Lucas Lima

Desenvolvimento de uma Ferramenta de Pintura 3D com Rastreamento por Visão Artificial para Determinação de Profundidade

Natal – RN

Janeiro de 2025

Lucas Lima

Desenvolvimento de uma Ferramenta de Pintura 3D com Rastreamento por Visão Artificial para Determinação de Profundidade

Trabalho de Conclusão de Curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte, apresentado como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Agostinho de Medeiros Brito Júnior

Universidade Federal do Rio Grande do Norte – UFRN

Curso de Engenharia de Computação

Natal – RN

Janeiro de 2025

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Central Zila Mamede

Lima, Lucas de Azevedo.
Desenvolvimento de uma Ferramenta de Pintura 3D com Rastreamento por Visão Artificial para Determinação de Profundidade / Lucas de Azevedo Lima. - 2025.
43f.: il.

Monografia (Graduação) - Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, Engenharia de Computação, Natal, 2025.
Orientação: Prof. Dr. Agostinho de Medeiros Brito Júnior.

1. Processamento Digital de Imagen - Monografia. 2. Visão Artificial - Monografia. 3. OpenCV - Monografia. 4. OpenGL - Monografia. 5. Desenho interativo - Monografia. I. Brito Júnior, Agostinho de Medeiros. II. Título.

RN/UF/BCZM CDU 004

Elaborado por Raimundo Muniz de Oliveira - CRB-15/429

Lucas Lima

Desenvolvimento de uma Ferramenta de Pintura 3D com Rastreamento por Visão Artificial para Determinação de Profundidade

Trabalho de Conclusão de Curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte, apresentado como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Agostinho de Medeiros Brito Júnior

Trabalho aprovado. Natal – RN, 16 de janeiro de 2025

Prof. Dr. Agostinho de Medeiros Brito Júnior - Orientador
UFRN

Prof. Dr. Pablo Javier Alsina
UFRN

Prof. Dr. Adelardo Adelino Dantas de Medeiros
UFRN

Natal – RN
Janeiro de 2025

Dedico este trabalho aos meus pais, Senhor Sadir e Dona Josilene pelo amor, apoio e dedicação incondicional em todas as etapas da minha vida; aos meus amigos, que sempre estiveram ao meu lado nos momentos mais desafiadores; e aos meus professores, que, com paciência e sabedoria, me guiaram ao longo deste caminho.

A cada um de vocês, minha eterna gratidão por acreditarem em mim e me inspirarem a seguir em frente, mesmo diante das dificuldades.

AGRADECIMENTOS

A realização deste trabalho só foi possível graças ao apoio e dedicação de várias pessoas que contribuíram direta ou indiretamente para o meu desenvolvimento pessoal e acadêmico.

Agradeço, primeiramente, aos meus pais, pelo amor incondicional, paciência e incentivo constante ao longo de toda a minha jornada. Vocês foram meu alicerce e minha maior fonte de inspiração nos momentos de dificuldade.

Aos meus amigos e colegas, pela parceria, pelas palavras de encorajamento e pelas conversas que aliviaram o peso dos desafios, em especial Gabriel, Ítalo, Jadson, Inaldo e Dorgival. Vocês tornaram essa caminhada mais leve e significativa.

Ao meu orientador, pela orientação precisa, pelas valiosas sugestões e pela confiança depositada em mim para realizar este trabalho. Sua experiência e paciência foram essenciais para o desenvolvimento deste projeto.

Aos professores que, ao longo da minha formação, compartilharam seus conhecimentos e me motivaram a buscar sempre mais. Vocês deixaram uma marca indelével na minha trajetória.

Por fim, agradeço a todas as pessoas que, de alguma forma, contribuíram para a realização deste trabalho, incluindo aqueles que acreditaram em mim, mesmo quando duvidei de mim mesmo. A todos, minha eterna gratidão.

Dedico este trabalho a todos que me ajudaram a chegar até aqui.

"A ciência é construída com fatos, assim como uma casa é construída com pedras; mas uma coleção de fatos não é ciência, do mesmo modo que um monte de pedras não é uma casa."

(Henri Poincaré)

RESUMO

Apresenta o desenvolvimento de um sistema que integra OpenCV e OpenGL para criar uma aplicação interativa capaz de rastrear cores específicas em tempo real e projetar essas informações em um espaço 3D virtual. O OpenCV é utilizado para captura e processamento de imagens, enquanto o OpenGL renderiza os dados em uma caixa cúbica tridimensional. A sincronização entre as bibliotecas é garantida por multithreading, permitindo operações simultâneas e respostas em tempo real. O sistema demonstrou eficiência e versatilidade, embora tenha limitações como sensibilidade à iluminação, sugerindo possibilidades de aprimoramento em trabalhos futuros. A solução contribui para a computação gráfica e visão computacional, destacando seu potencial para aplicações interativas, com especial destaque para uso em aplicações educacionais.

Palavras-chaves: Processamento Digital de Imagens, Visão Artificial, OpenCV, OpenGL, Desenho interativo, Visualização 3D.

ABSTRACT

This work presents the development of a system that integrates OpenCV and OpenGL to create an interactive application capable of tracking specific colors in real time and projecting this information into a 3D virtual space. OpenCV is used for image capture and processing, while OpenGL renders the data in a three-dimensional cubic box. Synchronization between the libraries is ensured through multithreading, enabling simultaneous operations and real-time responsiveness. The system demonstrated efficiency and versatility, although it has limitations such as sensitivity to lighting, suggesting potential improvements in future work. The solution contributes to computer graphics and computer vision, highlighting its potential for interactive applications, with a special emphasis on educational uses.

Keywords: Digital Image Processing, Artificial Vision, OpenCV, OpenGL, Interactive Drawing, 3D Visualization..

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A computação gráfica e a visão computacional têm se consolidado como áreas essenciais para o desenvolvimento de sistemas interativos, possibilitando aplicações que vão desde simulações educacionais até experiências imersivas em jogos. A integração de bibliotecas especializadas, como OpenCV e OpenGL, oferece uma plataforma robusta para combinar a captura de informações visuais do mundo real com a renderização gráfica em 3D.

Neste trabalho, propõe-se um sistema interativo que rastreia cores específicas em tempo real e utiliza essas informações para criar desenhos em um ambiente tridimensional. A interação ocorre a partir do rastreamento de objetos coloridos capturados pela câmera, cujas posições são utilizadas para desenhar e manipular elementos no espaço virtual.

A profundidade (*eixo* Z) do sistema é calculada com base em características extraídas dos objetos rastreados. Utilizando algoritmos de processamento de imagem do OpenCV, o sistema identifica a posição (X, Y) e o tamanho relativo (Z) dos objetos capturados pela câmera. Essa abordagem permite representar os objetos em um ambiente tridimensional de forma precisa e eficiente, utilizando apenas uma câmera convencional e algoritmos de visão computacional.

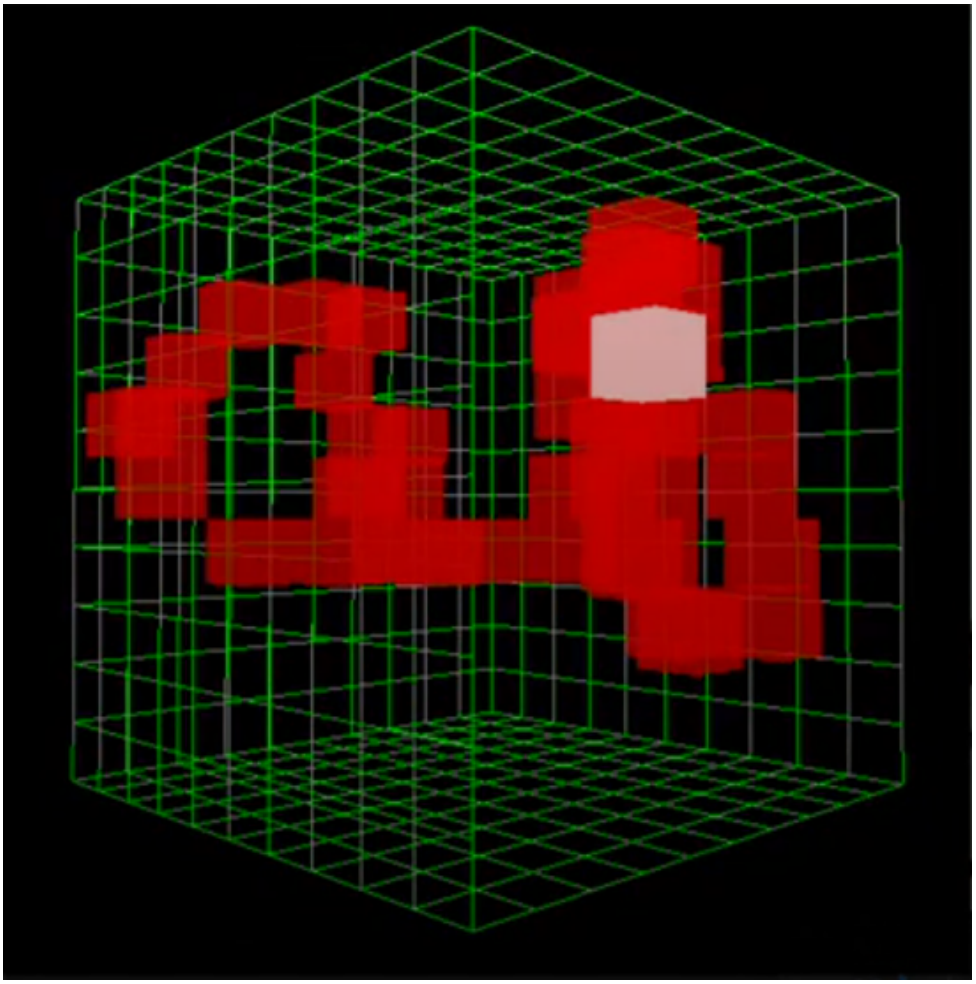


Figura 1 – Exemplo do resultado final: objeto sendo rastreado e representado em um ambiente 3D.

A aplicação faz uso do OpenCV para análise e processamento de imagens, enquanto o OpenGL é empregado para criar e manipular objetos gráficos em um espaço virtual. A sincronização entre as duas tecnologias é realizada por meio de multithreading, garantindo a execução simultânea e eficiente das tarefas. Essa combinação proporciona um sistema acessível e flexível, capaz de explorar a interatividade em 3D de maneira simplificada e de baixo custo.

1.1 Trabalhos Relacionados

A integração de tecnologias para rastreamento de movimentos e visualização tridimensional tem sido amplamente explorada em diversas áreas, como realidade aumentada, jogos e interfaces interativas. Um dos avanços mais notáveis nesse campo foi o desenvolvimento do KinectFusion, apresentado por Izadi et al. (IZADI et al., 2011a; IZADI et al., 2011b). Esse

sistema utiliza o sensor Kinect para realizar reconstrução 3D em tempo real, integrando dados de profundidade e cor para criar modelos detalhados do ambiente. A combinação de sensores de movimento com algoritmos avançados de reconstrução trouxe uma nova perspectiva para aplicações que exigem interação em ambientes tridimensionais.

O KinectFusion destaca-se por sua abordagem inovadora, que processa fluxos de dados em tempo real e utiliza algoritmos otimizados para estimar a pose da câmera e gerar modelos 3D precisos. Além disso, o sistema foi projetado para ser integrado com pipelines gráficos, permitindo o uso de bibliotecas como OpenGL para visualização interativa dos dados reconstruídos. Esse trabalho serve como referência importante para sistemas que buscam integrar visão computacional com gráficos 3D, evidenciando a viabilidade dessa combinação para aplicações interativas.

Além do Kinect, outros dispositivos como o PS Move também foram utilizados em interfaces interativas. Projetos como a PS Move API permitem rastrear a posição e orientação do controlador em tempo real, utilizando câmeras para capturar os movimentos e algoritmos de processamento de imagem para traduzir os dados em interações virtuais (PERL, 2010). Esses dispositivos ilustram o potencial de soluções acessíveis e de baixo custo para interatividade 3D, mesmo quando comparados a sensores mais sofisticados como o Kinect.

Este trabalho busca inspiração nos avanços apresentados por essas tecnologias, explorando uma abordagem alternativa baseada no rastreamento de cores utilizando bibliotecas OpenCV e OpenGL. Apesar de não utilizar sensores de profundidade, a aplicação proposta visa demonstrar como sistemas simples podem ser adaptados para proporcionar interatividade em ambientes tridimensionais.

1.2 Motivação

O avanço de tecnologias de hardware, como câmeras de alta resolução, e software, como bibliotecas de código aberto, possibilitou o desenvolvimento de aplicações interativas mais acessíveis. No entanto, muitos sistemas de interação 3D dependem de dispositivos especializados, como sensores de profundidade de custo elevado, o que pode limitar seu uso em aplicações domésticas. Este trabalho é motivado pela possibilidade de criar uma solução simples e acessível, utilizando apenas uma câmera convencional e algoritmos de

rastreamento de cores para proporcionar uma experiência interativa tridimensional.

Soluções dessa natureza oferecem amplas oportunidades de desenvolvimento de ferramentas interativas, principalmente para crianças, que são motivadas por atividades lúdicas. A experiência de desenhar utilizando aparatos como o PS Move da Sony, por meio do projeto Move.me, por exemplo, é bem divertida. Esse projeto, lançado em 2011, permitiu a desenvolvedores e educadores criarem aplicações que exploram a interatividade proporcionada pelo PS Move, combinando atividades de desenho e controle de movimentos (Sony Computer Entertainment, 2011).

Aplicativos como o Move & Draw (Qubacy, 2024) também ilustram o potencial de experiências criativas em ambientes tridimensionais, utilizando dispositivos móveis como interface. No entanto, o presente trabalho busca uma abordagem diferente, explorando o uso de rastreamento de cores para criar uma experiência interativa acessível, permitindo que os usuários desenhem em 3D utilizando apenas uma câmera comum. Ao estender essa possibilidade para um ambiente tridimensional, amplia-se a realimentação visual e cria-se uma base para estimular atividades de aprendizagem, inclusive multiusuário.

1.3 Objetivos

1.3.1 Objetivo Geral

O presente trabalho se propõe a desenvolver um sistema que integra OpenCV e OpenGL para rastrear e identificar posição e tamanho de regiões com cores específicas presentes em um aparato de interação de baixo custo e promover visualização tridimensional.

1.3.2 Objetivos Específicos

- Implementar um sistema de rastreamento de cores usando OpenCV.
- Converter os dados obtidos em coordenadas tridimensionais.
- Sincronizar as bibliotecas OpenCV e OpenGL por meio de programação multithreading.
- Renderizar objetos em um espaço 3D interativo com base nos dados rastreados.

1.4 Estrutura do Trabalho

Este trabalho está organizado em cinco capítulos. No **Capítulo 1**, apresenta-se a introdução, destacando a motivação, os objetivos e os conceitos básicos do projeto. O **Capítulo 2** aborda o referencial teórico, explorando as principais características das bibliotecas OpenCV e OpenGL, além de conceitos relacionados ao processamento de imagens e gráficos 3D. No **Capítulo 3**, detalha-se a metodologia utilizada no desenvolvimento do sistema, desde a captura de imagens, ferramentas e técnicas usadas até a integração das bibliotecas. O **Capítulo 4** apresenta os resultados obtidos e discute as limitações e potenciais melhorias. Por fim, o **Capítulo 5** apresenta as conclusões do trabalho e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os conhecimentos necessários para a compreensão do desenvolvimento deste trabalho, incluindo as ferramentas utilizadas e os fundamentos de processamento de imagens e gráficos tridimensionais. Cada tecnologia empregada é descrita em relação ao papel que desempenha na aplicação, além de conceitos básicos relevantes para o entendimento do sistema proposto.

2.1 Ferramentas Utilizadas

2.1.1 OpenCV

O OpenCV (Open Source Computer Vision Library) é uma biblioteca de código aberto amplamente utilizada para aplicações de visão computacional. Ela oferece uma vasta gama de ferramentas para captura, processamento e análise de imagens e vídeos. No contexto deste trabalho, o OpenCV é utilizado para:

- Capturar imagens em tempo real de uma câmera.
- Aplicar técnicas de detecção de cores, utilizando máscaras baseadas em faixas de cores.
- Realizar o rastreamento de posições a partir de contornos detectados, convertendo-as para coordenadas tridimensionais.

A escolha do OpenCV deve-se à sua robustez e à ampla documentação disponível, o que facilita a implementação e solução de problemas durante o desenvolvimento.

2.1.2 OpenGL

O OpenGL (Open Graphics Library) é uma API padrão para renderização de gráficos em 2D e 3D. Ele permite a criação de ambientes gráficos interativos e é amplamente utilizado em jogos, simulações e aplicações científicas. No sistema proposto, o OpenGL desempenha as seguintes funções:

- Renderizar um ambiente tridimensional interativo (caixa cúbica).
- Atualizar as posições dos objetos gráficos com base nos dados obtidos pelo OpenCV.
- Permitir a manipulação do ambiente 3D, incluindo rotação e interação do usuário.

A escolha pelo OpenGL baseia-se em sua eficiência e compatibilidade com diversos sistemas operacionais, além de permitir integração direta com linguagens de programação como C++.

2.2 Fundamentos de Processamento de Imagens

O processamento de imagens consiste em uma série de operações realizadas para extrair ou manipular informações contidas em uma imagem digital. No contexto deste trabalho, são utilizados os seguintes conceitos:

2.2.1 Detecção de Cores

A detecção de cores é realizada utilizando faixas específicas de valores de intensidade nas representações RGB da imagem. O OpenCV oferece a função `inRange`, que permite isolar pixels dentro de uma faixa de cor especificada, criando uma máscara binária. Essa etapa é essencial para identificar os objetos coloridos que serão rastreados.

2.2.2 Contornos e Momentos

Após a aplicação da máscara, os contornos dos objetos detectados são extraídos usando a função `findContours`. Os contornos são utilizados para calcular o centro de massa (momento) do objeto, que é então convertido em uma coordenada tridimensional.

2.2.3 Correção de Gama

A correção de gama é uma técnica utilizada para ajustar a luminosidade das imagens, de modo a compensar variações de iluminação que possam prejudicar o processamento subsequente. Em termos matemáticos, a correção aplica uma função não linear aos valores de intensidade dos pixels, o que é particularmente útil em aplicações de visão computacional (POYNTON, 1993).

2.3 Fundamentos de Gráficos Tridimensionais

Os gráficos tridimensionais são responsáveis pela visualização do ambiente virtual. No contexto deste trabalho, os seguintes conceitos são utilizados:

2.3.1 Sistema de Coordenadas

O OpenGL utiliza um sistema de coordenadas tridimensional para posicionar objetos no espaço. Cada ponto do sistema é definido por três valores: x, y e z, que representam a largura, altura e profundidade, respectivamente.

2.3.2 Renderização de Objetos

A renderização é o processo de desenhar objetos no ambiente virtual. Neste trabalho, são utilizados cubos sólidos, que são posicionados no espaço com base nas coordenadas geradas pelo OpenCV.

2.3.3 Interação com o Ambiente

O sistema permite que o usuário interaja com o ambiente 3D, manipulando a posição dos objetos rastreados. Além disso, o movimento do mouse é usado para controlar a rotação do ambiente, oferecendo uma perspectiva mais dinâmica.

3 O SISTEMA DESENVOLVIDO

Neste capítulo, é descrito o processo de desenvolvimento do sistema proposto, detalhando as etapas de implementação e integração das bibliotecas OpenCV e OpenGL para rastreamento de cores e visualização tridimensional. Cada etapa do processo é apresentada com ilustrações e diagramas que auxiliam na compreensão da solução implementada.

3.1 Ferramentas Utilizadas

3.1.1 Visual Studio Code (VSCode)

O Visual Studio Code foi utilizado como ambiente de desenvolvimento integrado (IDE). Essa ferramenta foi escolhida devido à sua leveza, facilidade de uso e suporte a extensões específicas para C++, como configurações de depuração e integração com compiladores como o GCC. Além disso, sua popularidade e documentação ampla facilitaram a configuração e o uso no projeto (Microsoft,).

3.1.2 Compilador GCC

O compilador GCC (GNU Compiler Collection) foi utilizado para compilar o código-fonte em C++. O GCC é conhecido por sua compatibilidade com diversas bibliotecas e sistemas operacionais, sendo amplamente utilizado no desenvolvimento de projetos que envolvem bibliotecas como OpenCV e OpenGL (GNU Project,). Sua robustez garantiu um desempenho eficiente no projeto.

3.1.3 OpenCV

A biblioteca OpenCV foi empregada para o processamento de imagens e rastreamento de cores. Ela oferece ferramentas otimizadas para tarefas de visão computacional, como segmentação de cores, detecção de contornos e manipulação de máscaras binárias. O OpenCV é amplamente utilizado na comunidade de desenvolvedores e pesquisadores, sendo um padrão de mercado para processamento de imagens (OpenCV Team,).

3.1.4 OpenGL e GLUT

A biblioteca OpenGL foi utilizada para renderização gráfica e criação de um ambiente tridimensional interativo. O OpenGL oferece recursos avançados de projeção, blending e manipulação de matrizes, sendo amplamente reconhecido como uma API padrão para desenvolvimento gráfico (Khronos Group,). Para facilitar o gerenciamento de janelas e eventos do usuário, foi utilizado o GLUT (OpenGL Utility Toolkit), uma biblioteca complementar que simplifica a interação gráfica (FreeGLUT Development Team,).

3.1.5 Multithreading com C++

O uso de multithreading foi fundamental para integrar os módulos OpenCV e OpenGL, permitindo que o rastreamento de cores e a renderização gráfica fossem executados simultaneamente. Para garantir a consistência dos dados compartilhados entre threads, foram utilizados mecanismos de sincronização como `std::mutex`, disponíveis na biblioteca padrão do C++ (WILLIAMS, 2019).

3.2 Técnicas e Metodologias

3.2.1 Correção de Gama

A correção de gama foi utilizada para ajustar os valores de intensidade dos pixels capturados pela câmera. Essa técnica compensa variações de iluminação, melhorando a detecção de cores. A aplicação prática dessa técnica no sistema foi baseada na geração de uma tabela de mapeamento (*lookup table*) que transforma os valores dos pixels (POYNTON, 1993).

3.2.2 Rastreamento de Cores

O rastreamento de cores foi implementado utilizando o modelo RGB (Red, Green, Blue), que representa as cores como uma combinação de intensidades dos três canais primários. Essa escolha foi feita pela simplicidade do modelo e pela sua compatibilidade direta com os dados capturados pela câmera, eliminando a necessidade de conversões adicionais.

3.2.3 Projeção Tridimensional

As coordenadas extraídas pelo módulo de rastreamento foram transformadas para o espaço tridimensional utilizando projeção perspectiva. O OpenGL gerencia essa transformação com matrizes de projeção, permitindo a renderização precisa dos objetos rastreados (SHREINER et al., 2018).

3.2.4 Renderização e Interatividade

A renderização gráfica foi implementada utilizando blending para criar transições visuais suaves e buffers de profundidade para gerenciar a ordem dos objetos no espaço tridimensional. A interação do usuário foi configurada por meio de callbacks no GLUT, permitindo controle em tempo real da câmera e da visualização do ambiente (FreeGLUT Development Team,).

3.3 Visão Geral do Sistema

O sistema desenvolvido possui duas partes principais: o módulo de processamento de imagens, responsável por capturar e processar os dados de entrada, e o módulo de renderização gráfica, que utiliza esses dados para atualizar o ambiente tridimensional. A integração entre esses dois módulos é feita por meio de multithreading, garantindo a execução simultânea e eficiente das tarefas.

A Figura 2 apresenta uma visão geral do sistema. À direita, temos o módulo de entrada, representado pela câmera, que captura imagens em tempo real. Essas imagens são processadas pelo módulo de processamento de imagens, utilizando a biblioteca OpenCV, para detectar e rastrear objetos com base em suas cores.

À esquerda, é exibido o ambiente tridimensional gerado pelo módulo de renderização gráfica, que utiliza os dados fornecidos pelo módulo de processamento para atualizar a posição, profundidade e visualização dos objetos. A interação entre esses módulos ocorre de forma simultânea e sincronizada por meio de programação multithreading, garantindo que os dados capturados sejam imediatamente refletidos no ambiente 3D.

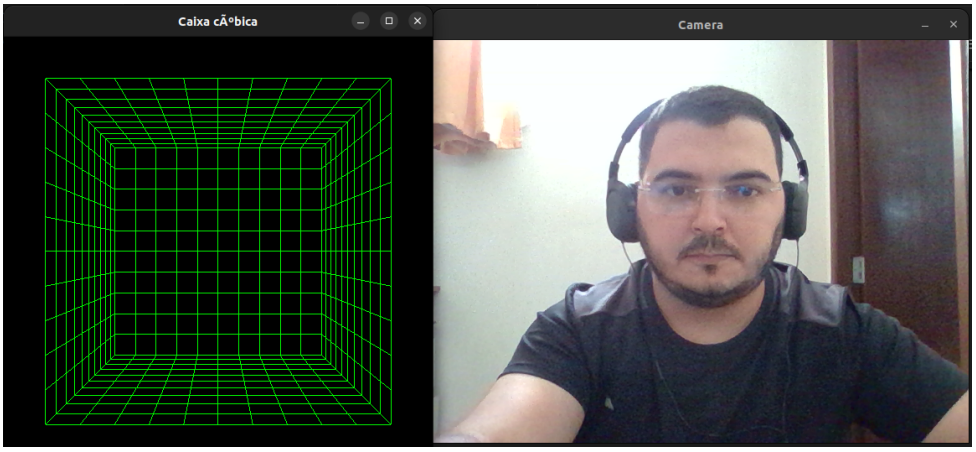


Figura 2 – Visão geral do sistema. À direita, o módulo de captura de imagens; à esquerda, o ambiente tridimensional gerado.

Essa estrutura modular e integrada permite ao sistema operar em tempo real, oferecendo uma experiência interativa em que o usuário pode visualizar e manipular os objetos no ambiente tridimensional com base nas interações detectadas pela câmera.

3.4 Configuração Inicial

A primeira etapa do sistema é a configuração da câmera e do ambiente gráfico. O módulo OpenCV configura a captura de vídeo, ajustando parâmetros como resolução e exposição, enquanto o módulo OpenGL inicializa o ambiente tridimensional. Essa etapa é fundamental para garantir que os dados capturados pela câmera sejam corretamente utilizados no ambiente gráfico. Cada módulo é um arquivo cpp diferente, que é inserido chamando a função principal de cada módulo num arquivo central que inicia cada um e aplica a thread necessária para rodar ambos em paralelo.

3.4.1 Configuração da Câmera

O OpenCV é configurado para capturar imagens em tempo real. São ajustados parâmetros como:

- **Resolução**: Definida como 640×480 pixels para equilibrar qualidade e desempenho.
- **Exposição**: Ajustada manualmente para melhorar a detecção de cores, já que algumas câmeras tem um ajuste automático de foco e balanco de brilo.

Listing 3.1 – Configuração da câmera com OpenCV.

```
VideoCapture cap(0);
cap.set(CAP_PROP_FRAME_WIDTH, 640);
cap.set(CAP_PROP_FRAME_HEIGHT, 480);
cap.set(CAP_PROP_AUTO_EXPOSURE, 0);
cap.set(CAP_PROP_EXPOSURE, 100);
```

3.4.2 Inicialização do OpenGL

O ambiente gráfico é configurado para renderizar uma caixa cúbica com dimensões ajustáveis e interatividade. Para isso, diversas funções da biblioteca OpenGL são utilizadas para configurar o ambiente de visualização e habilitar recursos gráficos essenciais.

A seguir, são descritas as principais funções empregadas no processo de inicialização:

- `glutInit`: Inicializa a biblioteca GLUT, configurando os parâmetros básicos necessários para a criação da janela gráfica.
- `glutInitDisplayMode`: Define o modo de exibição, como buffers de cor e profundidade. Neste caso, utiliza-se o modo `GLUT_SINGLE` (buffer único) e `GLUT_RGB` (cores RGB).
- `glutInitWindowSize` e `glutCreateWindow`: Configuram o tamanho da janela gráfica e criam a janela com um título especificado.
- `glEnable` e `glBlendFunc`: Habilitam o recurso de blending (transparência) e definem a função de mistura de cores para criar efeitos visuais suaves.
- `glClearColor`: Define a cor de fundo do ambiente gráfico.
- `glMatrixMode` e `gluPerspective`: Configuram a matriz de projeção para criar uma visualização 3D com perspectiva.
- `glutDisplayFunc`, `glutIdleFunc`, `glutMouseFunc` e `glutMotionFunc`: Associam funções de callback para a exibição, atualização e interatividade (como controle do mouse).

A inicialização completa é realizada na função `initOpenGL`, cujo código é apresentado a seguir:

Listing 3.2 – Inicialização do ambiente gráfico com OpenGL.

```
void initOpenGL(int argc, char **argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(512, 512);
    glutCreateWindow("Caixa cubica");
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glShadeModel(GL_FLAT);
    glEnable(GL_DEPTH_TEST);
    glMatrixMode(GL_PROJECTION);
    gluPerspective(45.0, 1.0, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
    glutDisplayFunc(display);
    glutIdleFunc(updateCubePosition);
    glutMouseFunc(mouseButton);
    glutMotionFunc(mouseMotion);
    glutMainLoop();
}
```

Essas configurações garantem que o ambiente gráfico seja inicializado corretamente, com suporte à renderização tridimensional, blending para efeitos visuais e interação do usuário por meio de dispositivos de entrada, como o mouse. A estrutura modular dessa função facilita futuras expansões ou ajustes no sistema.

3.5 Rastreamento de Cores

O rastreamento de cores é realizado pelo módulo OpenCV, que processa as imagens capturadas em tempo real para identificar objetos com base em suas cores. Essa etapa é fundamental para o funcionamento do sistema e envolve os seguintes passos principais:

- **Captura da imagem em tempo real**: A câmera captura frames continuamente, que são processados pelo sistema.
- **Correção de gama**: Aplica-se um ajuste nos valores de intensidade dos pixels para melhorar a uniformidade da iluminação e a consistência na detecção de cores.
- **Criação de máscaras binárias**: Isolam-se os pixels que correspondem às faixas de cor definidas.
- **Extração de contornos e cálculo do centroide**: A partir da máscara, os contornos dos objetos rastreados são identificados. Com base no maior contorno encontrado, calcula-se o centroide e o raio do círculo mínimo que engloba o objeto.

A Figura 3 ilustra o processo de rastreamento para um objeto azul, destacando as etapas envolvidas:

1. **Imagem capturada**: O frame original capturado pela câmera.
2. **Imagem com correção de gama** : O mesmo frame após a aplicação da correção de gama, que uniformiza a iluminação, deixando o controle em destaque.
3. **Máscara binária**: Os pixels da cor azul rastreada são isolados, criando uma máscara binária.
4. **Detecção de contornos**: O maior contorno é identificado, e o menor círculo que o engloba é calculado.

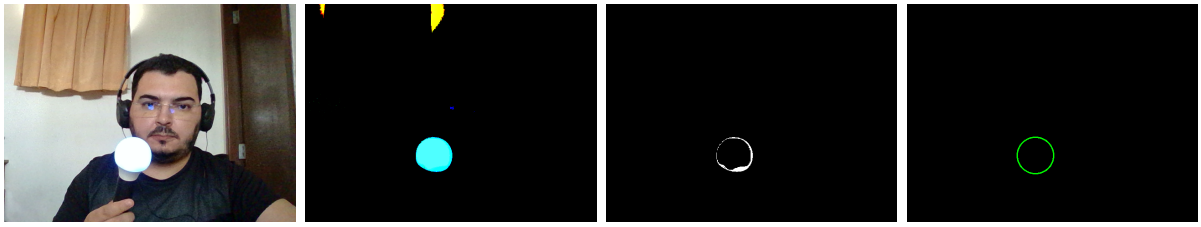


Figura 3 – Processo de rastreamento de cores (azul) com OpenCV: da captura à detecção de contornos.

A Figura 4 apresenta o mesmo processo aplicado para um objeto vermelho. As etapas são idênticas às do rastreamento do objeto azul, mas com uma máscara que isola a cor

vermelha. Isso demonstra a flexibilidade do sistema em rastrear múltiplas cores com a mesma lógica de processamento.

Figura 4 – Processo de rastreamento de cores (vermelho) com OpenCV: da captura à detecção de contornos.

Essas figuras ilustram como o sistema utiliza a captura de imagens em tempo real e o processamento com OpenCV para rastrear objetos de diferentes cores. A análise de cores é realizada separadamente para cada faixa de cor definida, permitindo que múltiplos objetos sejam rastreados e diferenciados no ambiente tridimensional.

3.5.1 Máscaras de Cor

As máscaras de cor são geradas pelas cores rastreadas, utilizando a função `inRange`, que isola pixels dentro de uma faixa de cor específica. Após a aplicação das máscaras, o cálculo do centroide dos objetos rastreados é realizado com a função `findContours`.

Duas máscaras são criadas simultaneamente:

- `maskColor1`: Detecta pixels da primeira cor rastreada com base nos valores médios e desvio padrão calculados durante a fase de configuração.
- `maskColor2`: Detecta pixels da segunda cor rastreada, usando a mesma abordagem da `maskColor1`.

A seguir, o código atualizado para a geração das máscaras e cálculo dos contornos é apresentado:

Listing 3.3 – Detecção de cores e criação de máscaras para duas cores.

```
inRange(gammaImage, average1 - stdev1, average1 + stdev1, maskColor1);
inRange(gammaImage, average2 - stdev2, average2 + stdev2, maskColor2);
```

```
if (countNonZero(maskColor1) > 40) {
    currentMask = maskColor1;
    drawing = true;
}

if (countNonZero(maskColor2) > 40) {
    currentMask = maskColor2;
    drawing = false;
}

vector<vector<Point>> contours;
findContours(currentMask, contours, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);
```

O sistema seleciona a máscara ativa com base no número de pixels detectados acima do limiar (`countNonZero`). A máscara ativa é utilizada para extrair os contornos do objeto rastreado e calcular o centroide, que será transformado em coordenadas tridimensionais. Essa abordagem permite o rastreamento simultâneo de dois objetos distintos, com comportamentos diferenciados definidos pelo parâmetro `drawing`.

3.6 Cálculo da Profundidade

A profundidade (*eixo Z*) dos objetos rastreados no sistema é determinada a partir de informações obtidas durante o processamento das imagens capturadas pela câmera. Essa etapa utiliza técnicas de processamento de contornos disponíveis no OpenCV para identificar e caracterizar os objetos detectados.

Primeiramente, após a detecção de um objeto na imagem, o sistema extrai seu contorno. Para simplificar e regularizar a forma do contorno, é aplicada a técnica *Convex Hull*, que aproxima o contorno detectado a uma forma convexa. Essa técnica reduz irregularidades e garante que o objeto seja representado de maneira consistente, independentemente de sua forma original.

Em seguida, o sistema calcula o menor círculo que engloba o contorno do objeto utilizando as extremidades do *Convex Hull*. Esse círculo fornece duas informações funda-

mentais:

- As coordenadas (X, Y), que correspondem à posição do centro do círculo na imagem capturada.
- O raio do círculo, utilizado para definir a profundidade (*eixo Z*).

A escolha do raio como parâmetro para o *eixo Z* baseia-se na relação inversa entre o tamanho do objeto projetado na imagem e sua distância da câmera: quanto maior o objeto na imagem, menor sua distância em relação à câmera. Essa abordagem é eficiente, pois elimina a necessidade de sensores especializados, como câmeras de profundidade, utilizando apenas os dados obtidos pela câmera convencional.

Com essa abordagem, o sistema é capaz de gerar uma representação tridimensional eficiente, utilizando apenas processamento de imagem e sem a necessidade de hardware adicional. Essa simplicidade aumenta a acessibilidade do sistema, tornando-o viável para aplicações em ambientes de baixo custo.

3.7 Renderização e Interação

O módulo OpenGL utiliza as coordenadas rastreadas pelo OpenCV para posicionar objetos no ambiente 3D. As posições são atualizadas em tempo real, e o sistema permite a interação com o ambiente por meio do mouse. A variável `drawing` compartilhada pelo módulo OpenCV controla a liberação das função de desenho, nisso dependendo da cor detectada é mostrado um objeto se movimentando, mostrando a posição em que o controle está naquele momento e também quando detectada a cor 1, ativando a função de desenho, atualizando os dados de posições desenhadas.

3.7.1 Atualização de Coordenadas

Os dados do OpenCV são escalados para o sistema de coordenadas do OpenGL. Essa conversão garante que as posições rastreadas sejam corretamente mapeadas para o ambiente tridimensional, mesmo com diferentes dimensões do ambiente.

As coordenadas X, Y e Z são ajustadas para o intervalo tridimensional $[-1, 1]$ utilizando a função `escalarValor`, que normaliza os valores com base no mínimo e no máximo de cada eixo. O código atualizado é apresentado a seguir:

Listing 3.4 – Atualização das coordenadas com OpenGL.

```
double escalarValor(
  double x,
  double xMin,
  double xMax,
  double step = 0.2)
  {
    double normalizedX = (x - xMin) / (xMax - xMin);
    double scaledValue = -1 + normalizedX * 2;
    scaledValue = round(scaledValue / step) * step;
    if (scaledValue < -1) scaledValue = -1;
    if (scaledValue > 1) scaledValue = 1;
    return scaledValue;
}

void updateCubePosition() {
    if (configStep != CONFIGEXIT) return;

    lock_guard<mutex> lock(cubeMutex);

    float newX = escalarValor(matX, 0, NY, cubeSize);
    float newY = -escalarValor(matY, 0, NX, cubeSize);
    float newZ = escalarValor(matZ, zMin, zMax, cubeSize);

    if (!positionVisited(newX, newY, newZ)) {
        cubeX = newX;
        cubeY = newY;
        cubeZ = newZ;
        lock_guard<mutex> posLock(positionMutex);
```

```
        if (drawing)
            visitedPositions.push_back(make_tuple(cubeX, cubeY, cubeZ));
    }

    glutPostRedisplay();
}
```

3.7.2 Interação com o Usuário

O sistema permite que o usuário interaja com o ambiente tridimensional rotacionando a câmera com o mouse. As funções de callback do OpenGL capturam os movimentos do mouse e atualizam os ângulos de rotação em tempo real.

Listing 3.5 – Interação com o ambiente 3D.

```
void mouseButton(int button, int state, int x, int y) {
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {
        lastMouseX = x;
        lastMouseY = y;
    }
}

void mouseMotion(int x, int y) {
    if (lastMouseX >= 0 && lastMouseY >= 0) {
        int dx = x - lastMouseX;
        int dy = y - lastMouseY;

        rotationX += dy * 0.1;
        rotationY += dx * 0.1;

        lastMouseX = x;
        lastMouseY = y;
    }
    glutPostRedisplay();
```

```
}
```

3.8 Integração por Multithreading

Para que o OpenCV e o OpenGL operem simultaneamente, cada módulo é executado em uma thread separada. A sincronização entre threads é feita utilizando `std::mutex`, garantindo que os dados compartilhados sejam acessados de forma consistente.

O código atualizado para a integração dos dois módulos é apresentado abaixo:

Listing 3.6 – Configuração de threads para sincronização.

```
int main(int argc, char **argv) {
    thread opencvThread(processOpenCV);
    initOpenGL(argc, argv);
    opencvThread.join();
    return 0;
}
```

As threads são configuradas de forma que o OpenCV processe os dados de rastreamento de cores enquanto o OpenGL atualiza o ambiente gráfico. A função `std::mutex` é utilizada para evitar condições de corrida (*race conditions*) durante a leitura e escrita das variáveis compartilhadas.

3.9 Configuração Inicial do funcionamento

A etapa de configuração inicial do sistema é essencial para garantir o funcionamento adequado do rastreamento de cores e da interação tridimensional. Durante essa etapa, são definidos os seguintes parâmetros principais:

- **Cores Rastreadas**: O sistema permite configurar duas cores específicas, que serão detectadas e rastreadas durante a execução. Essa configuração é feita com base em amostras capturadas pela câmera.
- **Faixas de Coordenadas** Z: São definidas as coordenadas mínimas e máximas para a profundidade dos objetos no espaço tridimensional (`zMin` e `zMax`), calibrando o ambiente 3D.

(a) Definição de cor 1

(b) Definição de cor 2

(c) Definição de Z máximo

(d) Definição de Z mínimo

Figura 5 – Processo de configuração inicial do sistema.

3.9.1 Etapas de Configuração

O processo de configuração, como é possível ver na Figura 5, é realizado por etapas interativas, onde é mostrado ao usuário instruções de onde posicionar o controle e o que fazer naquela etapa para que ao final o rastreamento esteja pronto para ser executado da maneira mais eficaz.

Todo esse processo inicial é feito pela função *configureColors* que define as mensagens mostradas e a que passo está naquele momento dado um temporizador, cada etapa mostra um conteúdo diferente na tela e enquanto isso o módulo OpenCV trabalha definindo os dados necessários para o funcionamento.

Essa etapa é crucial para adaptar o sistema a diferentes cenários e condições de uso, garantindo que a detecção de cores e a renderização gráfica sejam realizadas de maneira eficiente e precisa.

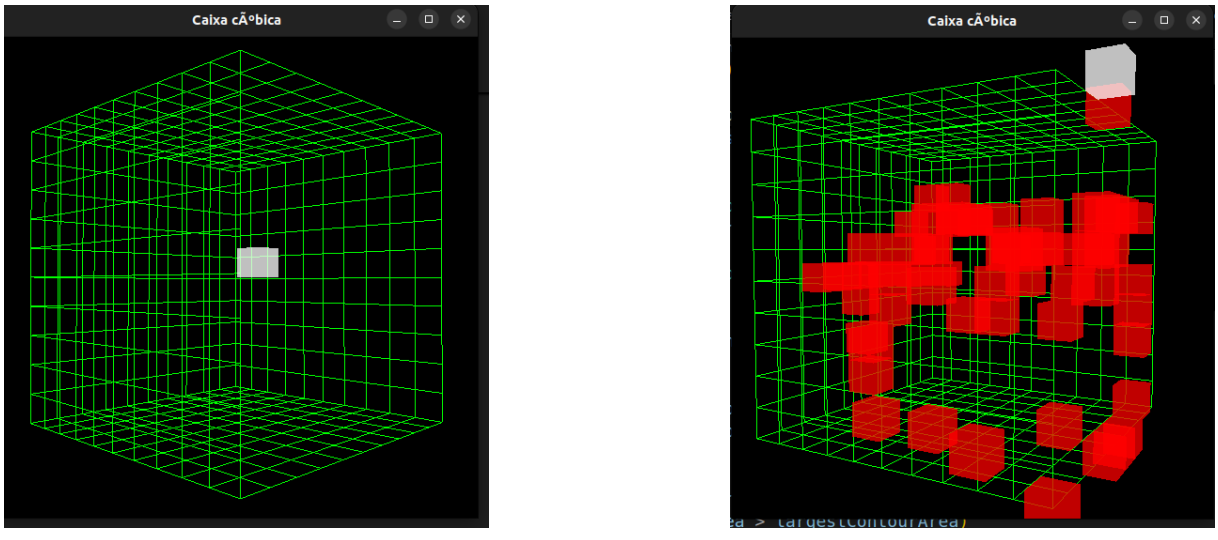
4 RESULTADOS OBTIDOS

Neste capítulo, são apresentados os resultados obtidos com a implementação do sistema, incluindo a análise do desempenho em rastreamento de cores, renderização gráfica e integração entre OpenCV e OpenGL. Além disso, são discutidos os sucessos alcançados, as limitações identificadas e as melhorias possíveis.

4.1 Análise Geral do Sistema

O sistema desenvolvido atingiu os objetivos propostos, demonstrando a capacidade de rastrear cores em tempo real, calcular coordenadas tridimensionais e renderizar interativamente objetos em um ambiente gráfico. Os testes realizados mostraram que o sistema é funcional e eficiente em condições controladas, apresentando um desempenho estável na maioria dos cenários.

O ambiente gráfico foi renderizado com sucesso utilizando OpenGL, enquanto o módulo de visão computacional baseado em OpenCV capturou imagens em tempo real e processou as máscaras de cor para extrair informações espaciais. A integração dos módulos realizada utilizando multithreading, permitiu a execução simultânea das tarefas de rastreamento e renderização.



(a) Ambiente gráfico sem interação (b) Ambiente gráfico com interação

Figura 6 – Rastreamento e plotagem gráfica

Como podemos ver na Figura 6, à esquerda o ambiente gráfico no seu modo inicial, onde nenhum dado foi rastreado e mapeado até o momento para o ambiente. Já na direita temos o ambiente após o teste, mostrando que o objeto foi rastreado e os dados foram renderizados de acordo com o movimento feito dentro do ambiente tridimensional.

4.2 Testes de Rastreamento de Cores

Foram realizados testes com objetos de cores distintas em diferentes condições de iluminação. O sistema demonstrou uma boa precisão ao rastrear cores com tons bem definidos e contraste adequado em relação ao fundo.

4.2.1 Resultados Obtidos

O sistema foi capaz de rastrear duas cores distintas em tempo real, com transição suave entre as máscaras de cor ativas. As métricas observadas incluem:

- **Precisão do Rastreamento**: A precisão em condições controladas, onde não tem outras fontes de brilho foi satisfatória, o controle é rastreado corretamente e conseguimos fazer o desenho de maneira fluida.
- **Desempenho em Ambientes Não Controlados**: A precisão caiu em cenários com iluminação variável, como o gamma na configuração isola esses pontos brilhosos, nesses tipos de ambientes fica um pouco difícil fazer a configuração das cores e consequentemente do rastreamento.

4.2.2 Transições entre Máscaras

O sistema alterna entre duas máscaras de cor (`maskColor1` e `maskColor2`) com base na contagem de pixels detectados. Isso foi particularmente eficaz para permitir a identificação de diferentes objetos no mesmo espaço. A lógica implementada garantiu que apenas uma máscara estivesse ativa a cada instante, minimizando interferências.

4.3 Limitações Identificadas

Apesar dos sucessos obtidos, algumas limitações foram observadas:

- **Sensibilidade à Iluminação**: O sistema apresentou dificuldades em cenários com iluminação variável, resultando em detecções inconsistentes, durante a definicção das cores, em um ambiente muito iluminado, o controle(luzes) ficam difíceis de diferenciar, ocasionando uma mal definição da cor de rastreamento.
- **Ruído em Fundos Complexos**: Em ambientes com fundos detalhados, as máscaras geradas incluíram ruídos, o que prejudicou a precisão do rastreamento.
- **Desempenho em Cenários Complexos**: A renderização gráfica foi impactada em situações de alta densidade de objetos, brilhos externos como janelas ou mudanças rápidas de posição.

4.4 Discussão dos Resultados

Os resultados demonstram que a integração entre OpenCV e OpenGL foi bem-sucedida para a proposta do trabalho. A capacidade de rastrear cores e atualizar objetos em tempo real destaca o potencial desse sistema para aplicações interativas. No entanto, as limitações apontam para áreas de melhoria, como:

- Aplicação de algoritmos mais robustos para pré-processamento de imagens e remoção de ruídos.
- Implementação de estratégias de otimização gráfica para aumentar a taxa de quadros em cenários complexos.
- Ajustes nos parâmetros de detecção de cores para melhorar a robustez em diferentes condições de iluminação.

4.5 Resumo

O sistema atingiu seus objetivos principais, demonstrando a eficácia do uso de tecnologias acessíveis para criar um ambiente interativo em 3D. As limitações enfrentadas são desafios comuns em sistemas de visão computacional e computação gráfica, mas também indicam oportunidades para trabalhos futuros.

5 CONCLUSÃO

Neste capítulo, são apresentadas as conclusões gerais do trabalho, as limitações identificadas, as contribuições realizadas e sugestões para trabalhos futuros.

5.1 Conclusões Gerais

O desenvolvimento deste trabalho demonstrou a viabilidade de integrar as bibliotecas OpenCV e OpenGL para criar uma aplicação interativa de rastreamento de cores e visualização tridimensional. O sistema foi capaz de capturar e processar imagens em tempo real, rastrear objetos coloridos e renderizar suas posições em um ambiente tridimensional com interatividade. Os testes realizados confirmaram que, em condições controladas, o sistema apresenta um bom desempenho, sendo capaz de atualizar a visualização gráfica de forma estável.

Este projeto mostrou como tecnologias acessíveis e bem documentadas podem ser combinadas para criar soluções inovadoras, mesmo com recursos limitados. A abordagem proposta destaca-se pela simplicidade de implementação e potencial para aplicações interativas em áreas como educação, jogos e simulações.

5.2 Limitações

Apesar dos resultados positivos, algumas limitações foram identificadas ao longo do desenvolvimento:

- **Sensibilidade à Iluminação**: O rastreamento de cores é impactado por variações na iluminação ambiente, o que reduz a precisão em cenários não controlados.
- **Ruído em Fundos Complexos**: Em cenas com fundos complexos ou com baixa distinção de contraste, o sistema apresentou dificuldades na detecção precisa.
- **Desempenho em Cenários Complexos**: O desempenho gráfico foi afetado em situações com muitos objetos rastreados, evidenciando a necessidade de otimizações

no processamento.

5.3 Contribuições

Este trabalho contribuiu para o avanço da integração entre visão computacional e computação gráfica, oferecendo uma solução que:

- Implementa uma aplicação acessível para rastreamento de cores e visualização 3D.
- Demonstra a eficácia do uso de multithreading para sincronização entre OpenCV e OpenGL.
- Fornece uma base para o desenvolvimento de sistemas interativos em diversas áreas.

5.4 Trabalhos Futuros

Com base nas limitações identificadas, várias melhorias podem ser exploradas em trabalhos futuros:

- **Uso de Algoritmos Avançados de Rastreamento**: Implementar técnicas mais robustas, como redes neurais ou rastreamento baseado em características, para melhorar a precisão em condições de iluminação variada e fundos complexos.
- **Otimização do Desempenho**: Explorar o uso de APIs gráficas modernas, como Vulkan, para melhorar a eficiência da renderização.
- **Expansão da Interatividade**: Adicionar novos tipos de interação, como gestos ou reconhecimento de formas, para ampliar as possibilidades do sistema.
- **Integração com Dispositivos de Entrada**: Incorporar dispositivos como sensores de movimento ou câmeras de profundidade para aumentar a flexibilidade do sistema.

5.5 Considerações Finais

Este trabalho serviu como uma importante experiência de aprendizado, permitindo o aprofundamento nos conceitos de visão computacional, computação gráfica e integração

de tecnologias. As limitações enfrentadas não apenas destacam áreas de melhoria, mas também abrem caminhos para novas ideias e implementações futuras. A aplicação proposta representa uma base sólida para a exploração de sistemas interativos que combinam percepção visual e visualização gráfica em tempo real.

O código-fonte do sistema está disponível publicamente no Github (LIMA, 2024), permitindo que futuros pesquisadores e desenvolvedores utilizem e expandam a solução proposta.

REFERÊNCIAS

FreeGLUT Development Team. *FreeGLUT Documentation*. http://freeglut.sourceforge.net/docs.html. Acesso em: 05 set. 2024.

GNU Project. *GCC Documentation*. https://gcc.gnu.org/onlinedocs/. Acesso em: 04 set. 2024.

IZADI, S. et al. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In: ACM. *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. 2011. p. 559–568. Acesso em: 16 dez. 2024. Disponível em: <https://www.microsoft.com/en-us/research/wp-content-/uploads/2016/02/ismar2011.pdf>https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/ismar2011.pdf.

IZADI, S. et al. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera (uist companion). In: ACM. *UIST '11 Adjunct Proceedings*. 2011. Acesso em: 16 dez. 2024. Disponível em: <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/kinectfusion-uist-comp.pdf>https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/kinectfusion-uist-comp.pdf.

Khronos Group. *OpenGL Documentation*. https://www.opengl.org/documentation/. Acesso em: 04 set. 2024.

LIMA, L. de A. *Sistema de Rastreamento de Cores e Renderização 3D*. 2024. https://github.com/LucasAzLima/ColorTrackingVisualizer. Acesso em: 14 jan. 2025.

Microsoft. *Visual Studio Code Documentation*. https://code.visualstudio.com/docs. Acesso em: 23 dez. 2024.

OpenCV Team. *OpenCV Documentation*. https://docs.opencv.org/. Acesso em: 18 set. 2024.

PERL, T. *PS Move API: Open Source library for working with the PlayStation Move Motion Controller*. 2010. https://github.com/thp/psmoveapi. Acesso em: 15 nov. 2024.

POYNTON, C. A. Gamma and its importance in video and computer graphics. *SMPTE Journal*, Society of Motion Picture and Television Engineers, v. 102, n. 12, p. 1099–1108, 1993.

Qubacy. *Move & Draw*. 2024. https://play.google.com/store/apps/details?id=com.qubacy.moveanddraw. Acesso em: 01 out. 2024.

SHREINER, D. et al. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.5 with SPIR-V*. 9th. ed. [S.l.]: Addison-Wesley, 2018.

Sony Computer Entertainment. *Move.me Available Today on PlayStation Store, Free for Students and Educators*. 2011. https://blog.playstation.com/2011/07/26/move-me-available-today-on-playstation-store-free-for-students-and-educators/. Acesso em: 16 nov. 2024.

WILLIAMS, A. *C++ Concurrency in Action: Practical Multithreading*. 2nd. ed. [S.l.]: Manning Publications, 2019.