa função mapeia as variáveis dos dados (as colunas da tabela) a atributos estéticos do gráfico (eixos, cores, tamanhos).
3. **Saída:** Um gráfico estatístico sofisticado e de fácil interpretação.

Por exemplo, um scatterplot mostra a relação entre duas variáveis contínuas, enquanto um heatmap revela a correlação entre todas as variáveis de um dataset de uma só vez. Cada exemplo transforma uma tabela de dados brutos em um insight visual.

A execução é trivial: basta instalar a biblioteca (pip install seaborn) e rodar o código de qualquer exemplo da galeria oficial.

### **Comparando as Áreas:**

Apesar de todas lidarem com imagens ou visualizações, essas áreas têm propósitos bem diferentes:

* **Computação Gráfica** cria imagens a partir de modelos e descrições matemáticas.
* **Processamento de Imagens** transforma imagens já existentes, aplicando filtros e modificações.
* **Visão Computacional** tenta entender o que está em uma imagem e gerar dados sobre isso.
* **Visualização Computacional** pega dados brutos e os transforma em gráficos ou representações visuais para facilitar a análise.

### **Conclusão:**

Cada uma dessas áreas da Computação Visual resolve um tipo específico de problema, com técnicas, entradas e saídas diferentes. A Computação Gráfica foca na criação do visual; o Processamento de Imagens o modifica; a Visão Computacional o interpreta; e a Visualização Computacional transforma dados invisíveis em imagens compreensíveis.

Embora em muitas aplicações reais essas áreas se combinem (como em jogos, sistemas de realidade aumentada ou veículos autônomos), é importante entender onde cada uma começa e termina. Os projetos analisados aqui deixam essas diferenças bem claras na prática.

### **Referências**

1. **Tiny Renderer:** Dmitry V. Sokolov, GitHub,<https://github.com/ssloy/tinyrenderer>
2. **Rembg:** Daniel Gatis, GitHub,<https://github.com/danielgatis/rembg>
3. **YOLOv5:** Ultralytics, GitHub,<https://github.com/ultralytics/yolov5>
4. **Seaborn: Statistical data visualization:** Michael Waskom, GitHub,<https://github.com/mwaskom/seaborn>