UNIVERSIDADE FEDERAL DE GOIÁS / INSTITUTO DE INFORMÁTICA

# Visão 3D para Tecnologias Imersivas

Uma abordagem inovadora de geração de ambientes dentro do Meta Quest 3

RODRIGO MENDES DE CARVALHO

UFG
UNIVERSIDADE FEDERAL DE GOIÁS

UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
INSTITUTO DE INFORMÁTICA (INF)

RODRIGO MENDES DE CARVALHO

# VISÃO 3D PARA TECNOLOGIAS IMERSIVAS

Uma abordagem inovadora de geração de ambientes dentro do Meta Quest 3

Goiânia
2024

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

# TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

**1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)**

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): **RODRIGO MENDES DE CARVALHO**

Título do trabalho: **Visão 3D para Tecnologias Imersivas**
**Uma abordagem inovadora de geração de ambientes dentro do Meta Quest 3**

**2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento** [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

**Casos de embargo:**

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

**Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.**

Documento assinado eletronicamente por **Rodrigo Mendes De Carvalho**, **Discente**, em 04/08/2024, às 15:29, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Fernando Marques Federson**, **Professor do Magistério Superior**, em 12/09/2024, às 10:54, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4713863** e o código CRC **D2197B7C**.

**Referência:** Processo nº 23070.037995/2024-11 SEI nº 4713863

RODRIGO MENDES DE CARVALHO

# VISÃO 3D PARA TECNOLOGIAS IMERSIVAS

Uma abordagem inovadora de geração de ambientes dentro do Meta Quest 3

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.
Orientador: Prof. Dr. Fernando Marques Federson

Goiânia
2024

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

CARVALHO, RODRIGO MENDES DE
VISÃO 3D PARA TECNOLOGIAS IMERSIVAS [manuscrito] : Uma abordagem inovadora de geração de ambientes dentro do Meta Quest 3 / RODRIGO MENDES DE CARVALHO. - 2024.
240 f.

Orientador: Prof. FERNANDO MARQUES FEDERSON.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Instituto de Informática (INF), Inteligência Artificial, Goiânia, 2024.
Apêndice.
Inclui gráfico, tabelas, algoritmos.

1. inteligência artificial. 2. visão computacional. 3. tecnologias imersivas. I. FEDERSON, FERNANDO MARQUES, orient. II. Título.

CDU 004

RODRIGO MENDES DE CARVALHO

# VISÃO 3D PARA TECNOLOGIAS IMERSIVAS

Uma abordagem inovadora de geração de ambientes dentro do Meta Quest 3

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Data da Aprovação: 06 de agosto de 2024.

gov.br Documento assinado digitalmente
FERNANDO MARQUES FEDERSON
Data: 06/08/2024 13:35:56-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Fernando Marques Federson
Orientador (INF-UFG)

gov.br Documento assinado digitalmente
ALDO ANDRE DIAZ SALAZAR
Data: 09/08/2024 18:14:40-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Aldo André Díaz Salazar
Coordenador de TCC do BIA (INF-UFG)

gov.br Documento assinado digitalmente
ANDERSON DA SILVA SOARES
Data: 11/08/2024 20:49:00-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Anderson da Silva Soares
Coordenador do BIA (INF-UFG)

gov.br Documento assinado digitalmente
IWENS GERVASIO SENE JUNIOR
Data: 09/08/2024 11:15:29-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Iwens Gervasio Sene Junior
(INF-UFG)

RODRIGO MENDES DE CARVALHO

# VISÃO 3D PARA TECNOLOGIAS IMERSIVAS

Uma abordagem inovadora de geração de ambientes dentro do Meta Quest 3


## RESUMO

Este Relatório de Conclusão de Curso tem como objetivo reunir os resultados da minha jornada para me tornar um especialista em **Visão Computacional**. Uma ilustração e sua narrativa descrevem os períodos de trabalho. Os Apêndices contêm os Termos de Aceite de Entrega e os resultados obtidos durante cada período de trabalho.


Palavras-chave: inteligência artificial, visão computacional, tecnologias imersivas.


## ABSTRACT

This Course Completion Report aims to bring together the results of my journey to become an expert in **Computational Vision**. An illustration and its narrative describe the work periods. The Appendices contain the Delivery Acceptance Terms and the results obtained during each work period.


Keywords: artificial intelligence, computer vision, immersive technologies.

Goiânia

2024

# Minha Jornada

Rodrigo Mendes de Carvalho

Especialista em:

Visão 3D aplicada a Tecnologias Imersivas

# MINHA JORNADA

## Nome: Rodrigo Mendes de Carvalho

## Especialidade: Visão 3D aplicada a Tecnologias Imersivas

## Objetivo deste documento

Durante o processo da disciplina Residência em IA[1], foram gerados diversos resultados na construção da minha especialização. A cada semana, um conjunto de resultados foi formalizado por um Termo de Aceite de Entrega e avaliado por uma banca, considerando o planejado e o realizado para o período. Este documento tem como objetivo descrever esses resultados obtidos, fazendo referência aos Termos de Aceite de Entrega e seus documentos associados.

## Minha Jornada

Minha jornada na Residência em IA começou com a definição da área de atuação e o mapeamento dos objetivos que seriam abordados ao longo do processo. Durante a Semana 1, dediquei-me a entender melhor os temas definidos pela *Conference on Computational Science and Computational Intelligence* (CSCI'23), escolhendo focar em redes neurais e suas aplicações, tecnologias emergentes e aplicações práticas em visão computacional, processamento de sinais e grandes modelos de linguagem. Este primeiro passo foi crucial para estabelecer uma base sólida para o meu projeto, permitindo alinhar meus objetivos com as direções mais promissoras da pesquisa atual.

Para formalizar essa etapa inicial, realizei a leitura e o aprofundamento nas áreas de *Research Track on Signal & Image Processing*, *Computer Vision & Pattern Recognit*ion (CSCI-RTPC). A análise detalhada desses documentos de revisão forneceu uma compreensão abrangente dos tópicos mais relevantes e emergentes nessas áreas. Compilando e organizando esses materiais, criei um repositório online para armazenar todos os documentos e scripts relacionados ao projeto. Essa organização não só facilitou meu acesso aos recursos durante a pesquisa, mas também permitiu um compartilhamento mais eficiente de informações com meus colegas e orientadores.

Na Semana 2, continuei a desenvolver a infraestrutura necessária para o projeto, gerando uma organização no GitHub onde depositaria todos os documentos e scripts relevantes. Além disso, comecei a levantar artigos científicos e livros focados no processamento de sinais, um dos temas centrais do meu estudo. Estudar esses tópicos iniciais levantados na CSCI'23, especialmente em áreas como identificação de sinais, reconstrução de sinal e análise espectral, foi essencial para construir um conhecimento técnico aprofundado que

[1] Onze semanas, entre abril de 2024 e julho de 2024.

seria aplicado nas etapas seguintes do projeto. Os documentos referentes às Semanas 1 e 2 podem ser encontrados no Apêndice 1.

Durante as Semanas 3 e 4, meu foco principal foi aprofundar meu conhecimento nas áreas de Processamento de Sinais e Visão Computacional. Na Semana 3, comecei a explorar as ferramentas de Processamento de Sinais mencionadas no GATE do dia 17 de abril. Este processo envolveu o uso de *notebooks* no GitHub, onde documentei meus experimentos e observações. Através desse estudo, consegui levantar uma série de artigos e livros sobre tópicos como redes de câmeras, sensores e tecnologias de Aprendizado de Máquina aplicadas à Visão Computacional. Este levantamento inicial de recursos foi crucial para formar uma base sólida de conhecimentos práticos e teóricos.

A entrega desta Semana incluiu um resumo dos artigos e tópicos levantados, os quais foram documentados e organizados no meu repositório do GitHub. A criação desse repositório serviu não apenas para armazenar as informações coletadas, mas também para facilitar o compartilhamento e a colaboração com outros pesquisadores e membros da equipe. Cada documento e código foi revisado e preparado para futuras referências e estudos adicionais. Este método estruturado de coleta e organização de dados garantiu que todos os recursos estivessem facilmente acessíveis e bem catalogados.

Na Semana 4, continuei a aprofundar meus estudos, agora focando especificamente nas ferramentas de Visão Computacional destacadas no GATE do dia 24 de abril. Dediquei-me a explorar tecnologias e metodologias relacionadas ao Reconhecimento de Padrões, incluindo algoritmos de Classificação supervisionada e não supervisionada, técnicas de Clusterização, métodos de Redução de dimensionalidade, Aprendizagem simbólica e algoritmos de Aprendizagem ensemble. Esta fase de estudos foi documentada detalhadamente no GitHub, com cada recurso sendo resumido e categorizado conforme sua relevância para os objetivos do projeto.

Durante essa Semana, também iniciei o mapeamento das áreas, ferramentas e frameworks de Visão 3D. Este mapeamento envolveu a identificação dos principais componentes e suas interações, criando uma visão clara e organizada da área de Visão 3D. A documentação deste mapeamento foi crucial para estruturar as próximas etapas do projeto, permitindo um planejamento detalhado das atividades futuras. Para a próxima entrega, planejei levantar mais artigos, livros e estudos focados em Visão 3D, preparando-me para aprofundar ainda mais nesse campo específico da Inteligência Artificial. Os documentos referentes às Semanas 3 e 4 podem ser encontrados no Apêndice 2.

Nas Semanas 5 e 6, meu foco principal foi levantar artigos, livros e estudos sobre o tópico de Visão 3D, aprofundando meus conhecimentos nesse campo essencial para a Tecnologia imersiva. Durante a Semana 5, dediquei-me a uma extensa pesquisa bibliográfica, compilando recursos relevantes que cobrem diversos aspectos da Visão 3D, desde fundamentos teóricos até aplicações práticas em Tecnologias imersivas. Esta pesquisa foi

documentada no repositório do GitHub, onde resumi os artigos e estudos levantados, facilitando o acesso e a consulta para futuras referências.

A entrega desta Semana incluiu um conjunto detalhado de resumos de artigos e tópicos levantados, organizados de maneira sistemática no meu repositório online. Através desta Revisão literária, consegui identificar as principais tendências e desafios na área de Visão 3D, preparando-me para as implementações práticas que se seguiram.

Durante a Semana 6, avancei para a exploração de ferramentas específicas e tecnologias aplicadas à Visão 3D. Este período foi marcado pela exploração do Meta Quest 3, uma plataforma crucial para experiências imersivas. Testei e documentei várias ferramentas, incluindo as técnicas NERF (*Neural Radiance Fields*) e a rede SAGNet, que são fundamentais para a criação de ambientes tridimensionais realistas. Cada experimento foi cuidadosamente documentado no GitHub, incluindo códigos, vídeos e observações detalhadas. Os documentos referentes às Semanas 5 e 6 podem ser encontrados no Apêndice 3.

Na Semana 7, meu foco principal foi o desenvolvimento de uma plataforma online que pudesse gerar texto a partir de áudio, utilizando ferramentas como *pydub* e *pyngrok*. Este projeto visava permitir que os usuários falassem ao microfone, gerando um prompt que seria posteriormente processado pelo modelo generativo *SD-T2I-360PanoImage* para criar imagens panorâmicas 360 graus. Este processo não apenas explorou novas formas de interação com a tecnologia, mas também demonstrou a capacidade de integrar diferentes ferramentas e técnicas para criar uma experiência de usuário imersiva e funcional.

Para concretizar esse projeto, desenvolvi um código no Google Colab que facilitou a geração de imagens 360 a partir de áudio. Esta plataforma foi testada e documentada, com um exemplo disponível no link fornecido no GitHub. Além disso, criei um visualizador de imagens 360 para simular a perspectiva do VR, permitindo aos usuários visualizar as imagens geradas de uma maneira mais envolvente e realista. Esta ferramenta foi fundamental para demonstrar o potencial das Tecnologias Imersivas quando combinadas com Processamento de Linguagem Natural e Visão Computacional.

Paralelamente, iniciei os primeiros módulos do curso *"Create with VR*" oferecido pela *Unity Learn*. Este curso foi essencial para aprofundar meu conhecimento sobre o desenvolvimento de aplicações em realidade virtual, fornecendo-me as habilidades práticas necessárias para implementar e otimizar a plataforma que estava desenvolvendo. O curso cobriu vários aspectos do desenvolvimento em VR, desde a configuração inicial até técnicas avançadas de interação e renderização. Os documentos referentes à Semana 7 podem ser encontrados no Apêndice 4.

Na Semana 8, o foco principal foi a criação e integração de uma nova versão do site, que agora inclui um visualizador de imagens 360. Este site foi desenvolvido para permitir que os usuários gerassem imagens panorâmicas a partir de áudio e as visualizassem diretamente na plataforma. Utilizando JavaScript para integrar as funcionalidades, consegui criar uma interface intuitiva e funcional. O processo de geração e visualização das imagens foi documentado em um vídeo exemplo, com links para o site e o repositório do GitHub onde o código está hospedado.

Além do desenvolvimento do site, elaborei um documento detalhando a estrutura da plataforma dentro do *Google Cloud Platform* (GCP). Este documento descreve como a infraestrutura foi configurada para suportar a aplicação Meta Quest 3 com Unity, proporcionando uma visão clara das interações entre os diferentes componentes do sistema. Esta documentação foi essencial para garantir que a plataforma fosse escalável e eficiente, permitindo futuras expansões e melhorias.

Também durante essa Semana, conduzi um estudo aprofundado sobre o Projeto Stable-DreamFusion. A documentação resultante incluiu detalhes técnicos e operacionais, destacando os desafios enfrentados ao tentar implementar a solução no GCP. Encontrei dificuldades ao subir uma máquina virtual no GCP para rodar o *Stable-DreamFusion*, documentando as falhas e erros encontrados tanto no ambiente de nuvem quanto localmente. Este processo de documentação foi vital para identificar áreas de melhoria e planejar as próximas etapas de implementação. Os documentos referentes à Semana 8 podem ser encontrados no Apêndice 5.

Nas Semanas 9 e 10, dediquei-me ao desenvolvimento e integração de uma API que permitisse a geração de imagens panorâmicas a partir de prompts enviados pelos usuários. Este desenvolvimento incluiu a criação de um documento detalhado explicando como utilizar a API em diferentes sistemas operacionais e a implementação de um Colab demonstrando sua funcionalidade. O link para a plataforma foi disponibilizado, permitindo que outros pudessem testar e verificar o funcionamento da API. Esta fase do projeto foi fundamental para garantir a interoperabilidade e facilidade de uso da plataforma, expandindo seu alcance e aplicabilidade.

Além da API, concluí o curso para criação de jogos e aplicativos em VR utilizando *o Unity HUB, integrando o Meta Quest Developer Hub e instalando o* Oculus Integration SDK. Este curso foi essencial para desenvolver um ambiente VR funcional, culminando na criação de um arquivo compilado do ambiente e uma documentação detalhada de instalação e configuração de dependências. A Plataforma desenvolvida permitiu uma experiência imersiva e interativa, proporcionando uma base sólida para futuras aplicações em VR. Durante este processo, enfrentei desafios ao tentar alocar máquinas virtuais para suportar a plataforma devido a limitações de cota na minha conta do GCP, que foram devidamente documentados.

No decorrer da Semana 10, iniciei o curso "*Start to Finish Unity® Games and Python Coding*" para aprimorar minhas habilidades na integração de Python com Unity. Este curso revelou algumas incompatibilidades com as versões mais recentes do Unity e do Meta Quest SDK, destacando a necessidade de adaptar as técnicas ensinadas para o ambiente atual. Paralelamente, comecei a integração da tecnologia DreamFusion na Plataforma web, enfrentando problemas de compatibilidade e documentação inadequada que resultaram em erros na implementação. Esse processo destacou a importância de manter a documentação atualizada e compatível com as versões mais recentes das tecnologias utilizadas. Os documentos referentes às Semanas 9 e 10 podem ser encontrados no Apêndice 7.

Na Semana 11, concentrei meus esforços na integração completa do Meta Quest 3 com a Unity, focando na criação de um ambiente imersivo com imagens 360. Este processo envolveu a configuração do ambiente de realidade virtual, onde uma imagem 360 foi incorporada como fundo, criando um cenário visualmente envolvente. A integração foi demonstrada através de um vídeo que capturou a experiência completa, mostrando a funcionalidade e a eficácia do sistema desenvolvido.

Além da integração visual, desenvolvi uma interface que permite ao usuário gravar áudio utilizando o microfone do Meta Quest 3. Este áudio é então transcrito em tempo real para um campo de texto, utilizando o serviço *Google Speech-to-Text*. Para implementar essa funcionalidade, criei um código em C# que integra botões ao sistema de áudio, permitindo um controle intuitivo e eficiente pelo usuário. A documentação detalhada do código e o passo a passo da criação da interface foram disponibilizados no GitHub, proporcionando uma base clara e acessível para futuros desenvolvimentos e melhorias. Os documentos referentes à Semana 11 podem ser encontrados no Apêndice 7.

Minha jornada para me tornar um especialista em Visão 3D para Tecnologias Imersivas foi marcada por desafios e conquistas. Cada semana trouxe novas aprendizagens e avanços que fortaleceram minha compreensão e habilidades nesta área complexa e inovadora. A experiência adquirida durante a Residência não apenas aprimorou minhas capacidades técnicas, mas também me proporcionou uma visão científica mais ampla e integrada. O próximo passo é continuar explorando as possibilidades oferecidas pela realidade virtual e aumentar ainda mais a qualidade e a interatividade das aplicações desenvolvidas. A reflexão sobre esse período mostra que a interseção entre aprendizado contínuo e aplicação prática é fundamental para o desenvolvimento de soluções tecnológicas que realmente impactam a sociedade.

# APÊNDICE 1

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("gate") de aprovação: 10 de abr. de 2024

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| Rodrigo Mendes de Carvalho |

## Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Conforme requerido para essa etapa inicial, essa primeira entrega consiste na definição da área de atuação do projeto, além do mapeamento dos objetivos que serão abordados ao decorrer do processo da residência em Inteligência Artificial.

- ☑ Mapeamento e escolha da área de atuação através dos temas da Conference on Computational Science and Computational Intelligence (CSCI):

  Conforme a leitura indicada para direcionamento das áreas, na Conference on Computational Science and Computational Intelligence (CSCI'23), esse projeto tem como tema definido: Neural networks and applications, Emerging technologies; and Applications (including: computer vision, signal processing, Large language models, emerging applications)
- ☑ Leitura e aprofundamento das áreas de Research Track on Signal & Image Processing, Computer Vision & Pattern Recognition (CSCI-RTPC).

Os documentos de revisão das áreas abordadas dentro  Research Track on Signal & Image Processing, Computer Vision & Pattern Recognition (CSCI-RTPC), estão em no seguinte link:
Entrega 10-04 Rodrigo Mendes de Carvalho

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Conforme planejado, para a entrega do próximo GATE, tenho os seguinte objetivos:

- Gerar uma organização disponibilizada na web (gitlab,github), para depositar os documentos e possíveis scripts desse projeto.
- Levantar artigos relevantes para a organização.
- Estudar tópicos iniciais levantados no tema (CSCI-RTPC) dentro da Conference on Computational Science and Computational Intelligence (CSCI'23), inicialmente envolvendo Processamento de imagem e sinal, como (Identificação de Sinais; Reconstrução de Sinal; Análise espectral; Processamento de sinais estatísticos e ópticos; Análise de Sinais Tempo-Frequência; Ferramentas de software para imagens)
- Resumo dos artigos e dos tópicos levantados.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

**ALDO ANDRÉ DÍAZ SALAZAR:** Go!

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("gate") de aprovação: 17 de abr. de 2024

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| Rodrigo Mendes de Carvalho |

## Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

- Gerar uma organização disponibilizada na web (gitlab,github), para depositar os documentos e possíveis scripts desse projeto.
  github Rodrigo Mendes Residência
- Levantar artigos relevantes para a organização.
  Artigos e Livros - Processamento de Sinais
- Estudar tópicos iniciais levantados no tema (CSCI-RTPC) dentro da Conference on Computational Science and Computational Intelligence (CSCI'23), inicialmente envolvendo Processamento de imagem e sinal, como (Identificação de Sinais; Reconstrução de Sinal; Análise espectral; Processamento de sinais estatísticos e ópticos; Análise de Sinais Tempo-Frequência; Ferramentas de software para imagens)
- Resumo dos artigos e dos tópicos levantados.
  Entrega 17-04 Rodrigo Mendes de Carvalho

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Estudar os livros levantados em Processamento de Sinais.
- Explorar as ferramentas de processamento de sinais citadas no GATE do dia 17
- Levantar artigos, livros e estudos do tópico de Visão Computacional: Redes de Câmeras e Visão; Sensores e Visão Precoce; Tecnologias de Aprendizado de Máquina para Visão; Extração de recursos de imagem; Visão Cognitiva e Biologicamente Inspirada; Reconhecimento de objeto;
- Resumo dos artigos e dos tópicos levantados.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

**ALDO ANDRÉ DÍAZ SALAZAR:** Go!

# Revisão das áreas abordadas dentro  Research Track on Signal & Image Processing, Computer Vision & Pattern Recognition (CSCI-RTPC)

## Introdução:

O amplo campo das Ciências de Sinais e Imagens preocupa-se principalmente com a geração, coleta, análise, modificação e visualização de sinais e imagens. Este campo é interdisciplinar, abrangendo tópicos tradicionalmente abordados em ciência da computação, física, matemática, engenharia elétrica, inteligência artificial, psicologia e teoria da informação, com a ciência da computação servindo como ponto de ligação entre todas essas diversas áreas (para uma definição formal de Ciência da Imagem). , consulte as páginas wiki relevantes). Do ponto de vista da ciência da computação, o núcleo da Ciência da Imagem inclui os três campos interconectados da ciência da computação abaixo: Processamento de Imagens, Visão Computacional e Reconhecimento de Padrões.

Processamento de Imagens trata do desenvolvimento de algoritmos e técnicas de manipulação e análise de imagens, com o objetivo de aprimorar, restaurar ou extrair informações delas. Este campo se preocupa com tópicos como filtragem de imagens, segmentação, operações morfológicas e extração de características.

A Visão Computacional, por outro lado, preocupa-se em permitir que os computadores interpretem e compreendam informações visuais do mundo, com o objetivo de permitir que as máquinas executem tarefas que normalmente exigiam habilidades visuais humanas. Este campo inclui tópicos como reconhecimento de objetos, compreensão de cena, estimativa de movimento e reconstrução 3D.

Por fim, Reconhecimento de Padrões é a área que se preocupa com o desenvolvimento de algoritmos e técnicas para identificação e classificação de padrões em dados, com aplicações em áreas como reconhecimento de imagem e fala, processamento de linguagem

natural e bioinformática. Este campo inclui tópicos como reconhecimento estatístico de padrões, aprendizado de máquina e redes neurais.

Juntos, esses três campos formam o núcleo da Ciência da Imagem, com a ciência da computação servindo como ponte entre eles e outras disciplinas relacionadas. Ao combinar os pontos fortes destas áreas, os investigadores e profissionais da Ciência da Imagem são capazes de desenvolver soluções inovadoras para uma vasta gama de problemas do mundo real, desde imagens médicas e robótica até processamento multimédia e segurança.

## Processamento de Imagem e Sinal:

O processamento de imagem e sinal envolve manipulação e análise de sinais e imagens para remoção de informações úteis. Algumas das principais áreas e temas nesta área incluem:

- Identificação de Sinais e Análise Espectral: técnicas usadas para identificar e analisar sinais em diferentes domínios de frequência.
- Processamento de Sinais Estatísticos e Ópticos: métodos usados para o processamento de sinais baseados em estatísticas e óptica.
- Análise de Sinais Tempo-Frequência: técnicas usadas para analisar sinais em ambos os domínios do tempo e da frequência.
- Ferramentas de Software para Imagens: softwares e ferramentas para o processamento de imagens.
- Geração, Aquisição e Processamento de Imagens: técnicas usadas para gerar, capturar e processar imagens.
- Modelagem e Algoritmos Baseados em Imagens: algoritmos e técnicas usadas para modelar e analisar imagens.
- Morfologia Matemática: técnicas usadas para analisar a forma e a estrutura de objetos em imagens.
- Geometria de Imagem e Geometria Multi-Visualização: técnicas usadas para visualizar imagens em diferentes perspectivas.
- Imagem 3D: técnicas usadas para analisar e processar imagens tridimensionais.
- Novos algoritmos de redução de ruído e restauração de imagem: desenvolvimento de novos métodos para reduzir o ruído em imagens.
- Técnicas de aprimoramento e segmentação: técnicas usadas para melhorar a qualidade das imagens e segmentar objetos de interesse.

- Técnicas de movimento e rastreamento: técnicas usadas para rastrear objetos em movimento em sequências de imagens.
- Métodos de marca d'água e wavelet: técnicas usadas para proteger imagens contra cópia e análise.
- Compressão, distribuição e criptografia de imagens: técnicas usadas para comprimir, codificar e criptografar imagens.
- Análise de vídeo: técnicas usadas para analisar e processar sequências de vídeo.
- Técnicas de imagem multi-resolução: técnicas usadas para processar imagens em diferentes resoluções.
- Análise e avaliação de desempenho: técnicas usadas para avaliar o desempenho de algoritmos e sistemas de processamento de imagens.

## Visão Computacional:

Visão computacional é o campo que estuda como os computadores podem ser programados para analisar imagens e vídeos. Algumas das principais áreas e temas nesta área incluem:

- Redes de Câmeras e Visão: técnicas usadas para capturar e processar imagens de múltiplas câmeras.
- Sensores e Visão Precoce: técnicas usadas para capturar e processar imagens de sensores e câmeras.
- Tecnologias de Aprendizado de Máquina para Visão: técnicas usadas para treinar computadores para reconhecer e classificar objetos em imagens.
- Extração de recursos de imagem: técnicas usadas para extrair características úteis de imagens e vídeos.
- Visão Cognitiva e Biologicamente Inspirada: técnicas usadas para desenvolver sistemas de visão computacional inspirados no funcionamento do cérebro humano.
- Reconhecimento de objeto: técnicas usadas para detectar e classificar objetos em imagens e vídeos.
- Visão Estéreo: técnicas usadas para capturar e processar imagens e vídeos em 3D usando múltiplas câmeras.
- Visão Ativa e Robótica: técnicas usadas para desenvolver sistemas de visão computacional para robótica e outras aplicações.
- Reconhecimento facial e de gestos: técnicas usadas para detectar e classificar rostos e gestos em imagens e vídeos.
- Técnicas fuzzy e neurais em visão: técnicas usadas para desenvolver sistemas de visão computacional usando lógica fuzzy e redes neurais.
- Processamento e análise de imagens médicas: técnicas usadas para analisar imagens médicas para diagnóstico e tratamento de doenças.
- Novas técnicas de compreensão de imagens de documentos: técnicas usadas para processar e extrair informações de documentos.

- Arquiteturas de máquinas para fins especiais para visão: técnicas usadas para desenvolver hardware otimizado para aplicações de visão computacional.
- Autenticação biométrica: técnicas usadas para identificar pessoas com base em características biométricas.

## Reconhecimento de Padrões:

Reconhecimento de padrões é o campo que estuda como os computadores podem ser programados para reconhecer e classificar padrões em diferentes tipos de dados. Algumas das principais áreas e temas nesta área incluem:

- Algoritmos de Classificação Supervisionada e Não Supervisionada: técnicas usadas para classificar dados com base em exemplos de treinamento.
- Técnicas de Clusterização: técnicas usadas para agrupar dados em clusters com base em suas características.
- Métodos de Redução de Dimensionalidade em Reconhecimento de Padrões: técnicas usadas para reduzir a dimensionalidade dos dados para facilitar o reconhecimento de padrões.
- Aprendizagem Simbólica: técnicas usadas para desenvolver sistemas de reconhecimento de padrões usando regras simbólicas.
- Algoritmos de Aprendizagem Ensemble: técnicas usadas para combinar múltiplos algoritmos de aprendizado para melhorar o desempenho do reconhecimento de padrões.
- Algoritmos de Análise: técnicas usadas para analisar dados para extrair informações úteis.
- Métodos Bayesianos em Reconhecimento e Correspondência de Padrões: técnicas usadas para desenvolver sistemas de reconhecimento baseados em modelos probabilísticos.
- Reconhecimento de Padrões Estatísticos: técnicas usadas para reconhecer e classificar padrões com base em modelos estatísticos.
- Invariância no Reconhecimento de Padrões: técnicas usadas para desenvolver sistemas de reconhecimento de padrões que são invariantes a certas transformações.
- Reconhecimento Baseado em Conhecimento: técnicas usadas para desenvolver sistemas de reconhecimento de padrões utilizando conhecimento prévio sobre os dados.
- Reconhecimento de Padrões Estruturais e Sintáticos: técnicas usadas para reconhecer e classificar padrões com base em sua estrutura e sintaxe.
- Aplicações em Segurança, Medicina, Robótica, GIS, Sensoriamento Remoto, Inspeção Industrial, Avaliação Não Destrutiva (ou NDE): diferentes aplicações do reconhecimento de padrões em diversas áreas.

- Estudos de caso e tecnologias emergentes: estudos de caso e novas tecnologias no campo do reconhecimento de padrões.

## Referências:


- Research Track on Signal & Image Processing, Computer Vision & Pattern Recognition (CSCI-RTPC)
- Sonka, M., Hlavac, V., & Boyle, R. (2014). Image processing, analysis, and machine vision. Pearson Education. https://www.pearson.com/us/
- Gonzalez, R. C., & Woods, R. E. (2018). Digital image processing. Pearson Education. https://www.pearson.com/us/
- Szeliski, R. (2010). Computer vision: algorithms and applications. Springer Science & Business Media. https://www.springer.com/us/
- Forsyth, D. A., & Ponce, J. (2012). Computer vision: a modern approach. Prentice Hall. https://www.cs.cmu.edu/~16385/
- Bishop, C. M. (2006). Pattern recognition and machine learning. Springer. https://www.springer.com/gp/
- Duda, R. O., Hart, P. E., & Stork, D. G. (2012). Pattern classification. John Wiley & Sons. https://www.wiley.com/en-us/
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press. https://www.deeplearningbook.org/
- Szeliski, R. (2021). Deep learning for computer vision: A tutorial. Foundations and Trends® in Computer Graphics and Vision, 14(3-4), 145-383. https://www.nowpublishers.com/article/Details/CGV-045
- Csurka, G. (2017). Handbook of Image and Video Processing. Academic Press. https://www.sciencedirect.com/book/9780128028074/handbook-of-image-and-video-processing
- Zhang, L., & Zhang, D. (2018). Computer Vision: Algorithms and Applications. Springer. https://link.springer.com/book/10.1007%2F978-3-319-49439-2
- Li, M., & Zhao, Y. (2019). Deep Learning for Image Super-Resolution: A Survey. IEEE Signal Processing Magazine, 36(2), 53-63. https://ieeexplore.ieee.org/document/864634
- Sultana, N., & Chen, Y. (2020). A Survey on Object Detection and Recognition: Recent Advances and Future Directions. IEEE Access, 8, 118006-118024. https://ieeexplore.ieee.org/document/9006262
- Hussain, A., & Muhammad, K. (2021). A Comprehensive Review on Deep Learning-Based Image Segmentation. IEEE Access, 9, 10866-10886. https://ieeexplore.ieee.org/document/9351652

- Zhang, Y., & Patel, V. M. (2021). A Survey on Deep Learning for Image and Video Inpainting. IEEE Transactions on Image Processing, 30, 3436-3452. https://ieeexplore.ieee.org/document/9351652

## Livros:

- "Computer Vision: Algorithms and Applications" por Richard Szeliski (2010): https://szeliski.org/Book/
- "Deep Learning for Computer Vision" por Siddha Ganju, Meher Kasam e Anirudh Koul (2021): https://www.deeplearningforcv.com/
- "Computer Vision: A Modern Approach" por David A. Forsyth e Jean Ponce (2012): https://www.cs.cmu.edu/~efros/courses/LBMV07/ForsythPonce.pdf
- "Deep Learning" de Ian Goodfellow, Yoshua Bengio e Aaron Courville (2016): https://www.deeplearningbook.org/

## Vídeos:

1. "Computer Vision Fundamentals" da Georgia Tech (2021): https://www.udacity.com/course/computer-vision-fundamentals--ud810
2. "Deep Learning for Computer Vision with Python" por Jose Portilla (2021): https://www.udemy.com/course/deep-learning-for-computer-vision-with-python/
3. "Pattern Recognition and Machine Learning" por Christopher Bishop (2006): https://www.microsoft.com/en-us/research/academic-program/pattern-recognition-and-machine-learning/
4. "Deep Learning Specialization" por Andrew Ng (2021): https://www.coursera.org/specializations/deep-learning

# Resumo dos artigos e dos tópicos levantados de Processamento de Sinais

## Advances in Hyperspectral Image and Signal Processing: A Comprehensive Overview of the State of the Art :

Esta revisão de literatura concentra-se no campo da análise de imagens hiperespectrais (HSIA), que envolve a extração de informações úteis de dados hiperespectrais. Os dados hiperespectrais são caracterizados por sua alta resolução espacial e espectral e estão cada vez mais disponíveis em diversas fontes, como satélites de observação da Terra, plataformas aéreas e sensores terrestres. A revisão cobre os seguintes tópicos:

### Desafios na HSIA

A HSIA enfrenta vários desafios, incluindo:

- Alta dimensionalidade de dados hiperespectrais
- Correção atmosférica precisa e confiável
- Desafios computacionais associados ao processamento de grandes volumes de dados hiperespectrais

### Técnicas de redução de dimensionalidade

A revisão apresenta uma visão geral do estado da arte em técnicas de redução de dimensionalidade para HSIA, incluindo:

- Análise de Componentes Principais (PCA)
- Análise de Componentes Independentes (ICA)
- Fatoração de Matriz Não Negativa (NMF)

## Técnicas de Classificação Supervisionada

A revisão cobre o que há de mais moderno em técnicas de classificação supervisionada para dados hiperespectrais, incluindo:

- Máquinas de vetores de suporte (SVM)
- Florestas Aleatórias (RF)
- Redes Neurais Convolucionais (CNN)

## Técnicas de pré-processamento

A revisão destaca a importância das técnicas de pré-processamento, como correção atmosférica, correção radiométrica e co-registro, na melhoria do desempenho dos algoritmos de classificação.

## Infraestruturas em nuvem e computação rápida

A revisão discute o papel das infraestruturas em nuvem na HSIA, particularmente no contexto de plataformas em tempo real. Ele destaca os desafios associados à computação rápida, incluindo problemas de alto consumo de energia e tolerância à radiação.

## Conclusão

A revisão conclui enfatizando a necessidade de mais pesquisas em HSIA, particularmente nas áreas de técnicas de pré-processamento, desenvolvimento de algoritmos e integração de plataformas. Os autores gostariam de agradecer o apoio de diversas instituições e indivíduos que forneceram conjuntos de dados e recursos para a revisão.

Este resumo fornece uma visão geral concisa e organizada da revisão da literatura, que pode ser facilmente integrada em um documento. A revisão abrange vários aspectos da HSIA, incluindo desafios, técnicas de redução de dimensionalidade, técnicas de classificação supervisionada, técnicas de pré-processamento e infraestruturas em nuvem. A revisão destaca a importância destes tópicos e a necessidade de mais pesquisas em HSIA.

## Capítulo 1 - Digital Image Processing Third Edition Rafael C. Gonzalez University of Tennessee Richard E. Woods MedData Interactive:

Esta seção discute as origens, a história e os componentes do processamento digital de imagens. O processamento digital de imagens envolve o uso de computadores digitais para processar imagens digitais, que são compostas por um número finito de elementos chamados pixels. O campo do processamento digital de imagens tem origem na indústria jornalística, com as primeiras imagens digitais sendo transmitidas entre Londres e Nova York no início da década de 1920. O desenvolvimento dos computadores digitais em meados do século 20 foi um fator importante no crescimento do processamento digital de imagens, pois forneceu o poder computacional e as capacidades de armazenamento necessários.

Os componentes de um sistema típico de processamento de imagens de uso geral incluem sensores de imagem, um digitalizador, hardware especializado de processamento de imagens, um computador de uso geral, software de processamento de imagens, armazenamento em massa, exibições de imagens e recursos de rede. Os sensores de imagem convertem a energia irradiada pelo objeto sendo fotografado em um sinal elétrico, que é então convertido em formato digital pelo digitalizador. Hardware especializado de processamento de imagens executa operações primitivas, como operações aritméticas e lógicas, em imagens inteiras em paralelo. O computador de uso geral executa software de processamento de imagem e controla a operação geral do sistema. O armazenamento em massa é usado para armazenar grandes quantidades de dados de imagem e exibições de imagens são usadas para visualizar as imagens processadas.

A história do processamento digital de imagens remonta ao desenvolvimento dos primeiros computadores em meados do século XX. Os primeiros computadores poderosos o suficiente para realizar tarefas significativas de processamento de imagens surgiram no início da década de 1960, e o campo cresceu rapidamente desde então. O processamento digital de imagens é agora usado em uma ampla variedade de aplicações, incluindo imagens médicas, sensoriamento remoto, inspeção industrial e aplicação da lei.

Portanto , o processamento digital de imagens envolve o uso de computadores digitais para processar imagens digitais. O campo tem suas origens na indústria jornalística e tem crescido rapidamente desde o desenvolvimento dos computadores digitais em meados do século XX. Um sistema típico de processamento de imagens de uso geral inclui sensores de imagem, um digitalizador, hardware especializado de processamento de imagens, um computador de uso geral, software de processamento de imagens, armazenamento em massa, exibições de imagens e recursos de rede. O processamento digital de imagens é agora utilizado numa ampla variedade de aplicações e é provável que a sua importância continue a crescer no futuro.

## Graph Signal Processing: Overview, Challenges and Applications:

Este artigo fornece uma visão geral do Processamento de Sinais em Grafos (GSP), um campo que busca desenvolver ferramentas para processar dados definidos em domínios de grafos irregulares. Ele discute a conexão entre GSP e o processamento de sinais digitais convencionais e dá uma breve perspectiva histórica sobre o desenvolvimento do GSP. O artigo então resume os avanços recentes em GSP, incluindo métodos para amostragem, filtragem e aprendizagem de grafos. Ele também revisa o progresso em vários campos de aplicação, como o processamento e análise de dados de redes de sensores, dados biológicos e aplicativos para processamento de imagens e aprendizagem de máquina.

## Introdução e Motivação:

Dados estão sendo cada vez mais gravados em estruturas irregulares e complexas, como redes sociais e redes biológicas. Os grafos oferecem a capacidade de modelar tais dados e

interações complexas entre eles. O GSP estende conceitos e ferramentas de processamento de sinais clássicos para dados residentes em grafos. Ele define um sinal de grafos como um conjunto de valores que residem em um conjunto de nós, que estão conectados via (possivelmente ponderados) arestas. Diferentes tipos de grafos modelam diferentes tipos de redes que esses nós representam.

## Processamento de Sinais em Grafos: Conceitos Básicos:

O GSP define um sinal de grafos como um conjunto de valores que residem em um conjunto de nós, que estão conectados via (possivelmente ponderados) arestas. Esses sinais podem ter propriedades, como suavidade, que precisam ser devidamente definidas. Eles também podem ser representados por átomos básicos e podem ter uma representação espectral. A transformada de Fourier em grafos permite o desenvolvimento da intuição adquirida no cenário clássico e a extensão para grafos, permitindo que as noções de frequência e limitação de banda sejam discutidas.

## Processamento de Sinais em Grafos: Aplicações

O GSP tem sido aplicado a vários campos, incluindo processamento e análise de dados de redes de sensores, dados biológicos e aplicativos para processamento de imagens e aprendizagem de máquina. Ele tem sido usado para abordar tarefas complexas, como amostragem de maneira fundamentada, e para desenvolver ferramentas para processamento de baixo nível, como denoising, inpainting e compressão.

## Conclusão

O GSP é um campo promissor que tem o potencial de desenvolver novas ferramentas e abordar problemas existentes de diferentes perspectivas. Ele tem fortes conexões com vários domínios de pesquisa teóricos e práticos e tem sido aplicado a vários campos,

incluindo processamento e análise de dados de redes de sensores, dados biológicos e aplicativos para processamento de imagens e aprendizagem de máquina.

## Understanding the Basis of Graph Signal Processing via an Intuitive Example-Driven Approach:

Os autores apresentam um modelo de sinal de grafo que pode ser usado para analisar e processar sinais em sistemas de comunicação e processamento de sinais em sistemas físicos e artificiais. A técnica de filtragem de sinais de grafo ponderados com normalização matricial D fornece resultados significativamente melhores do que os métodos de filtragem tradicionais. O modelo de sinal de grafo oferece uma base matemática e computacional sólida para a análise de sistemas de sinais e campos, e pode ser aplicado a uma ampla variedade de cenários e domínios de sinal.

A técnica de filtragem de sinais de grafo ponderados com normalização matricial D é usada para estimar o sinal de campo elétrico subjacente a um ruído de tensão altamente ruidoso. Ao aplicar essa técnica, os autores obtiveram um SNR (Relação Sinal/Ruído) bem superior ao do sinal original ruidoso, indicando a capacidade da técnica de filtragem de sinais de grafo ponderados para extrair informações relevantes do ruído em sistemas de comunicação e processamento de sinais.

O modelo de sinal de grafo pode ser aplicado a uma ampla variedade de cenários e domínios de sinal, além de fornecer uma base matemática e computacional sólida para a análise de sistemas de sinais e campos.

O artigo destaca que os sistemas de sinais de grafo oferecem uma nova perspectiva para o processamento de sinais em sistemas de comunicação e processamento de sinais em sistemas físicos e artificiais. Esses sistemas podem ser aplicados a uma ampla variedade de cenários e domínios de sinal, efetuando estimativas sem viés do sinal subjacente e melhorando a relação sinal/ruído em comparação com os métodos de filtragem tradicionais.

# Reconstruction of Time-varying Graph Signals via Sobolev Smoothness :

## Introdução:

Este artigo apresenta um novo algoritmo para a reconstrução de sinais de grafo temporalmente variáveis com base na extensão de uma função de smoothness de Sobolev. Os autores assumem que as diferenças temporais dos sinais de grafo são suaves e introduzem o algoritmo GraphTRSS para reconstruir sinais de grafo temporalmente variáveis a partir de amostras discretas. O algoritmo é avaliado em várias bases de dados, incluindo duas bases de dados de COVID-19 e duas bases de dados ambientais, e tem superado muitos métodos de estado da arte existentes para a reconstrução de sinais de grafo temporalmente variáveis.

## Contexto:

O Processamento de Sinais de Grafo (GSP) é uma área de pesquisa emergente que estende conceitos clássicos de processamento de sinais para grafos. O GSP tem diversas aplicações em áreas como redes de sensores, aprendizado de máquina e processamento de imagens. A amostragem e a reconstrução de sinais de grafo estáticos desempenham um papel central no GSP. No entanto, muitos sinais de grafo reais são intrinsecamente temporalmente variáveis, e a suavidade das diferenças temporais dos sinais de grafo pode ser usada como uma suposição prévia.

## Método:

Os autores assumem que as diferenças temporais dos sinais de grafo são suaves e introduzem um novo algoritmo com base na extensão de uma função de smoothness de Sobolev para a reconstrução de sinais de grafo temporalmente variáveis a partir de amostras discretas. O algoritmo é chamado de Reconstrução de Sinais de Grafo

Temporalmente Variáveis via Smoothness de Sobolev (GraphTRSS). Os autores exploram alguns aspectos teóricos da taxa de convergência do algoritmo GraphTRSS ao estudar o número de condições do Hessiano associado ao seu problema de otimização.

## Resultados:

O algoritmo GraphTRSS tem a vantagem de convergir mais rapidamente do que outros métodos baseados em operadores de Laplace sem exigir a decomposição de autovalores ou inversões de matrizes caras. O algoritmo tem mostrado excelente desempenho em duas bases de dados ambientais para a recuperação de sinais de partículas em suspensão e sinais de temperatura da superfície do mar.

## Conclusão:

O algoritmo GraphTRSS é um método promissor para a reconstrução de sinais de grafo temporalmente variáveis. O algoritmo tem superado muitos métodos de estado da arte existentes para a reconstrução de sinais de grafo temporalmente variáveis e tem mostrado excelente desempenho em duas bases de dados ambientais. Os autores sugerem que o algoritmo GraphTRSS poderia ser usado em várias aplicações, como a estimativa de novos casos de doenças infecciosas ou a recuperação de sinais de temperatura da superfície do mar para o estudo dos movimentos da Terra.

# Relay Vibration Protection Simulation Experimental Platform Based on Signal Reconstruction of MATLAB Software:

## Introdução:

A proteção de vibração de relés é uma medida técnica importante para garantir a operação segura e confiável do sistema de potência. Com o crescimento da escala do sistema de potência e o aumento do grau de automação, os problemas de controle de sistemas de potência tornam-se cada vez mais complexos, exigindo que os dispositivos de proteção de vibração de relés tenham desempenho superior. Antes de serem colocados em uso, é necessário realizar testes em diferentes ambientes para garantir a confiabilidade dos dispositivos de proteção. No entanto, os métodos tradicionais de análise teórica e experimentos físicos têm certas limitações. Portanto, a tecnologia de simulação digital de proteção de vibração de relés pode ser usada para resolver esses problemas. Este artigo propõe uma plataforma experimental de simulação de proteção de vibração de relés com base no software de reconstrução de sinal do MATLAB, que é usada para estudar o princípio de proteção de vibração de relés.

## Desenvolvimento do Trabalho:

### 1. Introdução

### 1.1. Importância da proteção de vibração de relés no sistema de potência

A proteção de vibração de relés desempenha um papel crucial no sistema de potência, pois garante a operação segura e estável do sistema de potência. Com o rápido desenvolvimento do sistema de potência, a proteção de vibração de reles baseada em computador tem sido amplamente utilizada em gerenciamento de linhas e equipamentos de subestações e usinas de potência em diferentes níveis de tensão. O desempenho e o nível de habilidade dos operadores de proteção de vibração de relés estão relacionados à operação segura da rede de potência.

### 1.2. Limitações dos métodos tradicionais de análise teórica e experimentos físicos

Os métodos tradicionais de análise teórica e experimentos físicos têm certas limitações. A análise teórica geralmente não pode refletir plenamente as características complexas do sistema de potência, e os experimentos físicos podem afetar a estabilidade do suprimento de energia e a segurança do equipamento, além de serem caros e demorados.

### 1.3. Significado da tecnologia de simulação digital de proteção de vibração de relés

A tecnologia de simulação digital de proteção de vibração de relés pode ser usada para resolver esses problemas. Ela pode modelar e analisar o sistema de potência, analisar as características de operação dos dispositivos de proteção de vibração de relés em diferentes condições de falha e anormal do sistema de potência, e testar acuradamente os dispositivos de proteção de vibração de relés. Além disso, a simulação de proteção de vibração de relés pode ser usada para estudar novos algoritmos e princípios de proteção de vibração de relés e avaliar a adaptabilidade de redes de potência específicas.

## 2. Revisão da Literatura

### 2.1. Simulação de sistemas de potência

A simulação de sistemas de potência é usada para modelar e analisar sistemas de potência e analisar as características de operação dos dispositivos de proteção de vibração de relés em diferentes condições de falha e anormal do sistema de potência. Ela é a base da pesquisa de simulação de proteção de vibração de relés.

## 2.2. Plataformas de simulação de proteção de vibração de relés

As plataformas de simulação de proteção de vibração de relés podem ser categorizadas em três tipos principais:

1. Plataformas baseadas em hardware: Estas plataformas utilizam circuitos analógicos integrados (ASICs) ou dispositivos lógicos programáveis (PLDs) para modelar e simular os sistemas de potência e dispositivos de proteção de vibração de relés. Exemplos de plataformas baseadas em hardware incluem o MILSIM de Dartmouth e o IPLIS da Powertek.
2. Plataformas baseadas em software: Estas plataformas utilizam softwares de reconstrução de sinal e processamento de vibração para simular os sistemas de potência e dispositivos de proteção de vibração de relés. Exemplos de plataformas baseadas em software incluem o PSIM do LMS-POL e o software de simulação de proteção de vibração de relés da Schneider Electric.
3. Plataformas baseadas em técnicas híbridas: Estas plataformas combinam hardware e software para modelar e simular os sistemas de potência e dispositivos de proteção de vibração de relés. Exemplos de plataformas baseadas em técnicas híbridas incluem o RPSIS do University of York e o ENSIS de Schlumberger.

## 2.3. Desafios na implementação da plataforma de simulação de proteção de vibração de relés

A implementação da plataforma de simulação de proteção de vibração de relés enfrenta vários desafios, como a escolha de modelos e técnicas de simulação apropriados, a validação de modelos e resultados de simulação, e a análise de sensibilidade de sistemas de potência e dispositivos de proteção de vibração de relés.

## 3. Plataforma Experimental de Simulação de Proteção de Vibração de Relé com Base no MATLAB

Para resolver os problemas mencionados no artigo, uma plataforma experimental de simulação de proteção de vibração de relés com base no software de reconstrução de sinal do MATLAB foi desenvolvida. Esta plataforma pode ser usada para estudar o princípio de proteção de vibração de relés, analisar as características de operação dos dispositivos de proteção de vibração de relés em diferentes condições de falha e anormal do sistema de potência, e testar acuradamente os dispositivos de proteção de vibração de relés.

O fluxo de trabalho da plataforma experimental de simulação de proteção de vibração de relés com base no MATLAB pode ser descrito como segue:

1. Selecionar o modelo de relé apropriado: A plataforma de simulação pode ser usada para simular diferentes modelos de relés, como modelos de linha única, triplas, retangulares, de transformador, e assim por diante.
2. Analisar a operação do relé: A plataforma de simulação pode ser usada para analisar a operação do relé em diferentes condições de operação e de falha. Isso inclui a análise da capacidade de carga, a resposta de frequência e a proteção de vibração.
3. Analisar a proteção de vibração de relés: A plataforma de simulação pode ser usada para analisar a proteção de vibração de relés em diferentes condições de operação e de falha. Isso inclui a análise das características de operação do protector de vibração de relés e a análise de sensibilidade do sistema de potência e dispositivos de proteção de vibração de relés.
4. Testar a proteção de vibração de relés: A plataforma de simulação pode ser usada para testar a proteção de vibração de relés em diferentes ambientes e condições. Isso inclui o teste da resposta ao tempo e a frequência do protector de vibração de relés e o teste do para-chama de proteção de vibração de relés.

5. Conclusão: A plataforma experimental de simulação de proteção de vibração de relés com base no MATLAB representa uma abordagem promissora para o estudo de proteção de vibração de relés. A plataforma pode ser usada para analisar as características de operação dos dispositivos de proteção de vibração de relés e testar acuradamente os dispositivos de proteção de vibração de relés em diferentes condições de falha e anormal do sistema de potência. Futuros trabalhos podem enfocar na otimização dos algoritmos e métodos de simulação e na validação dos modelos e resultados de simulação.

## Modern Trends in Hyperspectral Image Analysis: A Review:

Este artigo de revisão enfoca os fundamentos da análise de imagens hiperespectrais e suas aplicações modernas. Os autores discutem os princípios da imagem hiperespectral, terminologias comuns de sensoriamento remoto hiperespectral e destacam as aplicações da imagem hiperespectral em vários campos.

## Imagem hiperespectral:

As imagens hiperespectrais são caracterizadas pela sua resolução espacial e espectral. A resolução espacial mede a relação geométrica dos pixels da imagem entre si, enquanto a resolução espectral determina as variações nos pixels da imagem em função do comprimento de onda. Uma imagem hiperespectral possui duas dimensões espaciais (Sx e Sy) e uma dimensão espectral (Sλ). Os dados hiperespectrais são representados na forma de um cubo de dados hiperespectrais 3D.

## Resolução espacial:

A resolução espacial pode ser definida como o menor detalhe discernível em uma imagem. É inversamente proporcional ao tamanho do patch, o que significa que um tamanho menor

do patch resulta em detalhes mais altos que podem ser interpretados a partir da cena observada.

## Resolução Espectral:

A resolução espectral pode ser definida como o número de bandas espectrais e a faixa do espectro eletromagnético medido pelo sensor. Um sensor de imagem pode responder a uma grande faixa de frequência, mas ainda assim ter uma baixa resolução espectral se adquirir um pequeno número de bandas espectrais. As imagens hiperespectrais adquirem imagens em numerosas bandas espectrais contíguas e extremamente estreitas nos segmentos do infravermelho médio, infravermelho próximo e visível do espectro eletromagnético.

## Resolução temporária:

A resolução temporal depende das características orbitais do sensor de imagem. Geralmente é definido como o tempo necessário para a plataforma do sensor revisitar e obter dados exatamente do mesmo local.

## Compreendendo as assinaturas espectrais:

Os materiais presentes na superfície da Terra absorvem, transmitem e refletem as radiações eletromagnéticas do sol de uma forma única. Os sensores hiperespectrais permitem medir todos os tipos de energia eletromagnética dentro de uma faixa especificada à medida que interagem com os materiais, permitindo assim observar as características e mudanças distintas na superfície da Terra. A refletância é a medida da energia eletromagnética refletida na superfície de um material. Assinaturas espectrais ou curvas de resposta espectral podem ser plotadas para diferentes materiais presentes na superfície da Terra, que podem ser usadas para fins de classificação.

## Aplicações modernas de imagens hiperespectrais:

A imagem hiperespectral tem sido cada vez mais usada para uma ampla variedade de aplicações comerciais, industriais e militares. Os autores se concentram nas aplicações de HSI à qualidade e segurança alimentar, diagnóstico médico e cirurgia guiada por imagem, sensoriamento remoto, como agricultura de precisão e gestão de recursos hídricos, exames forenses, como detecção de falsificação de documentos e autenticação de obras de arte, e defesa e segurança interna.

## Avaliação da Qualidade e Segurança Alimentar:

Imagens hiperespectrais têm sido utilizadas para identificação de defeitos e detecção de contaminações em alimentos. Ele foi aplicado com sucesso para examinar a firmeza e o conteúdo de sólidos de mirtilos, o sabor farinhento da maçã, o teor de gordura entre os músculos da carne de porco e a distribuição de cores no filé de salmão.

## Diagnóstico médico:

A análise de imagens hiperespectrais está sendo amplamente utilizada para diagnóstico médico devido à sua capacidade de fornecer imagens em tempo real de informações de biomarcadores e informações espectrais de tecidos. Além do diagnóstico, os sistemas HSI também são utilizados em cirurgias guiadas por imagem.

## Sensoriamento remoto:

O sensoriamento remoto hiperespectral tem sido usado para agricultura de precisão, gerenciamento de recursos hídricos e identificação de materiais na superfície terrestre.

## Exame Forense:

A imagem hiperespectral tem sido usada para exame forense de documentos, como determinação da autenticidade de documentos legais, detecção de falsificação, datação retroativa e fraude.

## Autenticação de arte:

A imagem hiperespectral foi proposta como um método novo e não destrutivo de exame de obras de arte na literatura recente. Tem sido utilizado para conservação e restauro de obras de arte como pinturas, permitindo a identificação de regiões restauradas em pinturas e distinguindo-as das regiões importantes da pintura original.

## Defesa e Segurança Interna:

A imagem hiperespectral tem sido usada para aplicações de defesa e segurança interna, como contra-contramedidas para detecção e reconhecimento de alvos camuflados em aplicações militares.

## Time-Frequency Signal Analysis Boualem Boashash:

O livro apresenta os fundamentos da análise de sinais no domínio do tempo-frequência e suas aplicações em aplicações de imagem hiperespectral e processamento de sinais em outras modalidades de sensores. Ele também aborda algoritmos e métodos de aprendizado de máquina e aplicações de campo. A análise de sinais no domínio do tempo-frequência é uma técnica avançada que permite extrair informações sobre o conteúdo de sinais, como amplitudes, frequências e fases. As aplicações de análise de sinais no domínio do tempo-frequência em imagem hiperespectral incluem análise de imagens, detecção de anomalias e monitoramento de recursos. A aprendizagem de máquina pode ser usada para melhorar a precisão e a eficiência da análise de sinais no domínio do tempo-frequência.

## Fundamentos da análise de sinais no domínio do tempo-frequência:

A análise de sinais no domínio do tempo-frequência é uma técnica avançada que utiliza a teoria de espectros de sinais no domínio do tempo-frequência. A teoria de espectros de sinais permite extrair informações sobre o conteúdo de sinais, como amplitudes, frequências e fases. Existem duas representações espectralmente contínuas e discretas de sinais de tempo discreto e contínuo: a análise de espectrogramas e a análise de sinais temporais e de frequência.

## Aplicações de análise de sinais no domínio do tempo-frequência em imagem hiperespectral:

A análise de sinais no domínio do tempo-frequência é aplicada em imagem hiperespectral, uma técnica de análise de imagens que utiliza várias bandas espectrais para representar diferentes propriedades dos objetos. As aplicações de análise de sinais no domínio do tempo-frequência em imagem hiperspectral incluem: a análise de imagens de alta resolução espectral (HSI) em aplicações de agricultura, medicina, segurança e meio ambiente; a análise de sinais temporais e frequenciais em aplicações de telecomunicações, sistemas de radar e transmissão de sinais de controle; e o processamento de sinais de radiofrequência em aplicações de detecção de anomalias, monitoramento de recursos e detecção de eventos em sistemas de rádio de comunicação.

## Algoritmos e métodos de aprendizado de máquina para análise de sinais no domínio do tempo-frequência:

A análise de sinais no domínio do tempo-frequência pode ser aplicada em aprendizado de máquina, como regressão linear, análise de componentes principais (PCA) e máquina de

vetores de suporte (SVM). Além disso, técnicas de aprendizado profundo, como redes neurais artificiais (ANN) e redes neurais convolucionais (CNN), também podem ser aplicadas em análise de sinais no domínio do tempo-frequência.

## Conclusão:

A análise de sinais no domínio do tempo-frequência é uma técnica avançada que pode extrair informações úteis de imagens hiperespectrais e outros tipos de dados de sensores. Essas técnicas são aplicáveis em várias aplicações práticas, como agricultura, medicina, segurança e meio ambiente. A aprendizagem de máquina pode ser usada para melhorar a precisão e a eficiência da análise de sinais no domínio do tempo-frequência.

## The List of the Top Ten Image Processing Tools for 2023:

## OpenCV (Biblioteca de Visão Computacional de Código Aberto):

OpenCV é uma biblioteca de software de visão computacional e aprendizado de máquina de código aberto amplamente utilizada. Possui interfaces C++, Python e Java e oferece suporte a Windows, Linux, Mac, iOS e Android. OpenCV fornece uma variedade de técnicas de processamento de imagens, como filtragem de imagens, detecção de objetos e junção de imagens.

## Scikit-image (SciPy Imaging):

Scikit-image é um pacote de processamento de imagem específico do Python que se baseia na biblioteca SciPy. Possui uma variedade de algoritmos de processamento de imagem, incluindo filtragem, restauração, segmentação e detecção de recursos.

## Pillow (Python Imaging Library):

Pillow é uma poderosa biblioteca de imagens Python que oferece suporte à abertura, manipulação e salvamento de muitos formatos de arquivo de imagem diferentes. Ele fornece ampla funcionalidade para cortar, redimensionar, girar e aplicar vários efeitos às imagens.

## matplotlib (Matplotlib):

matplotlib é uma biblioteca de plotagem para a linguagem de programação Python. Ele fornece uma API orientada a objetos para incorporar gráficos em aplicativos usando kits de ferramentas GUI de uso geral, como Tkinter, wxPython, Qt ou GTK.

## ImageMagick:

ImageMagick é um conjunto de software para criar, editar e compor imagens bitmap. Inclui ferramentas e bibliotecas para manipular, redimensionar, girar, inverter ou converter imagens bitmap.

## scikit-learn (SciPy Machine Learning):

scikit-learn é uma biblioteca de aprendizado de máquina para Python que fornece ferramentas para a criação de modelos de aprendizado de máquina. Também possui ferramentas para pré-processamento e visualização de dados.

## PyTorch (Tensores e Computação Dinâmica):

PyTorch é uma biblioteca de aprendizado de máquina de código aberto para Python. É desenvolvido pelo laboratório de pesquisa de IA do Facebook (FAIR). PyTorch usa um gráfico computacional dinâmico para computação rápida e flexível.

## Tensor Flow (biblioteca de aprendizado de máquina de código aberto do Google):

Tensor Flow é uma biblioteca de aprendizado de máquina de código aberto desenvolvida pelo Google. Ele é usado para construir, treinar e implantar modelos de aprendizado de máquina.

## scikit-learn (SciPy Machine Learning):

scikit-learn é uma biblioteca de aprendizado de máquina para Python. Ele fornece ferramentas para criar modelos de aprendizado de máquina. Também possui ferramentas para pré-processamento e visualização de dados.

## OpenMVG (Open Multi-View Geometry):

OpenMVG é um software de código aberto projetado para resolver problemas nas áreas de Estrutura de Movimento, Multi-View Stereo e Multi-View Geometry. OpenMVG inclui vários métodos para lidar com múltiplas imagens, como estimar a matriz fundamental, a matriz essencial ou o movimento da câmera.

## dlib (Dlibrary for C++):

dlib é um kit de ferramentas C++ moderno que contém algoritmos de aprendizado de máquina e ferramentas para a criação de software complexo. dlib inclui um módulo de processamento de imagem com suporte para leitura e gravação de imagens em vários formatos, bem como uma variedade de funções de processamento de imagem, como filtragem, redimensionamento e detecção de objetos.

# APÊNDICE 2

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("gate") de aprovação: 24 de abr. de 2024

## Participantes da Entrega [matriculados em Residência em IA]:

RODRIGO MENDES DE CARVALHO

## Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

- Explorado as ferramentas de processamento de sinais citadas no GATE do dia 17

  https://github.com/Rod15greo/Residencia_IA_Rodrigo_Mendes/blob/main/Notebooks_colab/Processamento_de_Imagens/Notebook_Rodrigo_Processamento_de_imagens.ipynb

- Levantado artigos, livros e estudos do tópico de Visão Computacional: Redes de Câmeras e Visão; Sensores e Visão Precoce; Tecnologias de Aprendizado de Máquina para Visão; Extração de recursos de imagem; Visão Cognitiva e Biologicamente Inspirada; Reconhecimento de objeto;

  https://github.com/Rod15greo/Residencia_IA_Rodrigo_Mendes/tree/main/Artigos_e_Livros/Visao_Computacional

- Resumo dos artigos e dos tópicos levantados.

  Entrega 24-04 Rodrigo Mendes de Carvalho

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Explorar as ferramentas de processamento de sinais citadas no GATE do dia 24
- Levantar artigos, livros e estudos do tópico de Reconhecimento de padrões:Algoritmos de Classificação Supervisionada e Não Supervisionada; Técnicas de Clusterização; Métodos de Redução de Dimensionalidade em Reconhecimento de Padrões; Aprendizagem Simbólica; Algoritmos de Aprendizagem Ensemble;
- Resumo dos artigos e dos tópicos levantados.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

**ALDO ANDRÉ DÍAZ SALAZAR:** Go!

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("gate") de aprovação: 8 de mai. de 2024

## Participantes da Entrega [matriculados em Residência em IA]:

RODRIGO MENDES DE CARVALHO

## Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

- Explorado as ferramentas de visão computacional citadas no GATE do dia 24
  Rodrigo Mendes de Carvalho_Visão Computacional
- Levantado artigos, livros e estudos do tópico de Reconhecimento de padrões:Algoritmos de Classificação Supervisionada e Não Supervisionada; Técnicas de Clusterização; Métodos de Redução de Dimensionalidade em Reconhecimento de Padrões; Aprendizagem Simbólica; Algoritmos de Aprendizagem Ensemble;

  https://github.com/Rod15greo/Residencia_IA_Rodrigo_Mendes/tree/main/Artigos_e_Livros/Reconhecimento%20de%20Padr%C3%B5es

- Resumo dos artigos e dos tópicos levantados.
  Entrega 01-05 Rodrigo Mendes de Carvalho
- Explorado as ferramentas de reconhecimento de padrões
  Rodrigo Mendes de Carvalho_Reconhecimento de Padrões
- Feito o mapeamento das áreas, ramos, ferramentas e frameworks de visão 3D
- Documentado o mapeamento da área de visão 3D
  Entrega 08-05 Rodrigo Mendes de Carvalho

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Levantar artigos, livros e estudos do tópico de Visão 3D
- Resumo dos artigos e dos tópicos levantados.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

**ALDO ANDRÉ DÍAZ SALAZAR:** Go!

# Resumo dos artigos e dos tópicos levantados Computer Vision

## 1 - Camera Networks and Vision:

### 1.1 - A vision monitoring system for multipoint deflection of large-span bridge based on camera networking:

O artigo apresenta um sistema avançado de monitoramento por visão para a deflexão multiponto de pontes de grande extensão, denominado Camera Network System (CNS). Este sistema inovador foi desenvolvido para compensar os erros induzidos pelo movimento da câmera e permitir um monitoramento preciso e contínuo da deflexão de pontes em condições desafiadoras.

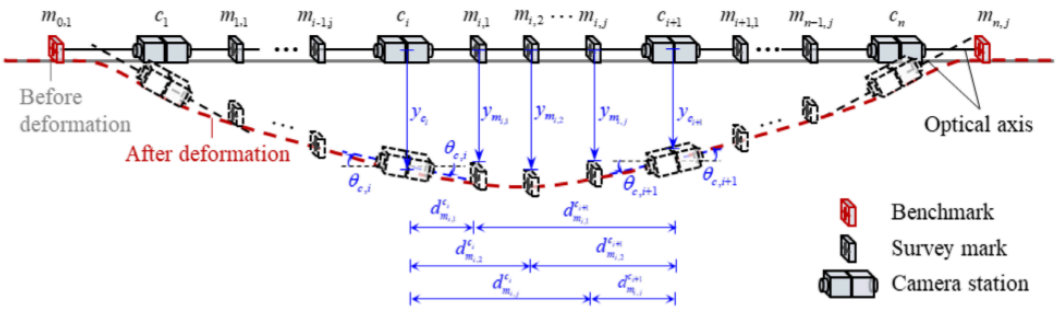



FIGURE 1 Schematic diagram of the proposed camera network system (CNS).

## Introdução:

A necessidade de monitorar a deflexão de pontes de grande extensão de forma precisa e contínua é crucial para garantir a segurança e integridade dessas estruturas. O CNS proposto neste estudo visa superar as limitações dos métodos tradicionais de monitoramento e oferecer uma solução eficaz e inovadora para a engenharia civil.

## Princípio de Medição e Componentes do CNS:

O CNS é baseado no princípio de videometria de deslocamento-relé, onde estações de câmeras duplas são posicionadas ao longo da ponte para capturar imagens de marcadores cooperativos. Cada estação de câmera é composta por dois módulos de câmera que atuam em conjunto, garantindo a precisão e consistência das medições. Além disso, um dispositivo

de suplemento de iluminação infravermelha é integrado para permitir o monitoramento contínuo, mesmo em condições de baixa luminosidade.

## Teste de Campo e Arranjo do CNS:

Um teste de campo foi realizado em uma ponte suspensa com 1038 metros de comprimento para validar a eficácia do CNS. As estações de câmeras foram instaladas dentro das vigas da ponte, permitindo a monitorização contínua da deflexão. Os resultados do teste demonstraram a capacidade do CNS em compensar os erros de movimento da câmera e fornecer medições precisas da deflexão da ponte em tempo real.

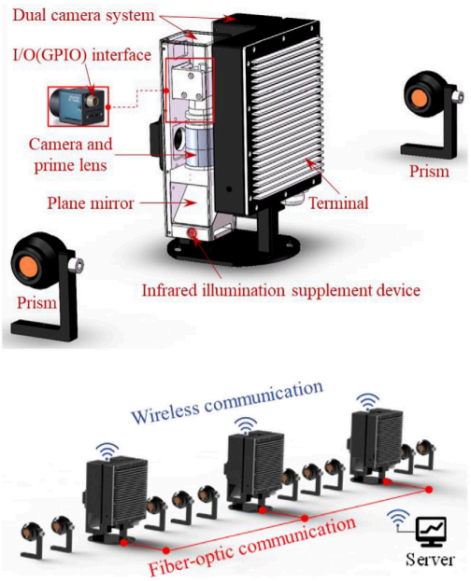



FIGURE 2 Camera network system (CNS) of bridge deflection monitoring.

## Desafios e Discussões:

O estudo destaca desafios futuros, como o design otimizado do CNS para melhorar a precisão e robustez do sistema, a implementação em ambientes externos sujeitos a condições climáticas adversas e a análise pós-processamento dos dados coletados. Além disso, a integração com técnicas avançadas de aprendizado de máquina é apontada como uma área de pesquisa em desenvolvimento para aprimorar a análise dos dados de deflexão da ponte.

## Conclusões e Resultados:

O CNS foi validado como uma solução viável e eficaz para o monitoramento da deflexão de pontes de grande extensão. O sistema demonstrou uma precisão submilimétrica nas medições e foi capaz de fornecer dados de deflexão comparáveis aos obtidos por sistemas tradicionais de monitoramento de longo prazo. Os resultados do teste de campo confirmaram a capacidade do CNS de compensar os erros de movimento da câmera e fornecer medições precisas da deflexão da ponte.

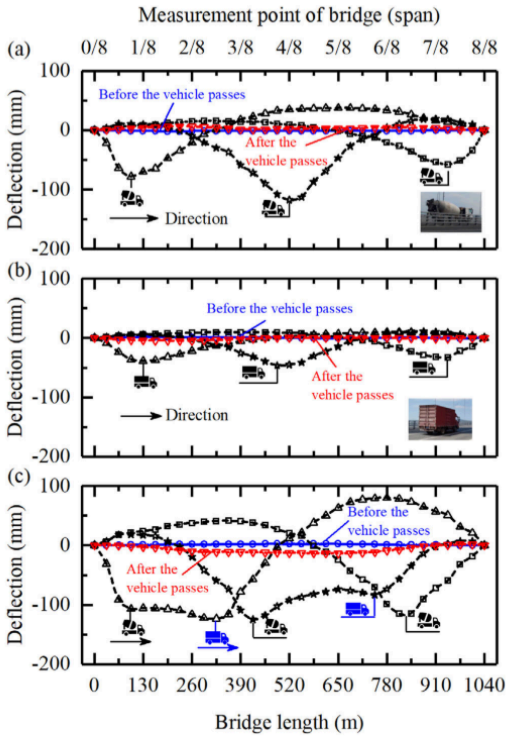



FIGURE 12 Effect of vehicle load on the full-span deflection evolution of bridge: (a) heavy truck, (b) medium truck, and (c) heavy and medium truck.

## Considerações Finais e Agradecimentos:

O estudo foi apoiado por programas de pesquisa da China e destaca a importância do CNS para o monitoramento estrutural de pontes de grande extensão. O artigo conclui ressaltando a eficácia do CNS em compensar os erros de movimento da câmera e fornecer dados precisos e contínuos de deflexão, destacando seu potencial para aplicações futuras e desenvolvimentos na área de monitoramento estrutural.

Este resumo detalhado destaca a inovação, eficácia e potencial do Camera Network System proposto para o monitoramento de deflexão em pontes de grande extensão, evidenciando sua importância e contribuição para a engenharia civil e estrutural.

## 1.2 - Automated monitoring for security camera networks: promise from computer vision labs

### Introdução:

A introdução destaca a importância crescente da monitorização automatizada para redes de câmeras de segurança, que visa melhorar a eficiência e a eficácia da vigilância por vídeo. Com o avanço da tecnologia de visão computacional, tornou-se possível desenvolver algoritmos sofisticados capazes de detectar eventos de interesse, identificar anomalias e resumir vídeos extensos de forma automatizada. Essa abordagem não apenas agiliza a revisão de imagens arquivadas, mas também melhora a capacidade de resposta a incidentes em tempo real.

### Tarefa 1: Detecção de Anomalias

A detecção de anomalias desempenha um papel crucial na segurança, permitindo a identificação de comportamentos suspeitos que fogem do padrão normal. Algoritmos de visão computacional são treinados para reconhecer padrões incomuns e alertar os operadores de segurança sobre possíveis ameaças. Essa capacidade de identificar atividades não usuais em tempo real é essencial para prevenir incidentes e garantir a segurança de ambientes monitorados por câmeras de segurança.

### Tarefa 2: Detecção de Eventos

A detecção de eventos envolve a identificação e classificação de ações específicas que são de interesse para a segurança, como agressões, furtos ou invasões. Por meio de algoritmos de CV, é possível treinar sistemas de vigilância para reconhecer automaticamente esses eventos e acionar alertas quando ocorrem. Essa capacidade de detecção de eventos em tempo real é fundamental para a tomada de decisões rápidas e eficazes em situações de emergência.

### Tarefa 3: Sumários de Vídeo de Vigilância

A sumarização automática de vídeos é uma técnica essencial para condensar grandes volumes de imagens de vídeo em resumos concisos e informativos. Esses sumários destacam as atividades mais relevantes e eliminar informações redundantes, permitindo aos operadores de segurança revisar rapidamente horas de gravações em busca de eventos importantes. Algoritmos avançados de CV são empregados para criar esses sumários de forma eficiente e precisa, melhorando a eficácia da vigilância por vídeo.

### Tarefa 4: Pesquisas Direcionadas por Consulta em Arquivos de Vídeo Longos

As pesquisas direcionadas por consulta permitem aos operadores de segurança buscar rapidamente segmentos de vídeo com características específicas, como a presença de indivíduos armados ou atividades suspeitas. Algoritmos de CV são aprimorados com informações fornecidas pelo usuário para melhorar a precisão das pesquisas e garantir a recuperação eficiente de dados relevantes em arquivos de vídeo extensos. Essa capacidade de pesquisa direcionada é essencial para investigações detalhadas e identificação de eventos críticos.

## Conclusão

A monitorização automatizada para redes de câmeras de segurança, utilizando visão computacional, representa uma abordagem inovadora e eficaz para melhorar a segurança e a eficiência da vigilância por vídeo. A aplicação de algoritmos avançados de CV nas tarefas de detecção de anomalias, eventos, sumarização de vídeos e pesquisas direcionadas por consulta demonstra o potencial dessa tecnologia para aprimorar a segurança em ambientes monitorados. Com a contínua evolução da visão computacional, espera-se que essas técnicas se tornem ainda mais sofisticadas e eficientes, contribuindo para a proteção e o monitoramento eficaz de espaços públicos e privados.

## 2 - Sensors and Early Vision:

## 2.1- Current State of Hyperspectral Remote Sensing for Early Plant Disease Detection: A Review

## Introdução:

A introdução discute a importância dos métodos de monitoramento remoto para resolver problemas relevantes na agricultura moderna, como o controle de pragas e doenças. Esses métodos têm potencial para fornecer informações úteis para práticas de manejo agrícola

específicas. A introdução também menciona a existência de dois tipos de tecnologias de monitoramento remoto: passivas (como ópticas) e ativas (como LiDAR e radar). As tecnologias passivas ópticas são geralmente divididas em duas categorias com base na resolução espectral dos sensores utilizados: multiespectral e hiperespectral. A resolução hyperspectral mostra potencial como uma ferramenta não invasiva e não destrutiva para monitorar o estresse biológico e abiótico entre as tecnologias de monitoramento remoto passivo.

Essa técnica coleta e armazena informações do espectro de um objeto em um cubo de dados que contém informações espaciais e centenas de comprimentos de onda contíguos na terceira dimensão. A imagem hiperespectral com centenas de bandas espectrais pode fornecer retratos espectrais detalhados, permitindo a detecção de variações sutis no solo, nas copas ou em folhas individuais. Portanto, as imagens hiperespectrais podem ser usadas para resolver uma classe mais ampla de problemas para a determinação precisa e oportuna do estado fisiológico das culturas agrícolas.

A detecção precoce da propagação de doenças e surtos de pragas pode não apenas evitar perdas significativas de colheitas, mas também reduzir o uso de pesticidas e mitigar seus impactos negativos na saúde humana e no meio ambiente, melhorando assim o manejo integrado de pragas existentes.

## Materials and Methods:

Esta seção discute o uso do sensoriamento remoto hiperespectral (HRS) na detecção de doenças em dendezeiros, o que é importante para o controle de pragas e doenças na indústria do óleo de dendê. A doença mais significativa é a podridão basal do caule (BSR) causada por Ganoderma boninense, que pode reduzir a produtividade em 80%. Estudos demonstraram que o HRS pode discriminar entre dendezeiros saudáveis e infectados com BSR com alta precisão, usando várias técnicas, como PLS-DA, índices de vegetação e técnicas baseadas em red edge. A pesquisa também se concentrou no desenvolvimento de índices de vegetação espectral (SVIs) para detecção precoce de BSR em mudas de dendê.

Outra doença do dendezeiro discutida é a mancha laranja (OS), causada pelo viróide cadang-cadang do coco (CCCV), que matou milhões de coqueiros nas Filipinas. Embora atualmente seja de menor importância nos dendezeiros, estudos mostraram que o HRS pode detectar OS em dendezeiros sintomáticos usando vários SVIs.

A Tabela 1 resume os estudos sobre detecção precoce de doenças do dendê usando SHR, incluindo ano de publicação, cultura, tratamento, equipamentos utilizados, faixas estudadas, faixas importantes, tipo de estudo, referência e localização.

**Table 1.** Oil palm disease early detection by HRS.

| Publication Year | Culture | Treat | Equipment | Studied Bands | Important Bands | Study Type | Reference | Location |
|---|---|---|---|---|---|---|---|---|
| 2009 | oil palm | basal stem rot | APOGEE spectroradiometer of unmentioned model | 450–1100 | 715, 734, 791 | field | [52] | Malaysia |
| 2009 | oil palm | basal stem rot | APOGEE spectroradiometer of unmentioned model | 300–1000 | 462, 487, 610.5, 738, 749 | field | [53] | Malaysia |
| 2010 | oil palm | basal stem rot | PP Systems Unispec-SC spectrometer | 310–1130 | 670–715, 490–520, 730–770, 920–970 | field | [50,51] | Indonesia |
| 2011 | oil palm | basal stem rot | APOGEE spectroradiometer of unmentioned model | 350–1000 | 495, 495.5, 496, 651.5, 652, 652.5, 653, 653.5, 654, 654.5, 655, 655.5, 656, 656.5, 657, 657.5, 658, 658.5, 659, 659.5, 660, 660.5, 661, 908 | field | [55] | Malaysia |
| 2014 | oil palm | basal stem rot | ASD spectrometer of unmentioned model | 325–1040 | not mentioned | field | [58] | Malaysia |
| 2017 | oil palm | basal stem rot | APOGEE spectroradiometer of unmentioned model | 325–1000 | 495, 495.5, 496, 651.5, 652, 652.5, 653, 653.5, 654, 654.5, 655, 655.5, 656, 656.5, 657, 657.5, 658, 658.5, 659, 659.5, 660, 660.5, 661, 908 | field | [56] | Malaysia |
| 2017 | oil palm | basal stem rot | GER 1500 spectrometer | 273–1100 | 540–560, 650–780 | field | [59] | Malaysia |
| 2018 | oil palm | basal stem rot | Specim spectrograph of unmentioned model | 350–1000 | 650–750 | field | [57] | Malaysia |
| 2020 | oil palm | basal stem rot | Cubert S185 camera | 325–1075 | 800–950 | greenhouse | [60] | Malaysia |
| 2014 | oil palm | orange spotting | ASD FieldSpec 4 spectrometer | 300–1050 | 400–401, 404–405, 455–499, 500–599, 600–699, 700–712 | field | [63,64] | Malaysia |
| 2019 | oil palm | orange spotting | ASD HandHeld 2 spectrometer | 400–1050 | 601–630 | field | [36] | Malaysia |
| 2019 | oil palm | orange spotting | ASD HandHeld 2 spectrometer | 325–1075 | 680–780 | field | [65,66] | Malaysia |

Além disso, a seção menciona brevemente o uso do HRS na detecção de doenças dos citros, como o cancro bacteriano dos citros (CBC) e a doença do greening dos citros (HLB), que são ameaças significativas à indústria citrícola. No entanto, o foco da seção está nas doenças do dendezeiro.

## Discussão:

No tópico de discussão do artigo, os autores destacam a importância de considerar os fatores abióticos na análise de estresse em plantas usando sensores hiperespectrais. Eles afirmam que não existe uma metodologia unificada para estudos hiperespectrais de doenças

em plantas que leve em conta a influência desses fatores. Por esse motivo, sugerem a realização de experimentos em condições controladas de laboratório ou em estufas industriais para eliminar ou reduzir essas influências.

Além disso, os autores enfatizam a importância de indicar a cultura e o cultivo das plantas estudadas, bem como o patógeno utilizado para a inoculação. Eles também discutem a possibilidade de usar sensores hiperespectrais para detectar outras doenças em plantas além das já mencionadas no artigo, como a mancha laranja e a podridão da base do caule.

Outro ponto importante abordado na discussão é a influência da fluorescência da clorofila nos espectros de plantas e seus índices relacionados, que podem ser uma contribuição significativa para a solução do problema de detecção precoce de doenças em plantas. Os autores também mencionam a possibilidade de estudar a influência das características genotípicas de um patógeno no perfil espectral de uma planta infectada.

Por fim, os autores discutem a importância dos efeitos das mudanças bioquímicas nas tecidos vegetais para a detecção precoce de doenças em plantas usando sensores passivos. Eles destacam que a refletância de luz em folhas de plantas é dependente de múltiplas interações biofísicas e bioquímicas, e que a detecção de elementos químicos individuais ou compostos químicos, incluindo compostos voláteis, em plantas usando HRS pode ser crucial para a identificação precoce de doenças em plantas. No entanto, essa tarefa é difícil e pouco estudada.

## Conclusão:

Na seção de conclusão do artigo, os autores reafirmam a possibilidade de detecção precoce de doenças em plantas usando hyperspectral remota sensoriamento (HRS). No entanto, eles também identificam algumas lacunas metodológicas e técnicas que precisam ser abordadas para garantir a replicabilidade dos experimentos.

Os autores sugerem que as diferenças nos espectros de plantas podem ser causadas por uma variedade de fatores abióticos e bióticos que causam estresse em plantas. Além disso,

o fenótipo ou genótipo da planta hospedeira pode influenciar a manifestação da doença, dependendo do nível de resistência da planta. A presença de infecção mista também pode ser um fator importante que influencia o espectro da planta.

Em termos técnicos e físicos, os autores destacam a importância de considerar o modelo de propagação da luz solar durante experimentos de campo ou as características das fontes de luz artificial usadas em experimentos de laboratório, uma vez que o HRS é um método de sensoriamento remoto passivo que depende das condições da fonte de luz externa. Outros problemas técnicos podem incluir o uso incorreto do equipamento e a necessidade de calibrar adequadamente o sensor ou câmera hiperespectral para coleta de dados precisos.

Os autores concluem que a criação de bancos de dados de retratos hiperespectrais de plantas de diferentes culturas e cultivares expostas a diferentes patógenos deve ser adiada até que os princípios gerais de detecção remota de doenças em plantas usando HRS sejam desenvolvidos. Eles acreditam que a solução do problema da detecção precoce de doenças em plantas usando HRS requer a colaboração interdisciplinar de especialistas em fisiologia vegetal, fitopatologia, resistência vegetal, genômica, bioinformática, tecnologias da informação, análise de sistemas e óptica ou fotônica.

## 2.2 - RedEye: Analog ConvNet Image Sensor Architecture for Continuous Mobile Vision

### Introdução:

A recente emergência de dispositivos wearables trouxe a ideia de "mostrar aos computadores o que você vê", ou visão contínua móvel. No entanto, a barreira para a eficiência energética é desafiadora. Tarefas de visão contínua esgotam a bateria do Google Glass em 40 minutos. A melhoria da eficiência energética dos circuitos digitais por meio de tecnologia de processo e otimização de nível de sistema pode continuar, mas medições recentes apontam para um gargalo fundamental na eficiência energética: o sensor de imagem, especialmente sua circuiteria de leitura analógica.

Esse gargalo analógico é fundamental por duas razões. Em primeiro lugar, em comparação com os circuitos digitais, a eficiência energética da leitura analógica melhora muito mais lentamente ao longo das gerações tecnológicas. Em segundo lugar, enquanto a tendência na comunidade de circuitos de estado sólido é mover funcionalidades do domínio analógico para o digital, a leitura analógica será necessariamente existente como uma ponte do mundo analógico físico para o reino digital.

Para abordar esse gargalo, nossa ideia-chave é empurrar o processamento antecipado no domínio analógico para reduzir a carga de trabalho da leitura analógica, resultando em nosso design do RedEye, um sensor de imagem convolutivo analógico para visão móvel contínua. RedEye descarta dados crus, exportando recursos gerados por processamento de uma Rede Convolucional (ConvNet) no domínio analógico. Isso economiza energia não apenas na leitura analógica do sensor, mas também em todo o sistema.

O design do RedEye aborda os seguintes desafios importantes: (i) os circuitos analógicos sofrem de alta complexidade de design, desencorajando a portabilidade entre nós de tecnologia de processamento. RedEye restringe a complexidade com um design modular que facilita a reutilização física dentro de um design de circuito e reutilização algorítmica por meio de fluxo de dados cíclico. (ii) A processamento analógica está sujeita a acúmulo de ruído, o que afeta a precisão da tarefa. RedEye identifica os principais compromissos entre ruído-energia para sacrificar energia para amenizar o ruído durante o processamento. RedEye fornece mecanismos para ajustar parâmetros de ruído para eficiência e precisão em tempo de execução. (iii) Os desenvolvedores que usam o RedEye requerem uma estimativa da precisão da tarefa da ConvNet e do consumo de energia sob a influência do ruído. Fornecemos um framework de simulação para ajudar os desenvolvedores a ajustar o equilíbrio de precisão e eficiência ao executar ConvNets no RedEye.

## Background:

A. Image sensing

Dispositivos móveis modernos usam sensores de imagem CMOS para fotografia e videografia. Embora mais eficientes que CCD, eles ainda consomem centenas de miliwatts devido ao seu analógico oneroso leitura [4].

Os sensores de imagem modernos consistem em três componentes principais: matriz de pixels, leitura analógica e lógica digital. A matriz de pixels converte a luz em elétrons e depois em tensões, consumindo pouca energia [5]. A leitura analógica amplifica sinais analógicos da matriz de pixels e os converte em valores digitais. A lógica digital mantém a operação do sensor, gerenciando a geração de tempo, scanner e motoristas de quadro e interface de dados. Os sensores modernos empregam uma arquitetura de coluna para

leitura analógica; pixels em cada coluna compartilham circuitaria dedicada. Amplificadores de coluna amostram e amplificam pixels crus para aproveitar o alcance de sinal disponível e superar o ruído; eles consomem energia estática para polarizar os transistores operando linearmente. Conversores A/D de coluna digitalizam sinais amplificados, consumindo energia estática e dinâmica. A leitura analógica consome 50%-75% da energia total em designs de sensores recentes [5]-[7]. Isso porque (i) todo o sinal de imagem deve passar por leitura analógica e (ii) leitura analógica é projetada para alta fidelidade.

Destacamos que a otimização digital, como hardware de baixa potência para redução de dados [8], descarte antecipado ou computação offloading, não afeta a energia consumida pela leitura analógica. Isso é o fundamento da nossa proposta para mover a processamento antes leitura analógica.

B. Eficiência de computação analógica

A computação analógica é mais eficiente do que a computação digital para operações básicas, como adição e multiplicação, com a energia por operação de implementações analógicas sendo ordens de magnitude menor [9]. Ao contrário da computação digital, que usa várias cargas binárias (bits) para representar o estado do circuito, a computação analógica usa uma única carga para representar um valor. Isso reduz a contagem de hardware; adição analógica só precisa de um interconexão para juntar duas correntes, enquanto um adder digital de 16 bits requer mais de 400 transistores.

Mais importante, trocas lucrativas de energia, eficiência e fidelidade de sinal podem ser feitas no domínio analógico, relevantes para tarefas de visão resistentes ao ruído. Isso porque o mantendo estado através da capacitância C é suscetível ao ruído térmico V 2 n = kT /C [12]. Embora aumentar C desencoraje o ruído, a energia necessária para encher a célula de memória aumenta proporcionalmente. Portanto, a escolha de C é um compromisso fundamental entre a fidelidade e a eficiência da computação analógica. Aqui nós identificamos os compromissos de energia-ruído inerentes à computação analógica.

Memória: Como um pipeline analógico deve ser construído em estágios para facilitar a configurabilidade e manutenção, a memória analógica é indispensável para buffers inter-estágio. As células de memória usam capacitores para manter estados e, portanto, exibem compromissos de energia-ruído na leitura e gravação de valores.

Aritmética: A aritmética em circuitos de carga, como em RedEye, geralmente requer um amplificador operacional(op amp), que induz ruído térmico $\propto 1/C$ e energia $\propto C$.

C. ConvNets

Convolutional Neural Networks (ConvNets) são usados para tarefas de visão contínua em RedEye devido à sua eficácia em detecção de objetos, reconhecimento de escrita manual e

detecção facial [18]. Essa proeminência resultou em pesquisa acadêmica e industrial ativa na implementação de ConvNets para tarefas de visão.

ConvNets opera processando dados tridimensionais através de camadas sequenciais de processamento para gerar recursos de visão. Cada camada consiste em um conjunto de neurônios; cada neurônio recebe múltiplos valores de entrada de um campo receptivo local de neurônios da camada anterior. O neurônio opera nestes valores para gerar uma única saída. Camadas de neurônios vêm em uma pequena variedade de tipos. Aqui cobrimos os principais tipos de camadas:

Neurônios de camada convolucional multiplicam um campo receptivo tridimensional de entradas com um kernel de pesos para gerar uma saída. Neurônios em uma camada convolucional compartilham pesos de kernel para executar seu processamento. Conjuntos múltiplos de pesos por camada geram canais de profundidade de dados. Um processo de retropropagação treina pesos para alterar a operação de convolução.

Neurônios da camada de não linearidade usam funções de ativação para introduzir saturação não linear dos outputs de camadas convolucionais. Funções sigmóide, tangente hiperbólica e funções de retação foram usadas para criar não linearidade.

Neurônios da camada de pooling máximo recebem recursos gerados de um campo receptivo de saída processada e identificam o recurso com a resposta máxima. Isso permite que a rede seja robusta a pequenas mudanças no espaço de recursos.

Essas camadas, além de outras camadas, como camadas de normalização para restringir a amplitude do sinal, podem ser encadeadas em ordem sequencial variada, invocando cada camada várias vezes para criar estruturas profundas. Por exemplo, a rede de reconhecimento de imagem ConvNet AlexNet usa 6 camadas convolucionais, 7 camadas de não linearidade, 3 camadas de pooling, 2 camadas de normalização. AlexNet também inclui 7 "outras" camadas no final de sua operação.

Devido à generalização de Redes treinadas ConvNets em entradas de imagem, eles são naturalmente robustos ao ruído [20]. Nós estudamos e exploramos a robustez para os compromissos de energia-ruído em RedEye.

## Redeye System Architecture (III. REDEYE SYSTEM ARCHITECTURE)

O terceiro tópico descreve a arquitetura do sistema redeye. O sistema redeye consiste em um coprocessador com processadores específicos e um sistema de memória altamente programável (HPMs). O coprocessador redeye é projetado para executar operações de CNN

específicas de maneira eficiente. Ele contém processadores específicos, como um processador de ponto fixo (PFP) para processamento de matrizes, e um processador de ponto flutuante (FPP) para processamento de pontos flutuantes. Além disso, o sistema redeye utiliza um sistema de memória altamente programável (HPMs) para armazenar dados de imagem e pesos de rede.

## Redeye Circuit Design (IV. REDEYE CIRCUIT DESIGN)

O quarto tópico aborda o design do circuito redeye. O design do circuito redeye é projetado para ser compacto, energeticamente eficiente e com alto desempenho. Ele utiliza um design de soma de produtos de múltiplos portadores (MCCA) para o processamento de matrizes. O design do circuito também utiliza uma topologia de árvore de pesquisa de intervalos para acelerar o processamento de pontos flutuantes. Além disso, o circuito redeye utiliza técnicas de baixa potência, como gerenciamento de tensão de alta e baixa velocidade e multiplexação de alimentação.

## Evaluation (V. EVALUATION)

O quinto tópico avalia o desempenho do sistema redeye em comparação com a solução baseada em FPGA (Field Programmable Gate Array) existente. O desempenho do sistema redeye é avaliado em termos de área de silício, consumo de energia, desempenho e eficiência. Os resultados mostram que o sistema redeye oferece um desempenho melhor em relação à solução baseada em FPGA em termos de desempenho e eficiência.

## Related Work (VI. RELATED WORK)

O sexto tópico apresenta as técnicas e soluções relacionadas à otimização do processamento de imagens em dispositivos portáteis. Ele menciona algoritmos de comutação rápida de alta velocidade, como o FPGA (Field Programmable Gate Array) e o ASIC (Application-Specific Integrated Circuit), e arquiteturas de baixa potência, como o HPMs (Highly Programmable Memories). Além disso, o tópico menciona técnicas de otimização de software, como a otimização de código e a otimização de memória.

## Concluding Remarks (VII. CONCLUDING REMARKS)

O sétimo tópico resume os principais resultados do artigo e apresenta as conclusões. O artigo propõe uma abordagem para o desenvolvimento de um coprocessador chamado "redeye" para acelerar o processamento de imagens em CNNs. O sistema redeye é projetado para ser compacto, energeticamente eficiente e com alto desempenho. Os resultados mostram que o sistema redeye oferece um desempenho melhor em relação à solução baseada em FPGA em termos de desempenho e eficiência. Além disso, o artigo menciona as técnicas e soluções relacionadas à otimização do processamento de imagens em dispositivos portáteis.

## 3 - Machine Learning Technologies for Vision:

## 3.1 - Machine Learning in Computer Vision Asharul Islam Khan a*, Salim Al-Habsib

## Introdução:

A introdução destaca a crescente importância do aprendizado de máquina e da visão computacional, visando capacitar os computadores com habilidades humanas de detecção e compreensão de dados. Essas tecnologias têm aplicabilidades variadas, desde a Internet das Coisas até interfaces cerebrais humanas. Métodos como aprendizado supervisionado, não supervisionado e semi-supervisionado são fundamentais, utilizando algoritmos como máquinas de vetores de suporte e KNN. Empresas oferecem soluções de reconhecimento de fala, análise de texto e classificação de imagens, acessíveis através de APIs. A detecção e análise de objetos são cruciais para diversas aplicações, como evitar colisões de trânsito e reconhecimento emocional.

Tecnologias como Tensor Flow e OpenPose são utilizadas para detecção de objetos. O aprendizado supervisionado de redes neurais convolucionais é destacado, juntamente com o desafio da anotação de dados. O serviço de aprendizado de máquina em nuvem é uma tendência, com empresas como Amazon, Microsoft e Google oferecendo plataformas para tal. O objetivo da pesquisa é analisar aplicativos de aprendizado de máquina em visão computacional, com base em uma revisão bibliográfica que resultou em 20 artigos selecionados para este estudo. O trabalho é dividido em cinco seções, abrangendo estudo de base, categorização de aplicativos existentes, resultados, discussões e conclusões.

## Background study

A visão computacional e o aprendizado de máquina são duas áreas importantes de pesquisa recente. A visão computacional utiliza mapeamento de imagem e padrões para encontrar soluções, considerando uma imagem como uma matriz de pixels. Automatiza tarefas de monitoramento, inspeção e vigilância. O aprendizado de máquina, um subconjunto da inteligência artificial, resulta na análise/rotulagem automática de vídeos. Existem três abordagens para aprendizado de máquina e visão computacional: supervisionado, não supervisionado e semi-supervisionado. O aprendizado supervisionado requer dados de treinamento rotulados, enquanto o semi-supervisionado possui alguns dados rotulados e outros não.

Problemas do mundo real muitas vezes se encaixam na categoria de aprendizado não supervisionado, onde padrões evoluem com base em agrupamento. Os paradigmas de aprendizado de máquina para visão computacional incluem máquinas de vetores de suporte, redes neurais e modelos gráficos probabilísticos. As Redes Neurais Convolucionais (CNNs) têm ganhado popularidade devido a conjuntos de dados acessíveis, GPUs e técnicas de regularização. A biblioteca OpenCV é integrada com diversas linguagens de programação e plataformas para processamento de imagem, análise de vídeo, detecção de objetos e aprendizado de máquina.

## Aprendizado de Máquina em Visão Computacional:

O estudo explorou diversas aplicações do aprendizado de máquina na visão computacional, abrangendo áreas como segmentação, extração de características, refinamento de modelos visuais, correspondência de padrões, representação de formas, reconstrução de superfícies e modelagem para ciências biológicas.

Algumas das aplicações incluem interpretação de dados em imagens de detecção de carros e pedestres, classificação automática de falhas em trilhos ferroviários, diferenciação de variedades de manga com base em atributos de tamanho, extração de informações gráficas e textuais de imagens de documentos, reconhecimento de gestos e rostos, visão de máquina, reconhecimento de caracteres e dígitos manuscritos, sistemas avançados de assistência ao motorista, estudos comportamentais, e estimativa de cinemática corporal completa humana para um ciclista e estimativa de pose.

Além disso, o estudo aborda aplicações em medicina, como detecção de hemorroidas, detecção de sangramento, aprimoramento de imagens endoscópicas e suporte à decisão clínica. Outras áreas de aplicação incluem ciências biológicas, meteorologia, detecção e contagem de tráfego, análise de desempenho em esportes, avaliação do estado de ferramentas de máquinas, manutenção preditiva e muito mais. As aplicações são detalhadas e agrupadas em uma tabela para facilitar a compreensão e o acesso.

## Resultados e Discussão:

Os resultados e discussões deste estudo destacam a crescente importância do aprendizado de máquina e da visão computacional em diversas áreas. As técnicas de aprendizado de máquina e visão computacional reduziram custos, esforços e tempo em engenharia, ciência e tecnologia. Um sistema automatizado baseado em aprendizado de máquina e visão computacional detecta as emoções humanas, prevê atividades humanas por meio de rotulagem e reconhecimento de padrões, e analisa o desempenho de equipes e jogadores individuais em esportes profissionais. Além disso, essas técnicas são utilizadas em indústrias para manutenção preditiva, o que impacta significativamente a eficácia e eficiência das unidades de fabricação. A visão computacional e o aprendizado de máquina aplicados em dados de câmeras públicas e dispositivos inteligentes ajudam na previsão e monitoramento do tráfego nas cidades. Os resultados mostram que as áreas de pesquisa avançadas incluem ciências biológicas, atividade humana, gestão de tráfego e esportes profissionais. A evolução do aprendizado de máquina, desde métodos tradicionais de reconhecimento de padrões até técnicas avançadas de compreensão de imagens, contribui para a dinâmica em mudança dos sistemas de visão computacional. Embora a visão computacional possa interpretar e extrair informações de áudio e vídeo de forma independente, o aprendizado de máquina adiciona a capacidade preditiva dos dados já processados. As aplicações de aprendizado de máquina em visão computacional têm saídas variadas dependendo do domínio, e os principais temas de pesquisa incluem classificação de objetos, detecção de objetos e reconhecimento de sequências. No entanto, a detecção e rastreamento de objetos ainda representam um desafio aberto na visão computacional. A qualidade da saída do aprendizado de máquina depende da precisão preditiva, recall e precisão.

## Conclusão:

O estudo conclui que a pesquisa comercial e acadêmica em visão computacional está crescendo, e a síntese entre aprendizado de máquina e visão computacional tem sido

fundamental para entender problemas complexos. O futuro trabalho incluirá a avaliação da precisão dos algoritmos de aprendizado de máquina em visão computacional.

## 3.2 - Deep learning for consumer devices and services: Pushing the limits for machine learning, artificial intelligence, and computer vision
## Lemley, Joseph; Bazrafkan, Shabab; Corcoran, Peter

### Introdução:

O texto introduz o conceito de Deep Learning, destacando seu rápido crescimento devido à disponibilidade de grandes conjuntos de dados e hardware especializado. Ele destaca a aplicabilidade do Deep Learning em uma ampla gama de problemas, incluindo a melhoria de dispositivos e serviços de consumo. Além disso, o texto explica os fundamentos das redes neurais artificiais (ANN), incluindo sua capacidade de aprendizado e generalização.

Ele discute o treinamento de redes neurais, abordando conceitos como underfitting e overfitting, além de métodos de avaliação de modelos. O texto também diferencia entre aprendizado supervisionado e não supervisionado e explora a estrutura interna das redes neurais, descrevendo camadas, neurônios e mecanismos de conexão. Finalmente, ele destaca o recente renascimento das redes neurais devido a avanços tecnológicos e ao crescimento do Big Data.

### Redes Neurais Convolucionais

As Redes Neurais Convolucionais (CNN) processam sinais analógicos, como imagens e sons, convertendo-os para o formato digital através da conversão analógico-digital (A2D). Elas operam em espaços dimensionais altos, representando amostras em um espaço de características. As CNNs utilizam camadas de convolução, que aprendem filtros durante o treinamento para extrair características importantes dos dados. Essas camadas aplicam o operador de convolução, que compara o sinal de entrada com um filtro para detectar padrões.

As CNNs também incluem camadas de pooling, que reduzem a dimensionalidade dos dados e fornecem invariância a transições. Além disso, as CNNs empregam funções de ativação não-lineares para modelar sistemas não-lineares. As arquiteturas de CNNs podem incluir camadas densas completamente conectadas após as camadas de convolução, formando redes neurais profundas genéricas. Outras arquiteturas de destaque são as Redes Neurais Totalmente Convolucionais (FCN), que não possuem camadas densas, os Autoencoders, que aprendem representações comprimidas dos dados, e as Redes Neurais Recorrentes (RNN), projetadas para dados sequenciais.

## Estado da Arte Hoje:

As tecnologias habilitadoras principais estão impulsionando o avanço das redes neurais profundas e suas aplicações. As GPUs (unidades de processamento gráfico) desempenham um papel crucial no treinamento dessas redes, sendo significativamente mais rápidas que as CPUs para tarefas altamente paralelizáveis, como operações matriciais. Enquanto as CPUs são otimizadas para problemas sequenciais, as GPUs são projetadas para processamento gráfico paralelo, o que as torna ideais para deep learning. Com centenas ou milhares de núcleos, as GPUs treinam redes neurais profundas em uma magnitude de ordem mais rápida do que as CPUs. A série Pascal da NVIDIA representa uma melhoria significativa no desempenho e no consumo de energia em comparação com modelos anteriores.

As GPUs podem ser combinadas para treinamento mais rápido, mas a escolha do dispositivo depende da memória e do número de núcleos necessários para a tarefa. Além das GPUs, surgem dispositivos de hardware personalizados para deep learning, como o "fathom neural compute stick" da Movidius, que consome significativamente menos energia que uma GPU equivalente. Este tipo de dispositivo é ideal para implementações em objetos inteligentes, como robôs e drones.

O software também desempenha um papel fundamental no desenvolvimento e na implementação de modelos de deep learning. Frameworks como Caffe, Theano e TensorFlow simplificam o design e a implantação de redes neurais profundas, oferecendo abstrações de alto nível e facilitando o desenvolvimento. O TensorFlow, desenvolvido pelo Google, está crescendo rapidamente em popularidade devido à sua facilidade de uso em múltiplas GPUs.

Essas tecnologias permitiram uma ampla gama de aplicações práticas, desde tradução visual instantânea até reconhecimento facial, passando por reconhecimento de voz e transformação de fotos em obras de arte. Empresas como Google, Nuance e Apple estão incorporando deep learning em seus produtos para melhorar a experiência do usuário e oferecer novos recursos, como reconhecimento de voz e imagem em tempo real. Além disso, frameworks como Keras tornam o desenvolvimento acessível mesmo para iniciantes, permitindo que uma ampla variedade de problemas seja abordada por meio de deep learning.

Em resumo, o estado atual da arte em deep learning é caracterizado por uma combinação de hardware avançado, software de alto nível e uma ampla gama de aplicações práticas, impulsionando o campo em direção a novos avanços e inovações em uma variedade de domínios.

## Conclusão:

Hoje, a aprendizagem profunda está sendo usada em nossos celulares, em nossos carros, em tablets e computadores. Ela ampliou os limites do que é possível para tarefas como segmentação de imagem, detecção de objetos, reconhecimento facial, análise de voz, detecção de emoções e reconhecimento de gênero.

Por que a aprendizagem profunda de repente catalisou pesquisas em tantos campos? Bem, é uma combinação de muitos fatores: o surgimento recente de hardware computacional baseado em GPU altamente acessível e denso forneceu os meios para processar conjuntos de dados muito grandes e implementar as metodologias de treinamento avançadas necessárias para desenvolver CNNs precisas; a disponibilidade generalizada de GPUs em dispositivos de hoje, combinada com serviços de processamento de dados baseados em nuvem, fornece os meios para aplicar essas arquiteturas de CNN a aplicativos cotidianos, como processamento de voz ou imagem. Grandes conjuntos de dados fornecem o combustível para impulsionar a atividade de pesquisa e refinar resultados até o ponto em que as soluções de aprendizagem profunda geralmente superam até mesmo as melhores ferramentas de reconhecimento de padrões projetadas pelo homem.

Hoje, estamos em um ponto em que muitos novos problemas podem ser abordados por meio de técnicas de aprendizagem profunda - muitos desses são problemas antigos, como reconhecimento de voz, que nunca foram bons o suficiente para sair do laboratório e entrar no uso diário. Mas este ano vimos vários lançamentos de 'alto-falantes inteligentes' que podem controlar sua casa - e por trás desses novos dispositivos estão uma série de tecnologias de aprendizagem profunda - analisando suas necessidades, traduzindo suas solicitações de voz e coordenando a logística necessária para entregar tudo em sua porta ou na sua TV.

Para começar a resolver seu próprio problema, você precisará de uma GPU de ponta - a mesma tecnologia usada nos últimos PCs de jogos - e a maioria do software principal está disponível gratuitamente na Internet. Existem vários pacotes de software em tendência no campo da aprendizagem profunda, incluindo, mas não se limitando a Theano (em Python), Lasagne (em Theano), Tensor Flow (em Python e C++), Caffe (em Python e MATLAB) e MatConvNet (em MATLAB).

# 4 - Object Recognition

## 4.1 - Research and Application of Visual Object Recognition System Based on Deep Learning and Neural Morphological Computation Le Yang1, *, Han Wang2, Jiajian Zheng3, Xin Duan4, Qishuo Cheng5

### Introdução:

A introdução destaca os desafios enfrentados pelos sistemas de computação tradicionais, baseados em arquiteturas de Von Neumann, como limitações na velocidade de processamento e aumento dramático no consumo de energia devido à separação entre unidades de processamento e armazenamento. Os pesquisadores propõem explorar a computação neuromórfica baseada em sinapses artificiais como uma solução promissora para esses desafios. Essa abordagem oferece vantagens significativas, como alta velocidade de processamento e baixo consumo de energia. O foco do artigo está na aplicação dessas tecnologias no reconhecimento visual de objetos, destacando os avanços recentes em visão computacional e deep learning.

### Trabalhos Relacionados:

Esta seção explora o conceito de computação neuromórfica e sua aplicação em redes neurais profundas (DNNs) para reconhecimento de padrões em grandes conjuntos de dados. Os autores discutem o desenvolvimento de hardware neuromórfico, incluindo memristores e sistemas VLSI, e como essas tecnologias estão sendo combinadas com redes neurais profundas para melhorar a eficiência de tarefas como reconhecimento de imagens e classificação. São abordadas as limitações históricas da computação neuromórfica, bem como os avanços recentes que a tornam uma opção mais viável.

### Metodologia:

O artigo descreve detalhadamente a metodologia utilizada para investigar o comportamento resistivo digital e analógico de folhas de WSe2 em memristores, juntamente com a simulação de funções sinápticas básicas e computação neuromórfica. Métodos de caracterização, como microscopia eletrônica de varredura (SEM), espectroscopia Raman e espectroscopia de fotoelétrons de raios-X (XPS), são empregados para analisar as propriedades físicas das folhas de WSe2. O processo de crescimento de nano lâminas de WSe2 é detalhado, assim como os resultados da caracterização por diferentes técnicas.

## Conclusão:

Os autores concluem destacando as perspectivas promissoras da integração entre computação neuromórfica e deep learning em sistemas de reconhecimento visual de objetos. Eles enfatizam como essa integração pode melhorar a eficiência e a precisão dos sistemas de reconhecimento visual, acelerando o progresso da tecnologia de visão computacional. Além disso, apontam que essa integração estabelece uma base sólida para o desenvolvimento de sistemas inteligentes de percepção e cognição. Os autores sugerem que futuras pesquisas devem explorar ainda mais a interseção entre computação neuromórfica e deep learning para avançar ainda mais a área de reconhecimento visual de objetos.

# Notebook Rodrigo Jupyter : Processamento de imagens:

Bem-vindo ao nosso notebook introdutório sobre processamento de imagens com diversas bibliotecas, incluindo o OpenCV, Pillow e scikit-image! Estas bibliotecas oferecem uma ampla gama de funcionalidades para manipulação, análise e processamento de imagens em Python. Neste tutorial, exploraremos algumas das funcionalidades básicas dessas bibliotecas, desde a leitura e exibição de imagens até transformações e detecção de recursos. Vamos mergulhar e descobrir como essas bibliotecas podem ser ferramentas poderosas para trabalhar com imagens em projetos de visão computacional e processamento de imagens.

## Introdução ao OpenCV

In [1]:

```
import cv2

print("OpenCV Version:", cv2.__version__)
```

OpenCV Version: 4.8.0

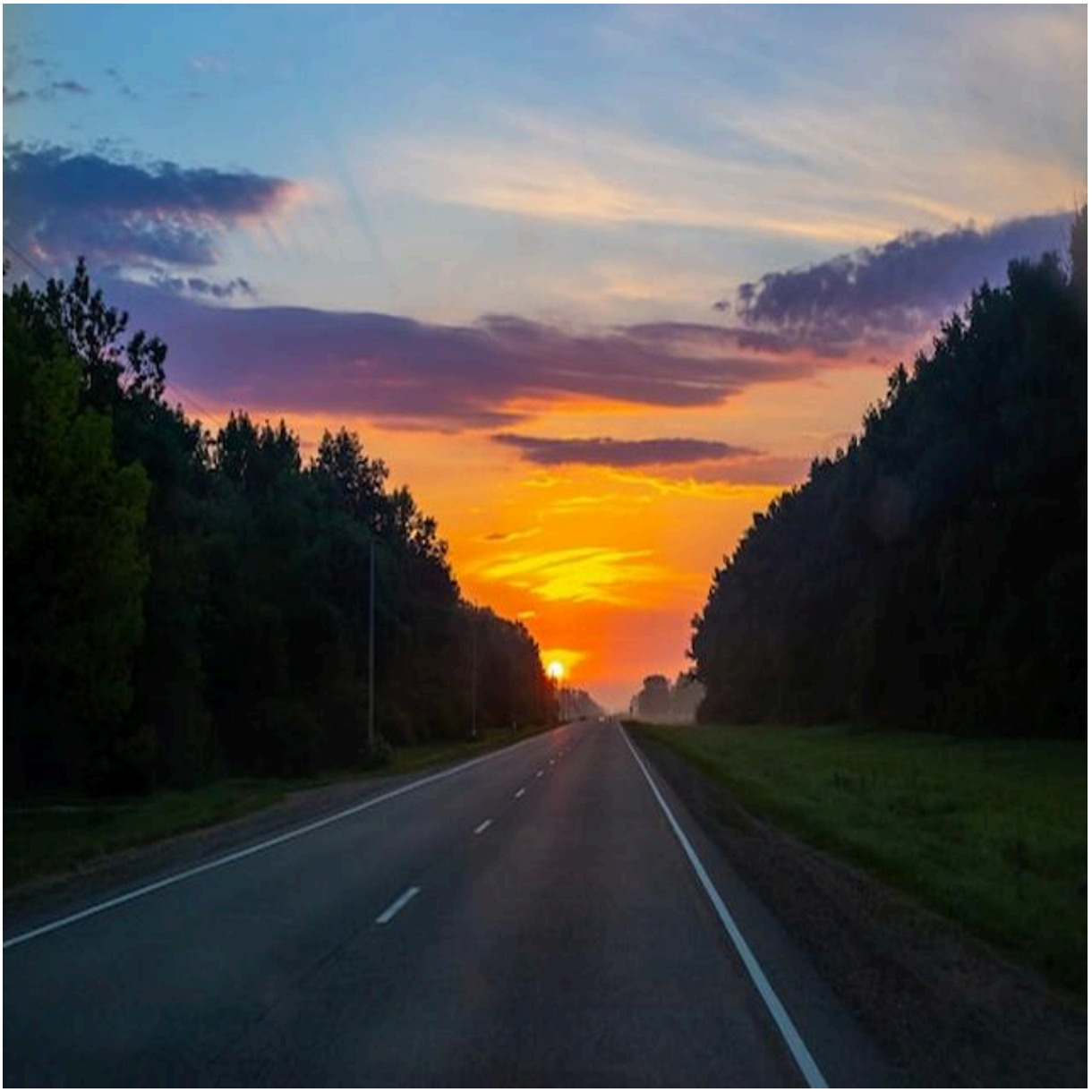
In [2]:

```
# Importar bibliotecas necessárias
from google.colab import files
import cv2
from google.colab.patches import cv2_imshow
import numpy as np


# Solicitar ao usuário que faça o upload de um arquivo de imagem
print("Por favor, faça o upload de uma imagem:")
uploaded = files.upload()

# Verificar se pelo menos um arquivo foi enviado
if len(uploaded.keys()) > 0:
    # Obter o nome do arquivo enviado
    file_name = list(uploaded.keys())[0]

    # Carregar a imagem usando o OpenCV
    image = cv2.imdecode(
        np.frombuffer(uploaded[file_name], np.uint8),
        cv2.IMREAD_COLOR
    )
```

```
    # Verificar se a imagem foi carregada corretamente
    if image is not None:
        print("Imagem carregada com sucesso!")
        # Exibir a imagem carregada
        cv2_imshow(image)
    else:
        print("Não foi possível carregar a imagem.")
else:
    print("Nenhum arquivo foi enviado.")
```

Por favor, faça o upload de uma imagem:

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving style_image_1_resized.jpg to style_image_1_resized.jpg
Imagem carregada com sucesso!

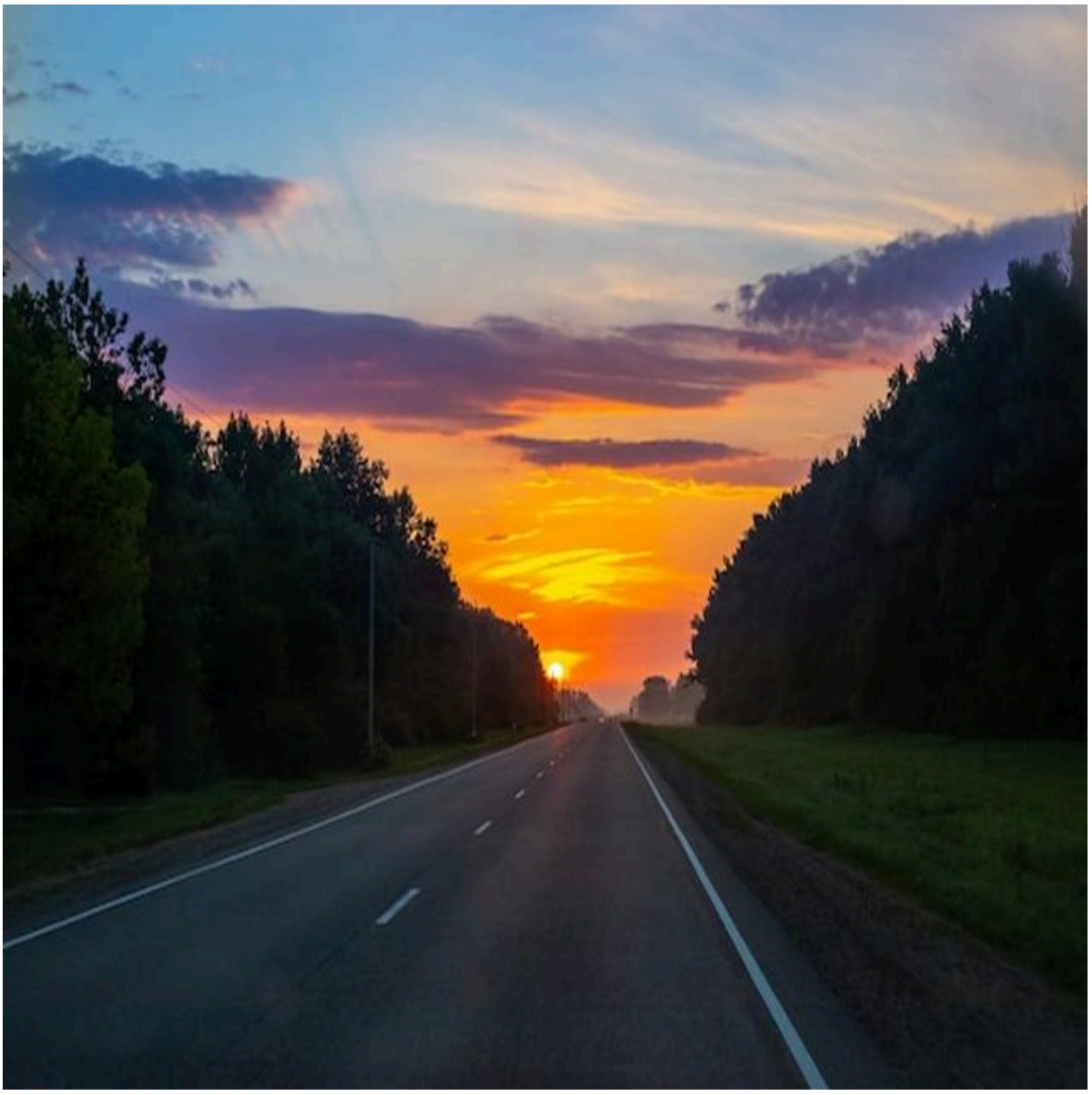


## Ler, escrever e exibir imagens

In [5]:

```
import cv2

# Ler imagem
img = image

# Exibir imagem
cv2_imshow(img)
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()

# Salvar imagem
cv2.imwrite('output.jpg', img)
```



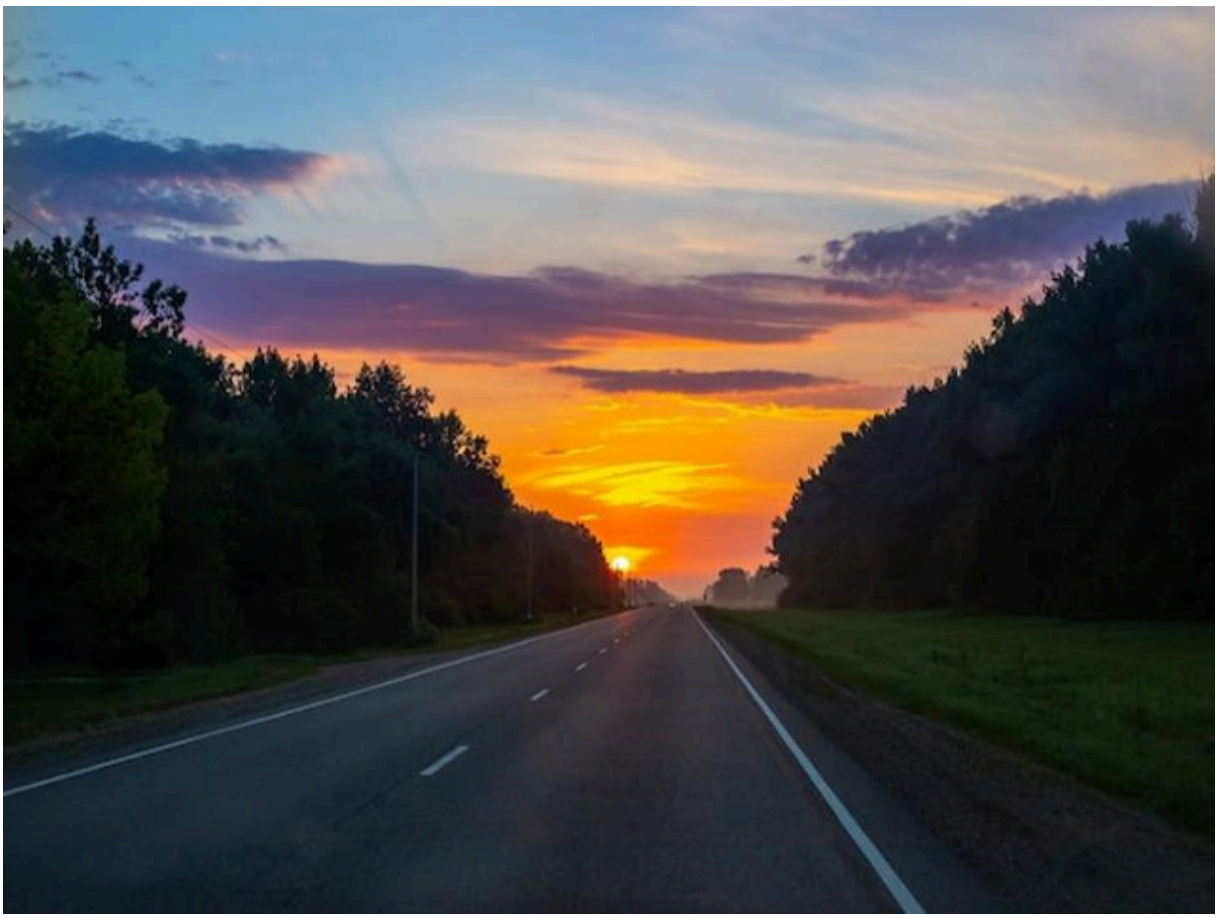
Out[5]:

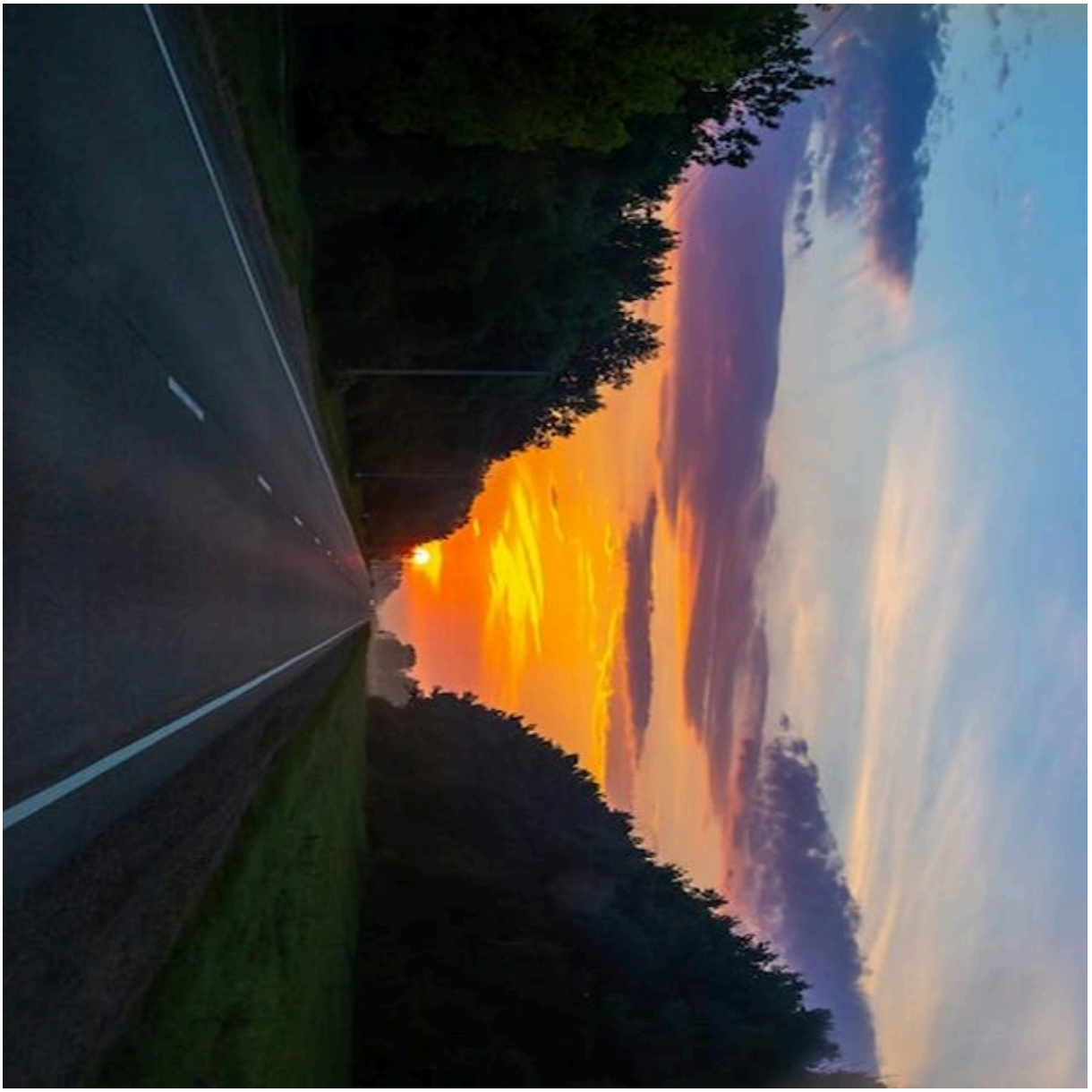
True

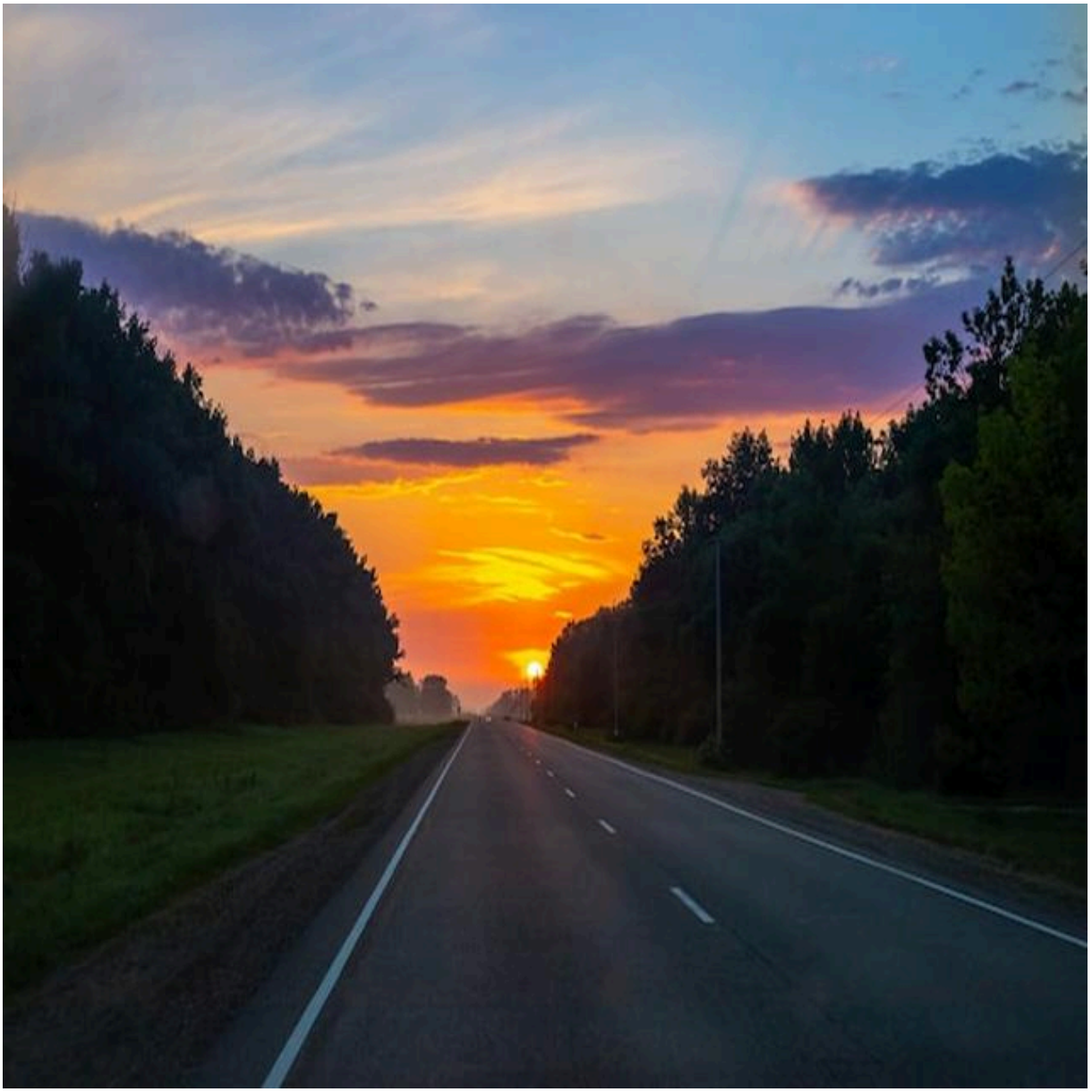
## Transformações de imagem (redimensionar, girar, inverter)

In [6]:

```
import cv2

img = image

# Redimensionar imagem
resized_img = cv2.resize(img, (640, 480))
cv2_imshow(resized_img)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Rotacionar imagem
rotated_img = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
cv2_imshow(rotated_img)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Flip imagem
flipped_img = cv2.flip(img, 1)
cv2_imshow(flipped_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```





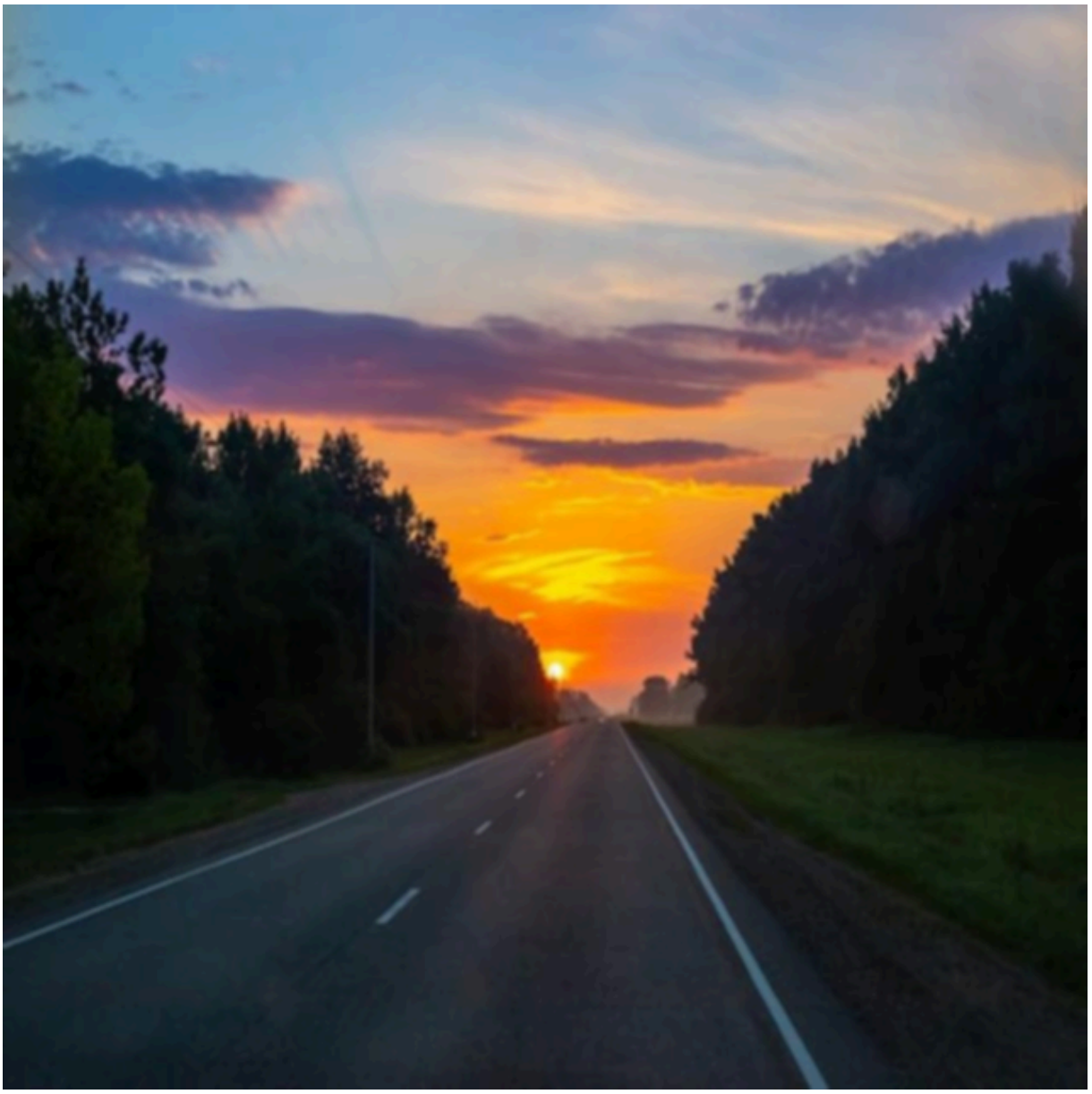


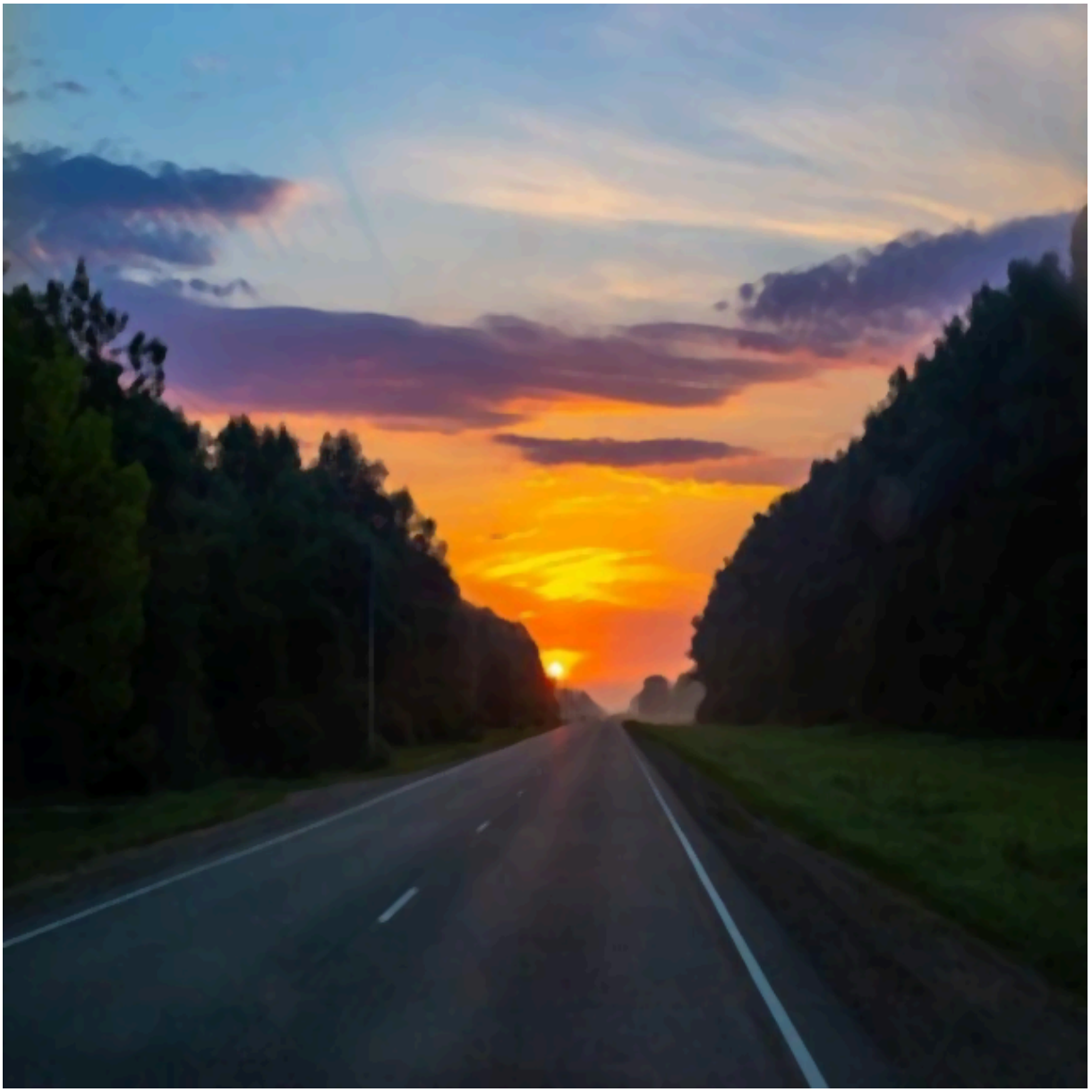
## Filtros de imagem (desfoque gaussiano, filtro mediano, filtro bilateral)

In [8]:

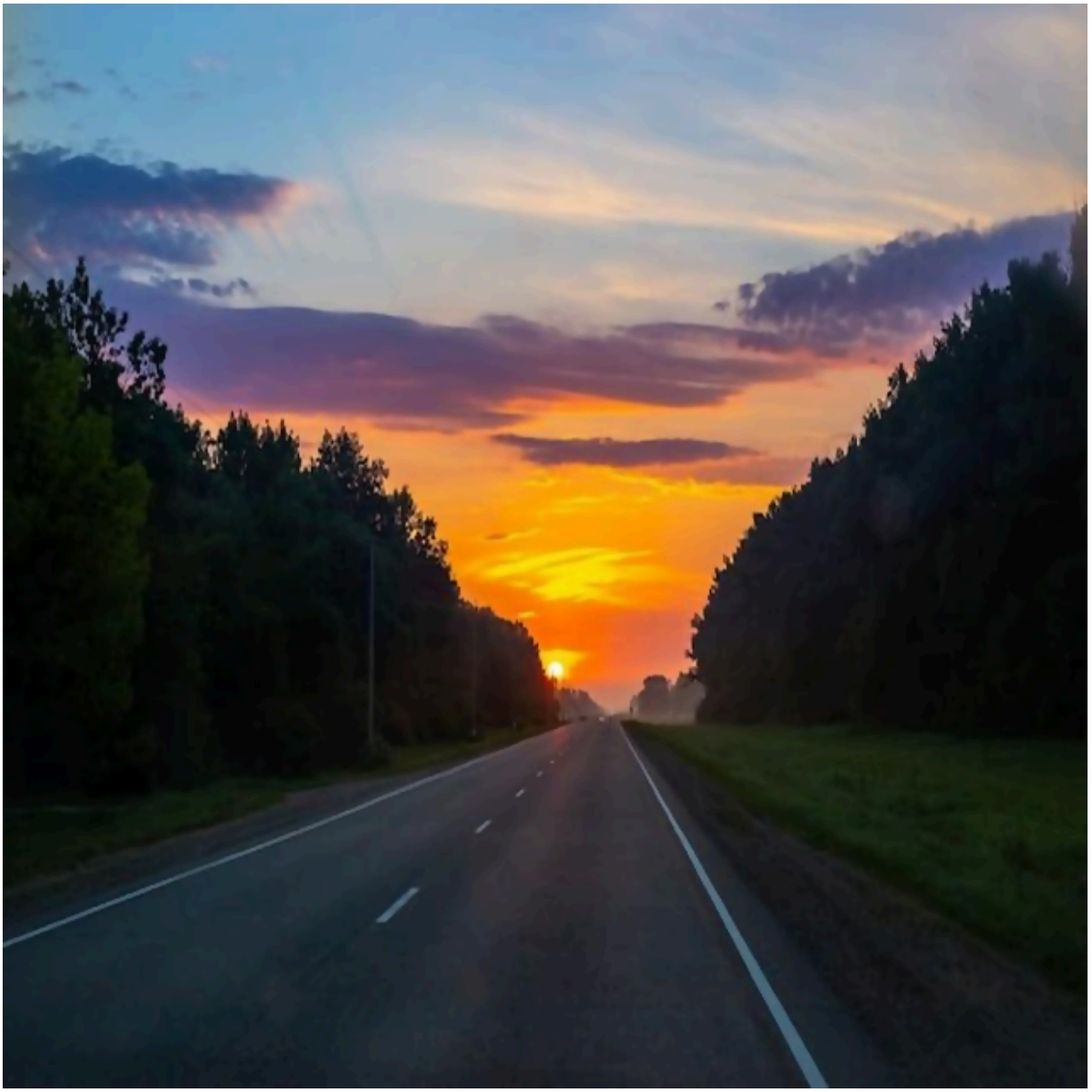
```
import cv2

img = image

# Filtro Gaussiano
gaussian_blur_img = cv2.GaussianBlur(img, (5, 5), 0)
cv2_imshow(gaussian_blur_img)
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()

# Filtro Mediano
median_filter_img = cv2.medianBlur(img, 5)
cv2_imshow(median_filter_img)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Filtro Bilateral
bilateral_filter_img = cv2.bilateralFilter(img, 5, 50, 50)
cv2_imshow(bilateral_filter_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```





Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182



## Image thresholding

In [12]:

```
import cv2

# Converter a imagem para tons de cinza
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Thresholding
_, thresh_img = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)

# Exibir a imagem thresholded
```

```
cv2_imshow(thresh_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## Corner detection (Harris, FAST)

In [18]:

```
# Carregar a imagem
img = image

# Converter a imagem para escala de cinza
```

```
gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Verificar se a conversão foi bem sucedida
if gray_image is not None:
    # Detector de cantos Harris
    dst = cv2.cornerHarris(gray_image, 2, 3, 0.04)
    dst = cv2.dilate(dst, None)
    img_with_harris_corners = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img_with_harris_corners[dst > 0.01 * dst.max()] = [255, 0, 0]
    cv2_imshow(img_with_harris_corners)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    # Detector de cantos FAST
    fast = cv2.FastFeatureDetector_create()
    fast_corners = fast.detect(gray_image)
    img_with_fast_corners = cv2.drawKeypoints(gray_image, fast_corners, None)
    cv2_imshow(img_with_fast_corners)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
else:
    print("Não foi possível converter a imagem para tons de cinza.")
```

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

## Scale-invariant feature transform (SIFT)

In [20]:

```
import cv2

img = gray_image

# Detector de características SIFT
sift = cv2.SIFT_create()
kp, des = sift.detectAndCompute(img, None)
cv2_imshow(cv2.drawKeypoints(img, kp, None))
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## Oriented FAST and rotated BRIEF (ORB)

In [23]:

```
img = gray_image

# Detector de características ORB
orb = cv2.ORB_create()
kp, des = orb.detectAndCompute(img, None)
```

```
cv2_imshow(cv2.drawKeypoints(img, kp, None))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## Correspondência de imagens e detecção de objetos

## Brute-forcematcher

In [25]:

```
import cv2
```

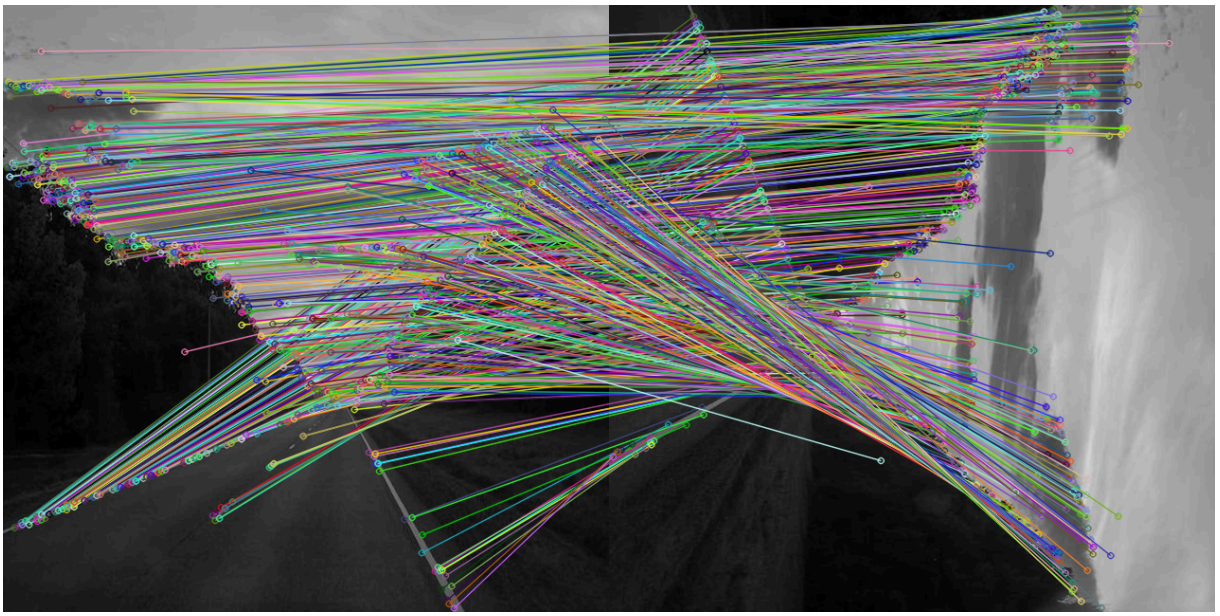
```
img1 = gray_image
img2 = cv2.cvtColor(rotated_img, cv2.COLOR_BGR2GRAY)

# Detector de características SIFT
sift = cv2.SIFT_create()

kp1, des1 = sift.detectAndCompute(img1, None)
kp2, des2 = sift.detectAndCompute(img2, None)

bf = cv2.BFMatcher()
matches = bf.match(des1, des2)
matches = sorted(matches, key=lambda x: x.distance)

result = cv2.drawMatches(img1, kp1, img2, kp2, matches[:10], None, flags=2)
cv2_imshow(result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## FLANN-based matcher

In [32]:

```
# Supondo que você já tenha as imagens img1 e img2
img1 = gray_image
img2 = cv2.cvtColor(rotated_img, cv2.COLOR_BGR2GRAY)

# Detector de características SIFT
sift = cv2.SIFT_create()
```

```
# Encontrar keypoints e descritores para ambas as imagens
kp1, des1 = sift.detectAndCompute(img1, None)
kp2, des2 = sift.detectAndCompute(img2, None)

# Matcher FLANN
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
search_params = dict(checks=50)
flann = cv2.FlannBasedMatcher(index_params, search_params)
matches = flann.knnMatch(des1, des2, k=2)

# Selecionar bons matches
good_matches = []
for m, n in matches:
    if m.distance < 0.7 * n.distance:
        good_matches.append(m)

# Desenhar os matches
result = cv2.drawMatchesKnn(img1, kp1, img2, kp2, [good_matches], None,
flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)
cv2_imshow(result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



## Homography matrix estimation

In [35]:

```
# Detector de características ORB
```

```
orb = cv2.ORB_create()

# Encontrar keypoints e descritores para ambas as imagens
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)

# Matcher de força bruta
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

# Encontrar correspondências
matches = bf.match(des1, des2)

# Classificar as correspondências com base na distância
matches = sorted(matches, key=lambda x: x.distance)

# Desenhar correspondências
img_matches = cv2.drawMatches(img1, kp1, img2, kp2, matches[:10], None,
flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

# Exibir as correspondências
cv2_imshow(img_matches)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## Introdução ao Pillow:

## Primeiro, você precisa importar o Pillow e outras bibliotecas necessárias:

In [37]:

```
from PIL import Image
import matplotlib.pyplot as plt
```

## Em seguida, você pode carregar uma imagem usando o Pillow:

In [38]:

```
# Fazer upload de uma imagem
uploaded = files.upload()
# Obter o nome do arquivo da imagem carregada
image_path = list(uploaded.keys())[0]
# Carregar a imagem usando o Pillow
image = Image.open(image_path)
```

Upload widget is only available when the cell has been executed in the current browser session.
Please rerun this cell to enable.
Saving Coral-Sol-1-1.jpg to Coral-Sol-1-1.jpg

## Agora, vamos explorar algumas funcionalidades básicas do Pillow:

In [41]:

```
# Exibir a imagem
plt.imshow(image)
plt.axis('off')
plt.show()
```

In [42]:

```
# Converter a imagem para escala de cinza
gray_image = image.convert('L')
plt.imshow(gray_image, cmap='gray')
plt.axis('off')
plt.show()
```

In [43]:

```
# Redimensionar a imagem
resized_image = image.resize((300, 300))
plt.imshow(resized_image)
plt.axis('off')
plt.show()
```

In [44]:

```
# Girar a imagem
rotated_image = image.rotate(45)
plt.imshow(rotated_image)
plt.axis('off')
plt.show()
```

In [45]:

```
# Aplicar filtro
from PIL import ImageFilter
filtered_image = image.filter(ImageFilter.BLUR)
plt.imshow(filtered_image)
plt.axis('off')
plt.show()
```

## Introdução ao Scikit-image (SciPy Imaging):

## Primeiro, você precisa importar o scikit-image e outras bibliotecas necessárias:

In [40]:

```
from skimage import io, color, filters, transform
import matplotlib.pyplot as plt
```

## Em seguida, você pode carregar uma imagem usando o scikit-image:

In [46]:

```
# Fazer upload de uma imagem
uploaded = files.upload()
# Obter o nome do arquivo da imagem carregada
image_path = list(uploaded.keys())[0]
image = io.imread(image_path)
```

Upload widget is only available when the cell has been executed in the current browser session.
Please rerun this cell to enable.
Saving create_a_landscape_with_a_house_near_a_lighthous.jpg to
create_a_landscape_with_a_house_near_a_lighthous.jpg

## Agora, vamos explorar algumas funcionalidades básicas do scikit-image:

In [47]:

```
# Exibir a imagem
plt.imshow(image)
plt.axis('off')
plt.show()
```

In [48]:

```
# Converter a imagem para escala de cinza
gray_image = color.rgb2gray(image)
plt.imshow(gray_image, cmap='gray')
plt.axis('off')
plt.show()
```

In [50]:

```
# Redimensionar a imagem
resized_image = transform.resize(image, (300, 300))
plt.imshow(resized_image)
plt.axis('off')
plt.show()
```

In [51]:

```
# Rotação da imagem
rotated_image = transform.rotate(image, 45)
plt.imshow(rotated_image)
plt.axis('off')
plt.show()
```

## Conclusão:

Neste notebook, demos os primeiros passos no mundo do processamento de imagens com o OpenCV, Pillow e scikit-image. Aprendemos a ler, exibir e manipular imagens, realizar transformações básicas como redimensionamento e rotação, aplicar filtros para melhorar a qualidade da imagem, e até mesmo detectar características importantes, como cantos e objetos. Essas são apenas algumas das muitas funcionalidades que essas bibliotecas oferecem para processamento de imagens. Com prática e exploração adicional, você poderá dominar ainda mais esses poderosos conjuntos de ferramentas e aplicá-los em uma ampla variedade de projetos e aplicações. Esperamos que este notebook tenha sido útil e inspirador para suas futuras aventuras em processamento de imagens com o OpenCV, Pillow e scikit-image!

# Notebook Rodrigo Jupyter : Introdução à Visão Computacional

Visão computacional é um campo da inteligência artificial que permite que computadores e sistemas interpretem e façam decisões baseadas em dados visuais. Esta tecnologia é essencial em várias aplicações, como carros autônomos, sistemas de vigilância, diagnósticos médicos e muito mais.

## Ambiente de Configuração

Para configurar o ambiente no Google Colab, precisamos instalar algumas bibliotecas essenciais:

In [ ]:

```
# Importar bibliotecas necessárias
import tensorflow as tf
from tensorflow.keras.datasets import mnist, cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
import matplotlib.pyplot as plt
import numpy as np
```

In [ ]:

```
# Instalar bibliotecas adicionais
!pip install keras
!pip install opencv-python-headless
!pip install yolov5
```

## Modelo de Classificação de Imagens com CNN

## Carregamento do Dataset (MNIST)

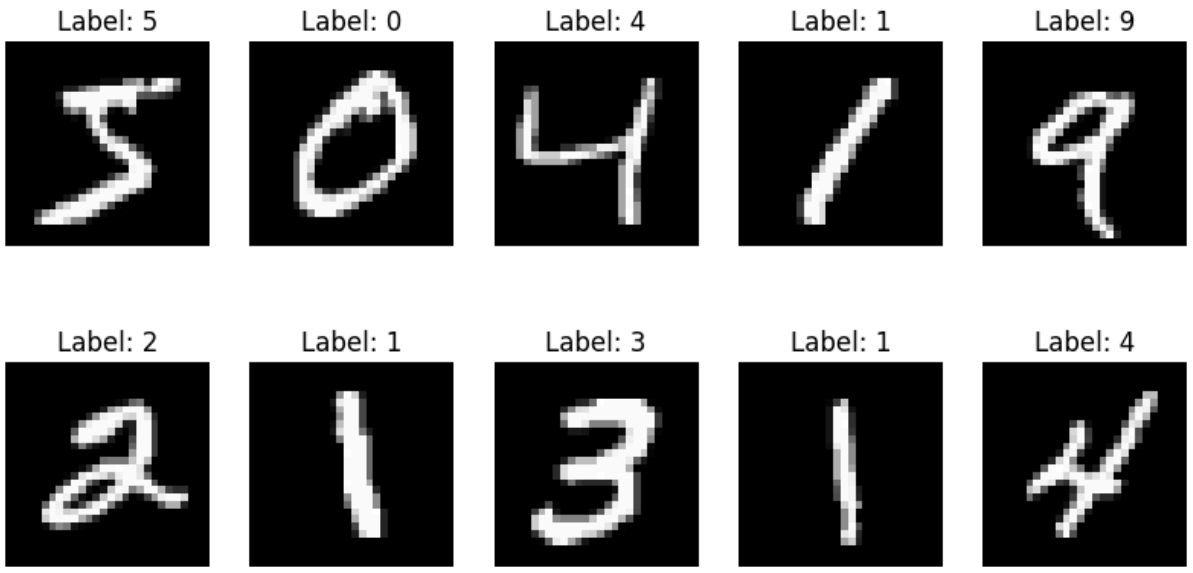
In [ ]:

```
# Carregar dataset MNIST
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.reshape(-1, 28, 28, 1).astype('float32') / 255
x_test = x_test.reshape(-1, 28, 28, 1).astype('float32') / 255

# Visualizar algumas imagens
```

```
plt.figure(figsize=(10, 5))
for i in range(10):
    plt.subplot(2, 5, i+1)
    plt.imshow(x_train[i].reshape(28, 28), cmap='gray')
    plt.title(f"Label: {y_train[i]}")
    plt.axis('off')
plt.show()
```

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz

11490434/11490434 [==============================] - 0s 0us/step




## Construção e Treinamento de um Modelo CNN Simples usando Keras

In [ ]:

```
# Construir modelo CNN
model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28,
1)),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
```

```
])

# Compilar modelo
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

# Treinar modelo
history = model.fit(x_train, y_train, epochs=5, validation_data=(x_test,
y_test))
```

```
Epoch 1/5
1875/1875 [==============================] - 54s 28ms/step - loss: 0.1533 - accuracy: 0.9541
- val_loss: 0.0577 - val_accuracy: 0.9808
Epoch 2/5
1875/1875 [==============================] - 44s 23ms/step - loss: 0.0514 - accuracy: 0.9841
- val_loss: 0.0404 - val_accuracy: 0.9865
Epoch 3/5
1875/1875 [==============================] - 56s 30ms/step - loss: 0.0323 - accuracy: 0.9901
- val_loss: 0.0379 - val_accuracy: 0.9862
Epoch 4/5
1875/1875 [==============================] - 43s 23ms/step - loss: 0.0213 - accuracy: 0.9930
- val_loss: 0.0380 - val_accuracy: 0.9869
Epoch 5/5
1875/1875 [==============================] - 41s 22ms/step - loss: 0.0146 - accuracy: 0.9949
- val_loss: 0.0361 - val_accuracy: 0.9889
```

## Avaliação do Modelo

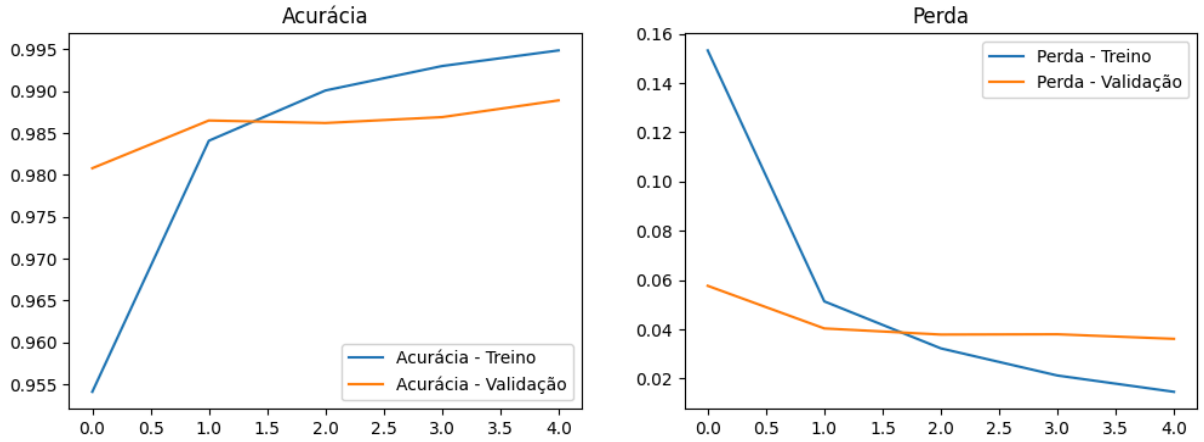
In [ ]:

```
# Avaliar modelo
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Acurácia no conjunto de teste: {accuracy*100:.2f}%")

# Plotar precisão e perda
plt.figure(figsize=(12, 4))
```

```
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Acurácia - Treino')
plt.plot(history.history['val_accuracy'], label='Acurácia - Validação')
plt.legend()
plt.title('Acurácia')
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Perda - Treino')
plt.plot(history.history['val_loss'], label='Perda - Validação')
plt.legend()
plt.title('Perda')
plt.show()
```

313/313 [==============================] - 2s 6ms/step - loss: 0.0361 - accuracy: 0.9889

Acurácia no conjunto de teste: 98.89%




## Modelo de Detecção de Objetos com YOLO

## Explicação sobre YOLO e sua Aplicação

YOLO (You Only Look Once) é uma técnica popular para a detecção de objetos em tempo real. Diferente de outros métodos que aplicam a detecção em várias partes da imagem, o YOLO processa a imagem completa em um único passo, tornando-o extremamente rápido e eficiente.

## Carregamento de um Modelo YOLO Pré-treinado

In [ ]:

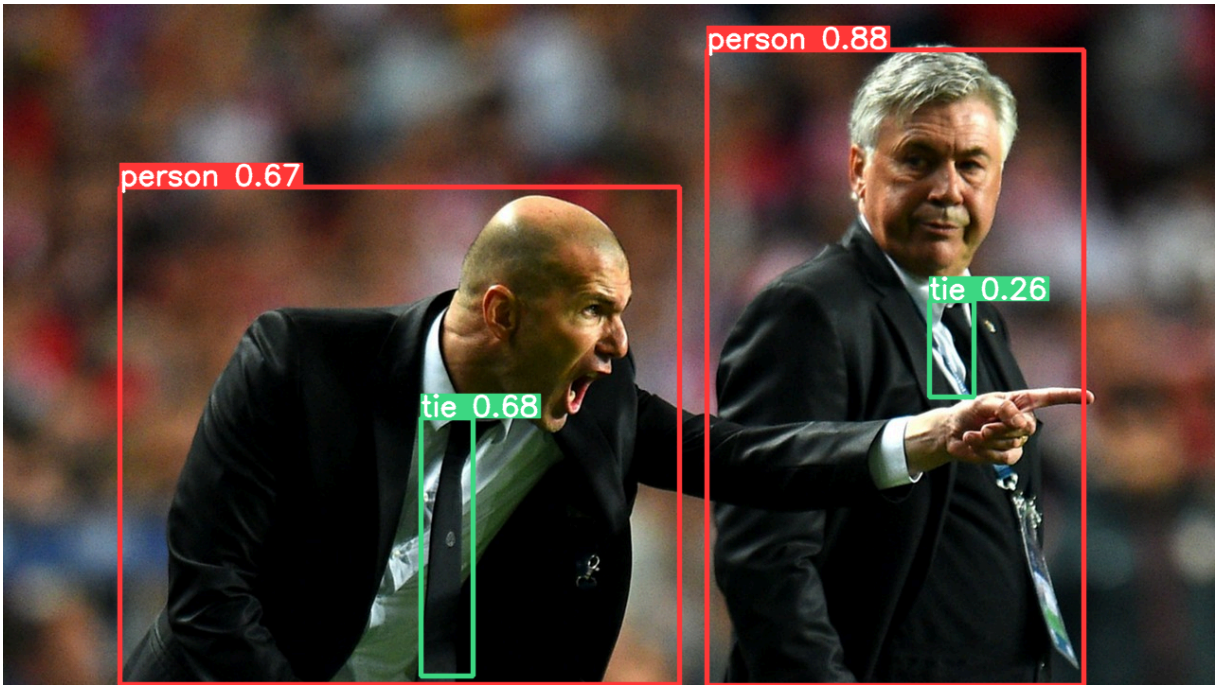
```
# Clonar repositório YOLOv5
!git clone https://github.com/ultralytics/yolov5
%cd yolov5
!pip install -r requirements.txt

# Importar YOLOv5
from yolov5 import YOLOv5

# Carregar modelo YOLOv5 pré-treinado
model = YOLOv5('yolov5s.pt')
```

## Demonstração de Detecção de Objetos em Imagens

In [ ]:

```
# Realizar detecção em uma imagem de exemplo
!wget -O example.jpg https://ultralytics.com/images/zidane.jpg
results = model.predict('example.jpg')

# Visualizar resultados
results.show()
```

```
example.jpg         100%[===================>] 164.99K  --.-KB/s    in 0.01s

2024-05-15 18:48:26 (12.0 MB/s) - ‘example.jpg’ saved [168949/168949]
```




## Modelo de Segmentação de Imagens com U-Net

## Introdução à Segmentação de Imagens e Explicação do Modelo U-Net

A segmentação de imagens é a tarefa de dividir uma imagem em várias partes ou segmentos, muitas vezes para simplificar ou mudar a representação de uma imagem para torná-la mais significativa e fácil de analisar. O modelo U-Net é uma rede neural convolucional projetada especificamente para tarefas de segmentação de imagens.

## Demonstração de Segmentação de Imagens usando um Modelo U-Net Pré-treinado

## Importação:

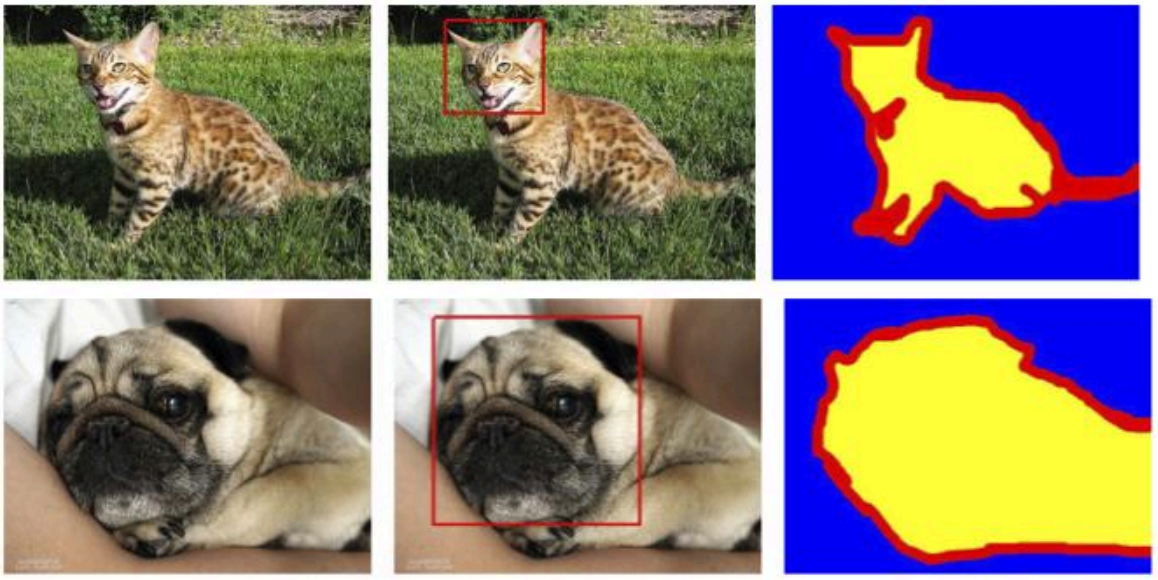
In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
import os
from PIL import Image
import keras
from keras.models import Model
from keras.layers import Conv2D, MaxPooling2D, Input, Conv2DTranspose,
Concatenate, BatchNormalization, UpSampling2D
from keras.layers import Dropout, Activation, LeakyReLU
```

```
from keras.optimizers import Adam, SGD
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
from keras import backend as K
from keras.utils import plot_model
import tensorflow as tf
import glob
import random
import cv2
from random import shuffle
```

## Conjunto de dados

Usaremos o conjunto de dados Oxford-IIIT Pet. Contém 37 classes de cães e gatos com cerca de 200 imagens por cada classe. O conjunto de dados contém rótulos como caixas delimitadoras e máscaras de segmentação. O número total de imagens no conjunto de dados é de pouco mais de 7K.



Baixe as imagens/máscaras e descompacte os arquivos

In [ ]:

```
!wget http://www.robots.ox.ac.uk/~vgg/data/pets/data/images.tar.gz
!wget http://www.robots.ox.ac.uk/~vgg/data/pets/data/annotations.tar.gz
!tar -xvzf images.tar.gz && tar -xvzf annotations.tar.gz
!rm  images/*.mat
```

## Geração

In [ ]:

```
def image_generator(files, batch_size = 32, sz = (256, 256)):
```

```
  while True:

    #extract a random batch
    batch = np.random.choice(files, size = batch_size)

    #variables for collecting batches of inputs and outputs
    batch_x = []
    batch_y = []


    for f in batch:

        #get the masks. Note that masks are png files
        mask = Image.open(f'annotations/trimaps/{f[:-4]}.png')
        mask = np.array(mask.resize(sz))


        #preprocess the mask
        mask[mask >= 2] = 0
        mask[mask != 0 ] = 1

        batch_y.append(mask)

        #preprocess the raw images
        raw = Image.open(f'images/{f}')
        raw = raw.resize(sz)
        raw = np.array(raw)

        #check the number of channels because some of the images are RGBA
or GRAY
        if len(raw.shape) == 2:
          raw = np.stack((raw,)*3, axis=-1)

        else:
          raw = raw[:,:,0:3]

        batch_x.append(raw)

    #preprocess a batch of images and masks
    batch_x = np.array(batch_x)/255.
    batch_y = np.array(batch_y)
    batch_y = np.expand_dims(batch_y,3)

    yield (batch_x, batch_y)
```

In [ ]:

```
batch_size = 32

all_files = os.listdir('images')
shuffle(all_files)

split = int(0.95 * len(all_files))

train_files = all_files[0:split]
test_files  = all_files[split:]

train_generator = image_generator(train_files, batch_size = batch_size)
test_generator  = image_generator(test_files, batch_size = batch_size)
```

In [ ]:

```
x, y= next(train_generator)
```

In [ ]:

```
plt.axis('off')
img = x[0]
msk = y[0].squeeze()
msk = np.stack((msk,)*3, axis=-1)

plt.imshow( np.concatenate([img, msk, img*msk], axis = 1))
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x7eda48b3a9b0>
```

## Métrica de IoU

A métrica de intersecção sobre união (IoU) é uma métrica simples usada para avaliar o desempenho de um algoritmo de segmentação. Dadas duas máscaras

$ytrue, ypred$

avaliamos

$$IoU = \frac{ytrue \cap ypred}{ytrue \cup ypred}$$

In [ ]:

```
def mean_iou(y_true, y_pred):
    yt0 = y_true[:, :, :, 0]
    yp0 = tf.cast(y_pred[:, :, :, 0] > 0.5, 'float32')
    inter = tf.math.count_nonzero(tf.logical_and(tf.equal(yt0, 1),
tf.equal(yp0, 1)))
    union = tf.math.count_nonzero(tf.add(yt0, yp0))
    iou = tf.where(tf.equal(union, 0), 1.0, tf.cast(inter / union,
'float32'))
    return tf.reduce_mean(iou)
```

## Model

In [ ]:

```
def unet(sz = (256, 256, 3)):
  x = Input(sz)
  inputs = x

  f = 8
  layers = []

  for i in range(0, 6):
    x = Conv2D(f, 3, activation='relu', padding='same') (x)
    x = Conv2D(f, 3, activation='relu', padding='same') (x)
    layers.append(x)
    x = MaxPooling2D() (x)
    f = f*2
  ff2 = 64


  j = len(layers) - 1
  x = Conv2D(f, 3, activation='relu', padding='same') (x)
  x = Conv2D(f, 3, activation='relu', padding='same') (x)
  x = Conv2DTranspose(ff2, 2, strides=(2, 2), padding='same') (x)
  x = Concatenate(axis=3)([x, layers[j]])
  j = j -1


  for i in range(0, 5):
    ff2 = ff2//2
```

```
    f = f // 2
    x = Conv2D(f, 3, activation='relu', padding='same') (x)
    x = Conv2D(f, 3, activation='relu', padding='same') (x)
    x = Conv2DTranspose(ff2, 2, strides=(2, 2), padding='same') (x)
    x = Concatenate(axis=3)([x, layers[j]])
    j = j -1



  x = Conv2D(f, 3, activation='relu', padding='same') (x)
  x = Conv2D(f, 3, activation='relu', padding='same') (x)
  outputs = Conv2D(1, 1, activation='sigmoid') (x)


  model = Model(inputs=[inputs], outputs=[outputs])
  model.compile(optimizer = 'rmsprop', loss = 'binary_crossentropy', metrics
= [mean_iou])

  return model
```

In [ ]:

```
model = unet()
```

## Retornos de chamada

Funções simples para salvar o modelo em cada época e mostrar algumas previsões

In [ ]:

```
def build_callbacks():
        checkpointer = ModelCheckpoint(filepath='unet.h5', verbose=0,
save_best_only=True, save_weights_only=True)
        callbacks = [checkpointer, PlotLearning()]
        return callbacks

class PlotLearning(keras.callbacks.Callback):

    def on_train_begin(self, logs={}):
        self.i = 0
        self.x = []
        self.losses = []
        self.val_losses = []
        self.acc = []
        self.val_acc = []

        self.logs = []
```

```
    def on_epoch_end(self, epoch, logs={}):
        self.logs.append(logs)
        self.x.append(self.i)
        self.losses.append(logs.get('loss'))
        self.val_losses.append(logs.get('val_loss'))
        self.acc.append(logs.get('mean_iou'))
        self.val_acc.append(logs.get('val_mean_iou'))
        self.i += 1

print('i=',self.i,'loss=',logs.get('loss'),'val_loss=',logs.get('val_loss'),'
mean_iou=',logs.get('mean_iou'),'val_mean_iou=',logs.get('val_mean_iou'))


        path = np.random.choice(test_files)
        raw = Image.open(f'images/{path}')
        raw = np.array(raw.resize((256, 256)))/255.
        raw = raw[:,:,0:3]


        pred = model.predict(np.expand_dims(raw, 0))


        msk  = pred.squeeze()
        msk = np.stack((msk,)*3, axis=-1)
        msk[msk >= 0.5] = 1
        msk[msk < 0.5] = 0

        combined = np.concatenate([raw, msk, raw* msk], axis = 1)
        plt.axis('off')
        plt.imshow(combined)
        plt.show()
```

## Treino

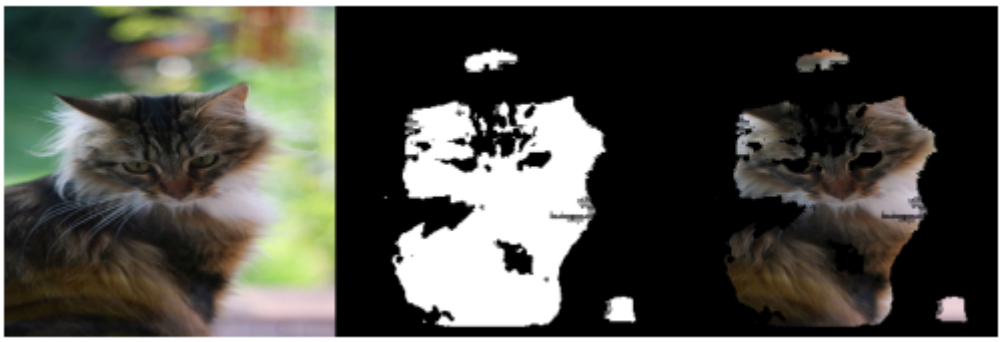
In [ ]:

```
train_steps = len(train_files) // batch_size
test_steps = len(test_files) // batch_size

model.fit(
    train_generator,
    epochs=30,
    steps_per_epoch=train_steps,
    validation_data=test_generator,
    validation_steps=test_steps,
    callbacks=build_callbacks(),
```

```
    verbose=1
)
```

```
Epoch 1/30
219/219 [==============================] - ETA: 0s - loss: 0.5772 - mean_iou:
8.6881e-05i= 1 loss= 0.5771530866622925 val_loss= 0.5287794470787048
mean_iou= 8.68810893734917e-05 val_mean_iou= 0.0
1/1 [==============================] - 1s 1s/step
```

```
219/219 [==============================] - 113s 415ms/step - loss: 0.5772 -
mean_iou: 8.6881e-05 - val_loss: 0.5288 - val_mean_iou: 0.0000e+00
Epoch 2/30
219/219 [==============================] - ETA: 0s - loss: 0.5150 - mean_iou:
0.0435i= 2 loss= 0.5150148868560791 val_loss= 0.46262693405151367 mean_iou=
0.043505702167749405 val_mean_iou= 0.3468187153339386
1/1 [==============================] - 0s 21ms/step
```



```
219/219 [==============================] - 88s 402ms/step - loss: 0.5150 -
mean_iou: 0.0435 - val_loss: 0.4626 - val_mean_iou: 0.3468
```

## Teste

In [ ]:

```
!wget
http://r.ddmcdn.com/s_f/o_1/cx_462/cy_245/cw_1349/ch_1349/w_720/APL/uploads/2015/06/caturday-shutterstock_149320799.jpg -O test.jpg
```

Saving to: ‘test.jpg’

test.jpg 100%[===================>] 61.74K 130KB/s in 0.5s

In [ ]:

```
raw = Image.open('test.jpg')
raw = np.array(raw.resize((256, 256)))/255.
raw = raw[:,:,0:3]


pred = model.predict(np.expand_dims(raw, 0))

msk  = pred.squeeze()
msk = np.stack((msk,)*3, axis=-1)
msk[msk >= 0.5] = 1
msk[msk < 0.5] = 0

combined = np.concatenate([raw, msk, raw* msk], axis = 1)
plt.axis('off')
plt.imshow(combined)
plt.show()
```

1/1 [==============================] - 0s 35ms/step

# Resumo dos artigos e dos tópicos levantados Pattern Recognition

## 1 - Supervised and Un-supervised Classification Algorithms:

## 1.1 - Summary of the Article: Unsupervised Machine Learning Algorithms: A Comprehensive Review:

### Introdução:

O artigo inicia destacando o crescente interesse e a importância do aprendizado de máquina (ML), especificamente no contexto do aprendizado não supervisionado (UL). Este tipo de aprendizado é crucial para avanços em inteligência artificial (AI) porque permite que sistemas descubram padrões ocultos em dados não rotulados. Ao contrário do aprendizado supervisionado, que depende de dados previamente etiquetados, o UL explora a estrutura inerente dos dados para aprender sem exemplos explícitos. Esta capacidade é particularmente valiosa em campos como visão computacional, onde o pré-etiquetamento de grandes conjuntos de dados é impraticável.

### Fundamentos e Relevância do UL:

O artigo destaca sua importância vital no campo da inteligência artificial (AI), particularmente em contextos onde dados rotulados são escassos ou difíceis de obter. O UL é apresentado como uma ferramenta poderosa para explorar grandes volumes de dados não estruturados, permitindo que os sistemas descubram padrões e estruturas sem a necessidade de supervisão direta. Esta capacidade é essencial para avanços em áreas como análise de redes sociais, biologia computacional e detecção de anomalias, onde a definição prévia de categorias pode ser impraticável ou limitadora.

O documento explora como o UL facilita a automação de processos de descoberta de conhecimento, reduzindo a dependência de intervenção humana e permitindo que insights mais profundos sejam extraídos diretamente dos dados. Isso é particularmente relevante em cenários industriais e científicos, onde o volume e a complexidade dos dados excedem a capacidade de análise manual. O aprendizado não supervisionado é descrito como uma abordagem que não apenas aumenta a eficiência dos processos analíticos, mas também potencializa a criação de modelos de machine learning mais robustos e adaptativos.

Além disso, o artigo aborda a relevância estratégica do UL na exploração de dados não explorados, sublinhando seu papel em descobrir correlações ocultas que podem não ser evidentes sem uma análise profunda. Ao identificar essas conexões subjacentes nos dados, o UL permite que organizações e pesquisadores façam descobertas inovadoras que podem levar a avanços significativos em diversas áreas. Este ponto é crucial para entender a capacidade do aprendizado não supervisionado de transformar grandes conjuntos de dados em insights valiosos que podem informar decisões estratégicas e fomentar novas pesquisas.

## Revisão da Literatura e Lacunas:

No artigo, a revisão de literatura existente sobre o aprendizado não supervisionado é meticulosamente avaliada, ressaltando os progressos realizados e os desafios persistentes. A análise revela que, apesar dos avanços significativos em algoritmos não supervisionados, a pesquisa tem sido predominantemente inclinada para abordagens supervisionadas. Isso evidencia uma oportunidade substancial para investigações focadas em aprimorar e expandir a aplicação de técnicas não supervisionadas, especialmente considerando sua utilidade em lidar com conjuntos de dados de grande volume e complexidade variada.

O texto critica a escassez de comparações detalhadas entre diferentes métodos de aprendizado não supervisionado, destacando um vácuo na avaliação de como esses algoritmos se comportam sob variadas condições de dados. Existe uma clara necessidade de estudos mais profundos que elucidem as situações em que cada técnica é mais eficaz, guiando assim a seleção de métodos mais apropriados para cada caso específico de uso. Essa abordagem é vital para otimizar a eficiência dos algoritmos e maximizar sua aplicabilidade prática.

Por fim, o artigo chama atenção para a necessidade urgente de desenvolver benchmarks e padrões de avaliação que se alinhem melhor com os princípios do aprendizado não supervisionado. As métricas atuais, muitas vezes adaptadas de contextos de aprendizado supervisionado, não captam integralmente as nuances e desafios específicos enfrentados por algoritmos não supervisionados. Propõe-se, então, a criação de novos critérios de avaliação que facilitariam uma análise mais precisa da eficácia dessas técnicas, incentivando assim o avanço da pesquisa e o desenvolvimento de novas abordagens adaptadas às complexidades dos dados não rotulados.

## Detalhamento de Algoritmos de Aprendizado Não Supervisionado:

Na análise dos algoritmos de aprendizado não supervisionado, o artigo examina uma série de métodos que têm sido fundamentais na análise de dados onde as etiquetas ou classificações não estão previamente definidas. Os algoritmos como K-Means e DBSCAN são explorados em profundidade, demonstrando como cada um se adapta a diferentes estruturas e volumes de dados. O K-Means é apreciado por sua eficiência em dividir grandes conjuntos de dados em clusters baseados na proximidade aos centros definidos aleatoriamente, o que facilita a identificação de padrões em conjuntos de dados densos e bem distribuídos. Em contraste, o DBSCAN é destacado por sua habilidade de formar clusters com base na densidade dos dados, permitindo identificar agrupamentos em dados que apresentam variações significativas em termos de agrupamento e densidade, além de ser capaz de manejar outliers de forma eficiente.

Outro método crucial discutido é a Análise de Componentes Principais (PCA), que é utilizada para reduzir a dimensionalidade dos dados enquanto mantém as características mais informativas. Este método é particularmente útil em contextos onde a interpretação visual dos dados se faz necessária, permitindo uma melhor visualização e compreensão das principais variâncias presentes em um conjunto de dados complexos. A PCA facilita não apenas a redução de ruído, mas também a identificação de padrões subjacentes que podem ser obscuros em uma análise de alta dimensão.

Além disso, a revisão aborda técnicas avançadas como o algoritmo FP-Growth, que é eficaz para identificar conjuntos frequentes sem a necessidade de gerar candidatos, o que reduz significativamente a quantidade de recursos computacionais necessários. Esse método é exemplificado principalmente em análises de mercado, onde a identificação rápida de padrões de compra frequentes pode oferecer insights valiosos para estratégias de negócios e marketing.

Cada algoritmo traz suas próprias vantagens e limitações, e o artigo faz um trabalho notável ao discutir como a escolha entre eles deve ser guiada por uma compreensão clara do problema específico em mãos, a natureza dos dados disponíveis e os objetivos da análise. Essa discussão não só ilumina as capacidades e aplicabilidades de cada técnica, mas também orienta os usuários na seleção do método mais adequado para extrair o máximo valor dos seus dados, garantindo assim que o aprendizado não supervisionado continue a ser uma ferramenta valiosa e versátil na ciência de dados moderna.

## Análise Comparativa e Avaliação de Desempenho:

Na seção dedicada à análise comparativa e avaliação de desempenho, o artigo fornece uma discussão detalhada sobre a eficácia de diversos algoritmos de aprendizado não

supervisionado, focando em aspectos como eficiência computacional, precisão na formação de clusters, robustez frente a outliers e adaptabilidade a diversos tipos de dados.

A eficiência computacional é abordada inicialmente, com destaque para algoritmos como K-Means e DBSCAN, que são notáveis por sua rapidez e capacidade de manejar grandes conjuntos de dados com um consumo relativamente baixo de recursos. Por exemplo, K-Means é elogiado por suas rápidas iterações que convergem para centróides de clusters de forma eficiente, enquanto o DBSCAN é valorizado por sua habilidade em processar dados sem a necessidade prévia de especificar o número de clusters, adaptando-se automaticamente às características do conjunto de dados.

A precisão na formação de clusters também é uma métrica crucial, com algoritmos como Hierarchical Clustering sendo preferidos por sua capacidade de criar representações visuais claras dos relacionamentos entre os dados através de dendrogramas. Essa capacidade é contrastada com a eficácia do PCA, que, embora seja um método de redução de dimensionalidade, contribui indiretamente para a formação de clusters mais claros e significativos ao preservar as variações mais importantes nos dados.

A análise também aborda a robustez dos algoritmos em relação a outliers e variações na escala dos dados. O DBSCAN, por exemplo, é destacado por sua proficiência em identificar e manejar outliers, diferenciando-se de algoritmos como o K-Means, que pode falhar em conjuntos de dados com muitas anomalias. Além disso, a capacidade do FP-Growth de manter um desempenho consistente mesmo com o aumento do volume de dados é discutida, demonstrando sua adaptabilidade a conjuntos de dados de diferentes tamanhos e complexidades.

A flexibilidade dos algoritmos para adaptar-se a variados tipos e formatos de dados é também um ponto crítico da discussão. Algoritmos baseados em densidade, como o DBSCAN, são apontados por sua habilidade de ajustar-se automaticamente às variações de densidade nos dados, o que os torna mais flexíveis e eficazes em cenários onde a distribuição dos dados é irregular ou desconhecida.

Finalmente, a comparação utiliza métricas quantitativas de desempenho como pureza do cluster, medida de silhouette e coeficiente de Rand ajustado para quantificar a qualidade dos clusters formados por cada algoritmo. Essas métricas são essenciais para avaliar objetivamente o desempenho e ajudam a ilustrar os trade-offs envolvidos na escolha do método mais adequado para cada aplicação específica.

## Conclusões e Futuras Direções de Pesquisa:

O artigo conclui que o aprendizado não supervisionado tem um potencial significativo para avançar a forma como os modelos de dados são construídos e utilizados, especialmente em um mundo onde os dados não rotulados são abundantemente disponíveis. Os autores também sugerem que futuras pesquisas deveriam focar na melhoria da eficiência dos algoritmos existentes e na exploração de novos métodos que possam lidar melhor com as complexidades dos dados modernos.

## 1.2 - A Simplified Un-Supervised Learning Based Approach for Ink Mismatch Detection in Handwritten Hyper-Spectral Document Images

### Introdução:

O artigo explora o uso avançado da imagem hiperespectral (HSI) no campo da análise de documentos, tanto impressos quanto manuscritos, destacando a superioridade dessa tecnologia sobre os sensores RGB tradicionais. Os sensores HSI são capazes de capturar informações em centenas de bandas espectrais, o que permite uma análise detalhada e precisa das propriedades dos materiais que compõem os documentos. Essa tecnologia é particularmente valiosa para diferenciar elementos que são visualmente semelhantes, abrindo novas possibilidades para a detecção de fraudes e a verificação de autenticidade de documentos importantes. Este estudo foca em desenvolver um método eficiente, baseado em aprendizado não supervisionado, para estimar o número de tintas diferentes presentes em uma imagem de documento hiperespectral, sem a necessidade de informações prévias sobre os dados.

### Metodologia:

A metodologia descrita no artigo aborda um procedimento detalhado para detectar a incompatibilidade de tintas em imagens hiperespectrais de documentos manuscritos usando uma abordagem de aprendizado não supervisionado. Inicialmente, o estudo enfoca o pré-processamento de imagens hiperespectrais, um passo crítico que prepara os dados para a análise subsequente. Este pré-processamento inclui a calibração espectral dos dados, a correção de iluminação, e o recorte das regiões de interesse. Essas etapas são fundamentais para garantir a precisão dos dados que serão analisados, removendo artefatos que poderiam distorcer os resultados da classificação de tintas. Os autores escolheram Python como linguagem de programação devido à sua vasta biblioteca de processamento de imagens e capacidade de manipulação de grandes conjuntos de dados, usando bibliotecas específicas como Spectral para lidar com dados hiperespectrais, Numpy para manipulação de matrizes, OpenCV para processamento de imagens, e Matplotlib para a visualização dos resultados.

Para a classificação das tintas, o estudo utiliza o algoritmo K-means, uma escolha motivada pela sua eficiência em formar clusters precisos de dados não rotulados e sua simplicidade operacional. O K-means é aplicado para segmentar as assinaturas espectrais dos pixels de tinta, agrupando-as em clusters que representam diferentes tipos de tinta presentes no documento. A escolha deste algoritmo é justificada pela sua capacidade de lidar com a complexidade e a alta dimensionalidade dos dados hiperespectrais, oferecendo uma solução escalável e relativamente rápida para o problema de detecção de incompatibilidade de tintas. Além disso, a implementação do K-means na biblioteca Spectral do Python facilita a integração e execução do processo de clusterização, permitindo aos pesquisadores concentrarem-se na análise dos resultados ao invés de detalhes de codificação complexos.

Finalmente, o artigo detalha o procedimento de validação da metodologia proposta, onde os clusters identificados pelo K-means são analisados para verificar a consistência e precisão na classificação das tintas. Este processo de validação é essencial para estabelecer a confiabilidade do método em aplicações práticas, como a autenticação de documentos e a detecção de fraudes. A análise dos clusters é realizada através da comparação das assinaturas espectrais derivadas com referências conhecidas, permitindo aos pesquisadores ajustar parâmetros e refinar o algoritmo para melhorar a precisão. A combinação dessas etapas metodológicas fornece uma base sólida para a aplicação eficaz do aprendizado não supervisionado na detecção de incompatibilidade de tintas em documentos importantes.

## Resultados Experimentais e Análise:

O artigo apresenta uma avaliação detalhada dos resultados experimentais obtidos com a aplicação da metodologia proposta, utilizando o dataset iVision HHID que inclui uma variedade de imagens hiperespectrais de documentos manuscritos. Este dataset é particularmente adequado para testar a eficácia do algoritmo K-means na identificação de diferentes tipos de tintas, visto que contém amostras com variadas composições de tinta. Durante os testes, cada imagem foi processada para extrair as assinaturas espectrais das tintas utilizadas, que foram posteriormente analisadas pelo algoritmo K-means para agrupar as tintas similares. Os resultados mostram que o algoritmo foi capaz de distinguir efetivamente entre as diferentes tintas, demonstrando a robustez da técnica em um contexto real de análise de documentos.

A análise dos resultados é complementada com uma série de visualizações gráficas que ilustram a distribuição dos clusters de tinta e a separação entre eles. Essas visualizações são fundamentais para entender como o algoritmo interpreta as variações nas assinaturas espectrais e como essas variações correspondem a diferentes tipos de tintas no documento. Além disso, foram realizadas comparações quantitativas dos clusters identificados com as informações conhecidas das amostras de tinta, proporcionando uma validação adicional da precisão do método. Essas comparações ajudaram a confirmar que os clusters formados pelo K-means correspondem de forma precisa às diferentes tintas usadas nos textos,

reforçando a aplicabilidade da técnica para tarefas de autenticação e análise forense de documentos.

Por fim, o artigo discute as implicações práticas dos resultados obtidos, enfatizando como a metodologia pode ser usada para detectar falsificações e autenticar documentos em cenários legais e históricos. A capacidade de detectar sutis diferenças de tinta sem a necessidade de etiquetas pré-definidas é particularmente valiosa em situações onde a proveniência dos documentos é incerta ou disputada. Esta análise aprofundada não apenas valida a técnica proposta mas também abre caminho para futuras investigações que podem explorar o uso de algoritmos mais complexos ou conjuntos de dados mais amplos para melhorar ainda mais a detecção e classificação de tintas em documentos hiperespectrais.

## Conclusão

A conclusão do estudo reitera a eficácia da abordagem simplificada baseada em aprendizado não supervisionado para a detecção de incompatibilidade de tintas em imagens hiperespectrais de documentos manuscritos. O algoritmo K-means, empregado neste contexto, demonstrou uma habilidade notável para identificar e classificar diversas tintas com precisão, sem a necessidade de um conjunto de dados pré-rotulado. Isso destaca a potencial aplicabilidade do método em uma variedade de situações práticas, especialmente na análise forense de documentos, onde a detecção rápida e precisa de falsificações pode ser crucial. Além disso, a técnica oferece uma ferramenta valiosa para conservadores de documentos e historiadores que buscam verificar a autenticidade de documentos antigos ou manuscritos de significado cultural ou histórico.

Porém, o estudo também reconhece as limitações da abordagem atual, incluindo a sensibilidade do algoritmo a variações na qualidade da imagem e a necessidade de refinamento nos processos de pré-processamento e seleção de parâmetros do algoritmo. O artigo sugere que pesquisas futuras poderiam explorar a integração de outras técnicas de aprendizado não supervisionado, como C-means, PCA e SVM, para comparar sua eficácia e potencialmente melhorar a robustez e precisão da detecção de tintas. Ao avançar, a adoção de um espectro mais amplo de algoritmos poderia ajudar a superar os desafios atuais e ampliar a aplicabilidade da tecnologia em diferentes contextos de análise de documentos.

## 2 - Clustering Techniques:

### 2.1- A comprehensive survey of clustering algorithms: State-of-the-art techniques and developments

### 1. Introdução

A introdução do artigo esclarece a importância do agrupamento como uma questão fundamental no aprendizado não supervisionado, abordando sua aplicabilidade em diversos campos como ciência da computação, biologia e comunicação. Apesar de não existir uma definição única e universalmente aceita de agrupamento, é geralmente entendido que o objetivo principal é agrupar instâncias de dados de modo que instâncias dentro de um mesmo grupo sejam mais semelhantes entre si do que com instâncias de outros grupos. Este processo envolve a escolha criteriosa de medidas de similaridade ou dissimilaridade que são capazes de quantificar eficazmente o quanto os elementos são parecidos ou diferentes.

O texto também discute os desafios associados ao desenvolvimento de um bom algoritmo de agrupamento, incluindo a seleção de características, design de algoritmo, avaliação de resultados e interpretação dos clusters formados. Essas etapas são críticas para assegurar que o agrupamento produza resultados válidos e úteis para a análise de dados. A seção estabelece o contexto para o restante do documento, que detalha as metodologias e técnicas específicas usadas para enfrentar esses desafios, preparando o terreno para uma discussão aprofundada sobre os diferentes tipos de algoritmos de agrupamento.

### 2. Medidas de Distância e Similaridade

Esta seção explica as bases dos algoritmos de agrupamento, concentrando-se nas métricas de distância e similaridade que determinam como os dados são agrupados. As medidas de distância, como a distância Euclidiana, de Minkowski e de Manhattan, são cruciais para quantificar a dissimilaridade entre pontos de dados em um espaço multidimensional. Por outro lado, as medidas de similaridade, como a similaridade de cosseno e a correlação de

Pearson, são utilizadas para avaliar o quão alinhados ou semelhantes são os pontos de dados em termos de suas direções ou perfis de variabilidade.

Além disso, o artigo discute como a escolha da medida adequada impacta diretamente a eficácia dos algoritmos de agrupamento e pode influenciar a formação dos clusters. Por exemplo, diferentes tipos de dados e estruturas de dados podem requerer a adaptação ou a escolha de uma medida específica que melhor reflita as características dos dados. Este entendimento é fundamental para desenvolver algoritmos que não apenas performar bem tecnicamente, mas que também ofereçam insights significativos e úteis para o usuário final.

## 3. Indicadores de Avaliação

Os indicadores de avaliação são cruciais para validar a eficácia de um algoritmo de agrupamento, garantindo que os grupos formados sejam internamente coerentes e distintos entre si. Esta seção do artigo detalha dois tipos principais de indicadores: internos e externos. Os indicadores internos avaliam a qualidade do agrupamento com base nas próprias informações do conjunto de dados, enquanto os indicadores externos utilizam dados de referência ou padrões conhecidos para avaliar os clusters.

A seção prossegue discutindo como esses indicadores são aplicados na prática e como podem ser utilizados para comparar diferentes algoritmos de agrupamento. Por exemplo, o coeficiente de silhueta, o índice Davies-Bouldin e a validação cruzada são métodos frequentemente usados para medir a qualidade dos clusters sem necessidade de supervisão externa. Esses indicadores ajudam os pesquisadores e analistas a entender melhor as capacidades e limitações dos diferentes métodos de agrupamento, facilitando a escolha do algoritmo mais adequado para um problema específico.

## 4. Algoritmos de Agrupamento Tradicionais

Esta seção revisa os algoritmos de agrupamento tradicionais, que são a base para muitos dos métodos mais avançados usados hoje. Os algoritmos são categorizados em vários tipos, como agrupamento baseado em partição (por exemplo, K-means e K-medoids),

agrupamento hierárquico (por exemplo, agrupamento aglomerativo e divisivo) e agrupamento baseado em densidade (por exemplo, DBSCAN). Cada categoria é explicada detalhadamente, com discussões sobre como cada método opera, suas vantagens específicas e as situações em que são mais eficazes.

Adicionalmente, a seção aborda as limitações dos métodos tradicionais, como sua sensibilidade a outliers, a necessidade de especificar o número de clusters a priori e a dificuldade em lidar com clusters de formas irregulares. Essas limitações motivam o desenvolvimento de novos algoritmos que tentam superar esses desafios, adaptando-se melhor às complexidades dos grandes volumes de dados e à variedade de tipos de dados encontrados nas aplicações modernas.

## 5. Algoritmos de Agrupamento Modernos

Nesta parte, o artigo apresenta os algoritmos de agrupamento modernos, que têm sido desenvolvidos para lidar com desafios específicos, como grandes volumes de dados, alta dimensionalidade e a necessidade de métodos mais flexíveis que não requerem predefinição do número de clusters. Esses algoritmos incluem métodos baseados em teoria espectral, propagação de afinidade e algoritmos baseados em modelos que utilizam técnicas de aprendizado profundo e otimização estocástica para melhorar a precisão e a eficiência do agrupamento.

A discussão é enriquecida com exemplos de como esses algoritmos modernos são aplicados em contextos reais, mostrando sua superioridade em certos cenários, como em conjuntos de dados com estruturas complexas ou quando os dados evoluem ao longo do tempo. Os algoritmos modernos oferecem uma flexibilidade significativamente maior e podem descobrir insights mais profundos nos dados, o que é crucial para aplicações em campos como bioinformática, análise de redes sociais e sistemas de recomendação.

## 6. Conclusão

A conclusão do artigo sintetiza os pontos discutidos, reafirmando a importância dos algoritmos de agrupamento na análise de dados. Os autores destacam como uma compreensão profunda dos diferentes algoritmos e suas aplicações pode ajudar a maximizar o valor extraído dos dados. Além disso, eles apontam para futuras direções de pesquisa no campo do agrupamento, incluindo a integração de métodos de aprendizado não supervisionado com aprendizado supervisionado e reforçado, para criar sistemas de análise de dados ainda mais robustos e adaptáveis.

Finalmente, a seção ressalta a necessidade de contínua inovação nos algoritmos de agrupamento para acompanhar o rápido desenvolvimento em outras áreas da tecnologia e ciência de dados. Os autores encorajam a comunidade acadêmica e profissional a colaborar no desenvolvimento de novas técnicas que possam enfrentar os desafios emergentes no tratamento e análise de dados complexos.

## 2.2 - Deep Clustering: A Comprehensive Survey

## 1. Introdução

A introdução do artigo destaca a crescente relevância do agrupamento profundo em análises de dados modernas, onde padrões complexos e estruturas ocultas nos dados devem ser identificados e explorados. A abordagem convencional de agrupamento muitas vezes não consegue lidar com a alta dimensionalidade e a diversidade dos dados atuais, como imagens, texto e sequências biológicas. Neste contexto, o agrupamento profundo emerge como uma solução promissora, capaz de aprender representações profundas e abstratas dos dados que facilitam a identificação de agrupamentos mais naturais e significativos.

O texto também discute a evolução das técnicas de agrupamento, desde os métodos tradicionais baseados em modelos simples até as abordagens recentes que utilizam redes neurais profundas para extrair características essenciais dos dados. Esse avanço tecnológico permite que os algoritmos de agrupamento superem desafios anteriores, como a sensibilidade a outliers e a dificuldade em escalar para grandes conjuntos de dados. A

introdução estabelece a base para uma discussão detalhada sobre diferentes categorias de agrupamento profundo, cada uma adaptada para enfrentar diferentes tipos de desafios de dados.

## 2. Agrupamento Profundo com Uma Única Visão

Esta seção aborda em profundidade o agrupamento profundo de dados de uma única visão, onde é crucial entender e transformar a representação dos dados para facilitar o agrupamento eficaz. Métodos baseados em redes como autoencoders e redes convolucionais são explorados por sua capacidade de reduzir a dimensionalidade e destacar características fundamentais dos dados que são críticas para o agrupamento. Essas técnicas são capazes de transformar os dados brutos em uma forma que ressalta a similaridade intrínseca entre os pontos de dados dentro de um mesmo cluster.

Adicionalmente, a seção discute diferentes abordagens para otimizar essas redes neurais, com foco em minimizar distâncias dentro de clusters e maximizar distâncias entre clusters diferentes. Estratégias como a utilização de funções de perda especializadas e o ajuste de parâmetros de rede são cruciais para alcançar uma separação clara e eficaz entre os grupos. A habilidade de adaptar essas técnicas às especificidades dos dados tratados é crucial para o sucesso do agrupamento, destacando a flexibilidade e a adaptabilidade do agrupamento profundo.

## 3. Agrupamento Profundo Semi-Supervisionado

O agrupamento semi-supervisionado é discutido como uma forma de combinar pequenas quantidades de dados etiquetados com grandes volumes de dados não etiquetados para melhorar a precisão do agrupamento. Este método aproveita o conhecimento prévio, que pode ser extremamente valioso em contextos onde os dados completos são difíceis de etiquetar. Introduzindo restrições baseadas em etiquetas dentro do processo de treinamento, é possível guiar o modelo de agrupamento para resultados mais precisos e confiáveis.

A seção detalha como a inclusão dessas informações de etiquetas modifica o processo de treinamento dos modelos de agrupamento profundo, permitindo que eles aprendam não apenas das características dos dados, mas também das relações implícitas entre grupos. Isso é especialmente útil em campos como biologia e medicina, onde as etiquetas podem indicar categorizações importantes baseadas em conhecimento especializado. Os desafios de integrar essas informações de maneira eficaz e as estratégias para superar tais desafios são também explorados, ressaltando a complexidade e a necessidade de abordagens inovadoras no agrupamento semi-supervisionado.

## 4. Agrupamento Profundo Multi-Visão

A seção sobre agrupamento multi-visão explora como dados provenientes de diferentes fontes ou sensores podem ser integrados para formar uma visão unificada que facilita o agrupamento. A complexidade dos dados multi-visão, como imagens capturadas de diferentes ângulos ou dados coletados por diferentes tipos de sensores, pode oferecer um desafio significativo devido à heterogeneidade e ao ruído. No entanto, técnicas de agrupamento profundo são capazes de extrair e combinar características relevantes dessas diversas fontes, realçando aspectos complementares e consistentes.

Além disso, a seção discute a aplicação de redes complexas que podem aprender a correlacionar e a sintetizar informações de múltiplas fontes, resultando em uma representação coerente e robusta que pode ser eficazmente agrupada. Isso inclui o uso de redes de autoencoders e redes neurais convolucionais que são especificamente ajustadas para tratar as nuances e a diversidade dos conjuntos de dados multi-visão, demonstrando como a inovação contínua em métodos de agrupamento profundo está expandindo as fronteiras do que é possível em termos de análise de dados complexos.

## 5. Agrupamento Profundo com Transferência de Aprendizado

A transferência de aprendizado é apresentada como uma técnica poderosa para melhorar o agrupamento de dados ao transferir conhecimento de um domínio para outro. Esta abordagem é particularmente útil quando os conjuntos de dados de treinamento e teste diferem significativamente em termos de distribuição ou quando os dados de treinamento são limitados. O texto explora como os modelos de agrupamento podem ser pré-treinados em um domínio com abundância de dados e depois ajustados para funcionar efetivamente em um novo domínio com menos dados etiquetados.

Detalhes sobre como essas técnicas são implementadas na prática, incluindo as adaptações necessárias nos modelos de rede neural profunda, são discutidos para ilustrar os desafios e as soluções técnicas envolvidas. A capacidade de adaptar modelos de uma tarefa para outra revela a flexibilidade do agrupamento profundo e abre novas possibilidades para sua aplicação em áreas onde os dados são escassos ou onde a coleta de dados etiquetados é impraticável.

## 6. Conclusão

A conclusão reitera a importância do agrupamento profundo como uma ferramenta vital na exploração de grandes conjuntos de dados em várias aplicações. Reflecte sobre como as

técnicas discutidas ao longo do artigo não apenas abordam as limitações dos métodos de agrupamento tradicionais, mas também abrem caminho para novas pesquisas e aplicações. A necessidade de explorar ainda mais as capacidades do agrupamento profundo, especialmente em combinação com outras abordagens de aprendizado de máquina, é destacada como um campo promissor para futuras inovações.

## 3 - Dimensionality Reduction Methods in Pattern Recognition:

## 3.1 - A survey of dimensionality reduction techniques

## 1. Introdução

A introdução do artigo destaca a relevância crescente da redução de dimensionalidade em campos diversificados como aprendizado de máquina, análise de dados e visão computacional. A necessidade de simplificar conjuntos de dados de alta dimensionalidade sem perder informações críticas é discutida, sublinhando a importância dessas técnicas para melhorar a eficiência computacional e a interpretabilidade dos resultados. Os autores ressaltam que a redução de dimensionalidade não só facilita a visualização de dados complexos mas também pode melhorar significativamente o desempenho de algoritmos de aprendizado.

Além disso, a introdução prepara o terreno para uma exploração detalhada das várias técnicas de redução de dimensionalidade, tanto lineares quanto não lineares. Ela enfatiza como essas técnicas são essenciais para lidar com os desafios impostos por grandes volumes de dados e alta dimensionalidade, frequentemente encontrados em aplicações modernas de ciência de dados.

## 2. Seleção e Extração de Características

A seção sobre seleção e extração de características detalha métodos para identificar as características mais relevantes dentro de um conjunto de dados. A distinção entre seleção de características, que envolve escolher subconjuntos de características existentes, e extração de características, que cria novas características a partir das originais, é discutida extensivamente. Este processo é crucial para reduzir a complexidade dos dados antes da aplicação de técnicas de redução de dimensionalidade.

Esta parte também explora diferentes abordagens e algoritmos utilizados para extração de características, como análise de componentes principais (PCA) e análise de fatores. A escolha da técnica adequada pode variar dependendo da natureza dos dados e dos

objetivos específicos da análise, algo que os autores abordam através de exemplos práticos que ilustram a aplicabilidade de cada método em diferentes cenários.

## 3. Redução de Dimensionalidade Linear

A discussão sobre redução de dimensionalidade linear abrange técnicas fundamentais como PCA, que é utilizada para diminuir a complexidade dos dados enquanto retém a maior parte da variância. A técnica de Análise Discriminante Linear (LDA) também é explorada, destacando seu uso na maximização da separabilidade entre classes conhecidas em um conjunto de dados.

Os autores explicam os princípios matemáticos subjacentes a essas técnicas e como elas podem ser aplicadas para transformar dados de alta dimensionalidade em um formato mais gerenciável sem perder informações essenciais. Exemplos de aplicação prática dessas técnicas são fornecidos para ilustrar como elas são implementadas na prática e os tipos de problemas que podem resolver.

## 4. Redução de Dimensionalidade Não Linear

Técnicas de redução de dimensionalidade não linear, como o mapeamento t-SNE e o embedding isométrico, são discutidas nesta seção. Estas técnicas são particularmente úteis para dados que contêm estruturas complexas que não podem ser capturadas adequadamente por métodos lineares.

Os autores detalham como essas técnicas funcionam para preservar as relações locais entre pontos em um conjunto de dados, enquanto ainda conseguem representar a estrutura global de forma eficaz. Eles também discutem as limitações e os desafios associados à aplicação dessas técnicas em grandes conjuntos de dados, incluindo questões de escalabilidade e interpretabilidade.

## 5. Conclusão

Na conclusão, os autores recapitulam os principais pontos discutidos ao longo do documento, enfatizando a importância crítica da redução de dimensionalidade em uma variedade de aplicações científicas e práticas. Eles também apontam para futuras direções de pesquisa na área, como o desenvolvimento de técnicas mais eficientes e adaptáveis que possam lidar com as crescentes demandas de dados de grandes dimensões e complexidade variável.

## 3.2 - Online inverse reinforcement learning with limited data

### 1. Introdução

A introdução do artigo enfatiza a importância do aprendizado por reforço inverso online (IRL), especialmente em contextos onde os dados são limitados e as dinâmicas do sistema são incertas. Os autores destacam como o IRL pode ser aplicado para entender e replicar o comportamento de um agente observando apenas suas trajetórias de estado e controle, sem a necessidade de intervenção direta ou de conhecimento completo do modelo do sistema. Esta abordagem é apresentada como uma solução para extrair a função de recompensa que guia o comportamento do agente, mesmo em cenários onde os dados disponíveis são escassos.

Além disso, a introdução define o cenário de aplicação do IRL em tempo real, onde os dados são coletados e processados simultaneamente à execução das tarefas pelo agente. Isso é contrastado com abordagens de IRL offline, que dependem de um conjunto de dados completos coletados previamente. Os desafios de trabalhar com dados limitados são discutidos, juntamente com a apresentação de uma nova abordagem recursiva baseada em modelos para o IRL, que permite o uso de pares de estados e ações fora da trajetória usual do agente.

### 2. Notação e Formulação do Problema

Esta seção detalha a notação matemática usada ao longo do artigo e descreve formalmente o problema de IRL online. Os autores introduzem a notação para espaços dimensionais, vetores, matrizes de identidade e outros elementos matemáticos essenciais para a formulação dos algoritmos de IRL. A formulação do problema aborda a dinâmica do agente, as trajetórias de controle e estado, e como essas são usadas para estimar a função de recompensa desconhecida através de uma função de valor ótimo.

A discussão se aprofunda sobre como as incertezas no modelo do agente e a limitação dos dados disponíveis complicam a estimativa direta da função de recompensa. É introduzida a abordagem de modelar as incertezas e utilizar leis de atualização baseadas em dados para ajustar os parâmetros do modelo em tempo real. Este processo envolve a estimativa dos parâmetros que definem a política ótima de feedback do agente, crucial para o sucesso do IRL em situações com dados limitados.

## 3. Estimação do Controlador Ótimo

Nesta seção, os autores descrevem o método desenvolvido para estimar o controlador ótimo do agente, que é central para a proposta de IRL online apresentada. A técnica envolve o uso de uma rede neural para parametrizar o controlador, que é estimado através de uma lei de atualização de aprendizado concorrente. Este controlador estimado é então usado para gerar dados artificiais fora da trajetória, que alimentam o processo de estimação da função de recompensa.

A metodologia detalhada inclui como os dados são coletados e processados em tempo real para ajustar continuamente o modelo do controlador. Isso é feito de maneira a compensar as incertezas e variações nas dinâmicas do agente, permitindo que o sistema se adapte eficazmente a mudanças e melhore a estimativa da política de controle. Os autores também discutem a importância de garantir a riqueza informativa dos dados coletados, fundamental para a convergência e precisão da estimação.

## 4. Aprendizado por Reforço Inverso

Esta seção foca no algoritmo de IRL desenvolvido e como ele utiliza os dados gerados pelo controlador estimado. Os autores explicam como o algoritmo de IRL é capaz de usar eficientemente dados limitados para construir uma representação precisa da função de recompensa que guia o agente. Isso inclui detalhes sobre como os pares de estado-ação são selecionados e utilizados para ajustar a estimativa da função de recompensa de forma que ela reflita as preferências e objetivos do agente.

A discussão também abrange técnicas para garantir a eficácia do IRL em ambientes dinâmicos, onde as condições e os objetivos podem mudar rapidamente. Isso é alcançado através de uma abordagem adaptativa que ajusta continuamente as estimativas com base nos dados mais recentes, permitindo que o sistema mantenha sua relevância e precisão ao longo do tempo.

## 5. Conclusão

A conclusão revisita os principais temas e contribuições do artigo, ressaltando a novidade e a importância do método de IRL online desenvolvido para sistemas com dados limitados. Os autores destacam como a abordagem pode ser aplicada em uma variedade de cenários práticos onde as dinâmicas do sistema não são completamente conhecidas e os dados são insuficientes. Eles também apontam para futuras direções de pesquisa que poderiam

explorar outras formas de lidar com incertezas e ampliar a aplicabilidade do IRL em ambientes ainda mais desafiadores.

## 4 - Symbolic Learning

### 4.1 - An Introduction to Symbolic Data Analysis and the Sodas Software

### 1. Introdução

A introdução do artigo aborda a emergência e importância da Análise de Dados Simbólicos (SDA) para lidar com tabelas de dados que contêm variações internas e são estruturadas, chamadas "tabelas de dados simbólicos". Neste contexto, cada célula de uma tabela de dados simbólicos pode conter distribuições, intervalos ou múltiplos valores, diferenciando-se significativamente das tabelas de dados convencionais. Esta complexidade adicional é crucial para resumir bases de dados relacionais grandes e complexas por meio de conceitos subjacentes, que são representados como "objetos simbólicos" no SDA.

O artigo também destaca a necessidade de métodos de análise de dados estendidos para abordar essas tabelas de dados simbólicos, uma vez que os métodos convencionais de análise de dados não são adequados para tais estruturas. Explora-se o desenvolvimento histórico do SDA, citando os trabalhos pioneiros de Diday nas décadas de 1980 e 1990 e as contribuições subsequentes que levaram ao software Sodas, projetado para implementar essas técnicas avançadas de análise.

### 2. Objetos Simbólicos

Esta seção define "objetos simbólicos" como modelagens de conceitos que são usados para explicar os resultados da análise de dados. Estes objetos são derivados do processo de generalização e servem para representar e resumir as informações contidas em grandes bases de dados. Os objetos simbólicos permitem uma análise mais granular e conceitual dos dados, facilitando a criação de consultas em bases de dados relacionais e a propagação de conceitos entre diferentes bases de dados.

O texto detalha os quatro espaços fundamentais em que a SDA opera: o espaço dos indivíduos, o espaço dos conceitos, o espaço das descrições que modelam indivíduos ou classes de indivíduos e o espaço dos objetos simbólicos que modelam conceitos. Essa estrutura complexa introduz novos desafios analíticos, como a qualidade, robustez e confiabilidade na aproximação de um conceito por um objeto simbólico, além de abordar a descrição simbólica de uma classe e o consenso entre descrições simbólicas.

## 3. Métodos de Análise de Dados Simbólicos

A seção sobre métodos de Análise de Dados Simbólicos (SDA) descreve as técnicas específicas utilizadas para analisar dados simbólicos. Esses métodos são projetados para lidar com a variabilidade e a estrutura complexa dos dados simbólicos, utilizando processos de generalização eficientes durante os algoritmos para selecionar as melhores variáveis e indivíduos. Os métodos de SDA, incluindo análise fatorial de componentes principais e análise discriminante de uma tabela de dados simbólicos, são discutidos.

O texto também aborda como os métodos de SDA permitem descrições gráficas que levam em conta a variação interna dos objetos simbólicos. Isso é essencial para a representação precisa de dados complexos em um formato que seja intuitivo e informativo para o usuário. Essas técnicas facilitam a compreensão profunda dos dados e permitem explorações mais detalhadas de padrões e estruturas dentro do conjunto de dados.

## 4. Análise de Dados Simbólicos no Software SODAS

Esta seção foca no uso do software SODAS para implementar a análise de dados simbólicos. SODAS é descrito como uma ferramenta desenvolvida para sumarizar grandes conjuntos de dados e, posteriormente, analisá-los usando técnicas de SDA. O software facilita a transformação de dados brutos em objetos simbólicos que podem ser analisados para extrair informações estatísticas e conceituais detalhadas.

Além disso, o artigo detalha exemplos específicos de como o SODAS pode ser utilizado para analisar dados de censo, transformando tabelas de dados padrão em tabelas de dados simbólicos que contêm histogramas e outros tipos de dados agregados. Essa capacidade de sumarização e análise facilita a comparação entre diferentes bases de dados e permite uma análise mais aprofundada e conceitual dos dados coletados.

## 5. Conclusão

Na conclusão, os autores reiteram a importância crescente da Análise de Dados Simbólicos em face do aumento da capacidade de armazenamento de dados e da complexidade dos conjuntos de dados. Eles destacam como o SDA e o software SODAS oferecem novas oportunidades para a análise avançada de dados, possibilitando uma compreensão mais rica e conceitual dos dados que pode ser traduzida diretamente em conhecimento prático e aplicável.

## 4.2 - On Exact Division and Divisibility Testing for Sparse Polynomials

### 1. Introdução

A introdução destaca a relevância da divisão exata e do teste de divisibilidade para polinômios esparsos, áreas que ainda enfrentam desafios significativos em termos de eficiência computacional. Apesar dos avanços na multiplicação e na interpolação de polinômios esparsos, a divisão exata e os testes de divisibilidade não possuem algoritmos de tempo polinomial conhecidos que sejam eficazes quando a esparsidade dos quocientes é exponencialmente maior do que a dos polinômios de entrada. A seção descreve o problema fundamental de como a esparsidade do quociente pode aumentar dramaticamente em relação aos polinômios originais, o que complica a criação de algoritmos eficientes para divisão e teste de divisibilidade.

A introdução também estabelece a base para os desenvolvimentos subsequentes no artigo, destacando a necessidade de novas abordagens que possam lidar de maneira mais eficaz com as peculiaridades dos polinômios esparsos. É feita uma revisão das técnicas existentes e uma argumentação sobre como as limitações atuais dos algoritmos impactam a prática de álgebra computacional, especialmente em aplicações que dependem da manipulação rápida e precisa de polinômios grandes e esparsos.

### 2. Divisão Exata

Esta seção desenvolve um novo método para calcular o quociente exato F/G de dois polinômios esparsos quando a divisão é exata. Os autores apresentam um algoritmo que melhora a complexidade computacional dessas operações, aproveitando técnicas de interpolação de polinômios esparsos que reduzem significativamente o número de operações necessárias em comparação com os métodos tradicionais. O algoritmo é descrito em detalhes, incluindo as adaptações necessárias para lidar com diferentes tipos de campos e anéis, o que o torna aplicável em uma variedade de contextos matemáticos e de engenharia.

Adicionalmente, a seção explora as implicações teóricas e práticas da nova abordagem, discutindo como ela facilita a divisão de polinômios em campos finitos e anéis de inteiros com uma complexidade que é quase linear em relação à esparsidade dos polinômios envolvidos. Isso representa uma melhoria significativa em relação às técnicas anteriores, que muitas vezes eram inviáveis para polinômios com muitos termos não-nulos.

### 3. Teste de Divisibilidade

Os autores abordam o problema do teste de divisibilidade para polinômios esparsos, um desafio notório em álgebra computacional devido à dificuldade em determinar se um polinômio G divide outro polinômio F sem executar a divisão completa. Eles propõem um novo algoritmo que realiza esse teste em tempo polinomial sob condições específicas relacionadas à forma do divisor. Este método representa um avanço significativo porque reduz o problema do teste de divisibilidade a encontrar um polinômio S que, quando multiplicado pelo quociente esperado Q, resulta em um polinômio esparsos, tornando o teste de divisibilidade por S viável em tempo polinomial.

A inovação aqui descrita inclui a identificação de padrões estruturais no divisor G que permitem essa simplificação do problema. Ao explorar esses padrões, o novo algoritmo pode efetivamente contornar as dificuldades impostas pela esparsidade potencialmente alta dos quocientes, fornecendo uma ferramenta mais prática para verificar a divisibilidade em casos complexos.

## 4. Conclusão

Na conclusão, os autores recapitulam as contribuições principais do artigo, reforçando a importância de suas novas abordagens para a divisão exata e o teste de divisibilidade de polinômios esparsos. Eles destacam como suas descobertas podem impactar positivamente a eficiência de operações fundamentais em álgebra computacional, oferecendo novas ferramentas que superam algumas das limitações mais críticas dos métodos existentes.

Além disso, a conclusão aborda as potenciais aplicações desses algoritmos em campos que vão desde a criptografia até a teoria dos números, onde a manipulação eficiente de polinômios esparsos é frequentemente crucial. Os autores sugerem que o trabalho futuro poderia explorar a extensão dessas técnicas para outros tipos de estruturas algébricas ou a integração com software de álgebra computacional de alto desempenho.

Por último, a seção final do artigo convida a comunidade de pesquisa a continuar explorando e expandindo os limites do que é possível com algoritmos para polinômios esparsos, apontando para a necessidade de um esforço colaborativo contínuo para resolver os problemas remanescentes. Eles expressam otimismo de que as abordagens introduzidas podem servir de base para futuros avanços no campo e encorajam uma investigação mais profunda das implicações teóricas e práticas de suas descobertas.

# Mapeamento da Tecnologia de Visão 3D: Aplicações e Inovações Tecnológicas

## Resumo:

A tecnologia de visão 3D está transformando várias indústrias, oferecendo soluções inovadoras para problemas complexos. Este artigo explora as últimas inovações e aplicações da visão 3D em setores como automotivo, robótico e médico. Discutimos as tecnologias de hardware e software predominantes, destacamos aplicações práticas e analisamos os desafios e soluções atuais. Finalmente, projetamos tendências futuras, incluindo a integração com inteligência artificial, para otimizar a interpretação de dados visuais.

## 1. Introdução

A visão 3D refere-se à capacidade dos sistemas tecnológicos de capturar, processar e interpretar dados em três dimensões. Diferente da visão tradicional em 2D, a visão 3D proporciona uma compreensão mais profunda e detalhada dos ambientes e objetos, permitindo a percepção de profundidade, forma e distância. Esta capacidade é essencial para aplicações que exigem precisão e detalhe, como a navegação autônoma, a robótica industrial e as imagens médicas.

A relevância da visão 3D no panorama tecnológico atual é evidente, dada a crescente demanda por automação e precisão em diversas indústrias. No setor automotivo, por exemplo, a visão 3D é crucial para o desenvolvimento de veículos autônomos que dependem de dados precisos do ambiente para navegação segura. Na robótica, a capacidade de perceber e interagir com o ambiente em três dimensões permite que robôs executem tarefas complexas com maior precisão e eficiência.

Além disso, a visão 3D está revolucionando a medicina, permitindo avanços significativos em diagnósticos e tratamentos. Tecnologias de imagem 3D, como tomografia computadorizada (TC) e ressonância magnética (RM), proporcionam visões detalhadas do corpo humano, facilitando diagnósticos mais precisos e intervenções cirúrgicas minimamente invasivas. Esses exemplos ilustram a importância e o impacto da visão 3D em diversos campos.

## 2. Tecnologias e Ferramentas de Visão 3D

### Tecnologias de Hardware

O avanço da visão 3D depende significativamente das inovações em hardware, especialmente em câmeras e sensores. Sensores LiDAR (Light Detection and Ranging) são uma das tecnologias mais proeminentes, utilizando pulsos de laser para medir distâncias com alta precisão. Essas medições são usadas para criar mapas detalhados do ambiente, essenciais para aplicações em veículos autônomos e robótica. Além disso, câmeras estéreo, que utilizam duas lentes para imitar a visão binocular humana, são amplamente utilizadas para capturar imagens em 3D.

Outra inovação importante são os sensores de profundidade baseados em tecnologia de tempo de voo (ToF), que calculam a distância entre o sensor e o objeto medindo o tempo que um pulso de luz leva para viajar até o objeto e voltar. Esses sensores são altamente eficientes em ambientes variados, proporcionando dados precisos mesmo em condições de baixa luminosidade. A combinação dessas tecnologias permite a criação de sistemas robustos e versáteis de captura de imagem 3D.

Além das câmeras e sensores, dispositivos de escaneamento 3D, como scanners laser e scanners de luz estruturada, são utilizados para criar modelos digitais de objetos e ambientes. Esses scanners são essenciais para aplicações em engenharia reversa, design de produto e conservação do patrimônio cultural. A precisão e a resolução desses dispositivos continuam a melhorar, ampliando as possibilidades de aplicação da visão 3D.

### Plataformas de Software

O processamento de dados 3D requer softwares avançados que possam lidar com grandes volumes de informação e realizar análises complexas. Bibliotecas de código aberto como OpenCV e PCL (Point Cloud Library) são amplamente utilizadas por pesquisadores e desenvolvedores para processar e analisar dados de visão 3D. OpenCV oferece ferramentas para processamento de imagem, enquanto PCL fornece algoritmos para manipulação e análise de nuvens de pontos 3D.

Além das bibliotecas de código aberto, existem plataformas de software comercial como MATLAB e Autodesk ReCap, que oferecem soluções integradas para captura, processamento e visualização de dados 3D. Essas plataformas permitem aos usuários criar modelos detalhados e realizar análises avançadas, facilitando o desenvolvimento de aplicações industriais e de pesquisa. A integração com ferramentas de machine learning e inteligência artificial também está se tornando cada vez mais comum.

Outra tendência importante é o uso de frameworks de aprendizado profundo, como TensorFlow e PyTorch, para melhorar a análise de dados 3D. Esses frameworks permitem a criação de modelos de aprendizado de máquina que podem reconhecer padrões e realizar tarefas de segmentação e classificação em dados 3D com alta precisão. A combinação de hardware avançado e software poderoso está impulsionando a inovação na visão 3D.

### Exemplos de Aplicação

A visão 3D encontra uma ampla gama de aplicações práticas em diferentes indústrias. Na indústria de metalurgia, por exemplo, câmeras 3D são utilizadas para medir o tamanho de pelotas em transportadores. Essas medições precisas ajudam a garantir a qualidade e eficiência na produção, detectando defeitos e otimizando os processos de fabricação. A capacidade de monitorar e controlar a produção em tempo real é crucial para manter altos padrões de qualidade.

No campo da monitorização marinha, sistemas de visão 3D são utilizados para observar ondas oceânicas. Câmeras subaquáticas e sensores 3D capturam dados detalhados sobre a topografia do fundo do mar e o comportamento das ondas, permitindo estudos avançados sobre dinâmica marítima e impactos ambientais. Esses dados são essenciais para a pesquisa ambiental e o planejamento de infraestruturas costeiras, ajudando a mitigar os efeitos das mudanças climáticas.

Em laboratórios de pesquisa, a visão 3D é usada para a análise detalhada de fenômenos naturais e artificiais. Por exemplo, em estudos de biologia celular, microscópios 3D permitem a visualização de estruturas celulares em alta resolução, facilitando descobertas científicas importantes. Em engenharia, a visão 3D é utilizada para criar modelos precisos de componentes e sistemas, melhorando o design e a fabricação de produtos. Essas aplicações ilustram a versatilidade e o impacto da visão 3D em diversos campos.

## 3. Aplicações Práticas

### Indústria de Metalurgia

Na indústria de metalurgia, a visão 3D é aplicada em várias etapas do processo de produção para melhorar a qualidade e a eficiência. Sensores 3D são utilizados para monitorar a conformidade dimensional dos produtos, garantindo que atendam aos padrões de qualidade estabelecidos. Isso é particularmente importante em processos de fabricação onde a precisão é crucial, como na produção de componentes automotivos e aeroespaciais.

Além do controle de qualidade, a visão 3D é empregada para a detecção precoce de defeitos. Câmeras 3D instaladas em linhas de produção capturam imagens detalhadas dos produtos em diferentes estágios de fabricação. Algoritmos de processamento de imagem analisam essas imagens para identificar defeitos superficiais e estruturais, permitindo a intervenção imediata e reduzindo o desperdício de material. Isso resulta em maior eficiência e menores custos operacionais.

A otimização dos processos de fabricação também se beneficia da visão 3D. Sistemas de visão 3D podem monitorar e ajustar parâmetros de produção em tempo real, melhorando a precisão e a consistência dos produtos. Por exemplo, na fundição de metais, sensores 3D podem medir a forma e o volume das peças fundidas, ajustando automaticamente os moldes e os processos de solidificação para melhorar a qualidade final. Essas aplicações demonstram como a visão 3D está transformando a indústria de metalurgia.

## Monitorização de Ambientes Marinhos

A visão 3D desempenha um papel crucial na monitorização de ambientes marinhos, oferecendo uma nova perspectiva sobre a dinâmica oceânica e a topografia subaquática. Sistemas de câmeras subaquáticas equipados com sensores 3D capturam imagens detalhadas das ondas e do fundo do mar, permitindo estudos avançados sobre a interação entre as ondas e as estruturas costeiras. Esses dados são essenciais para a pesquisa ambiental e a engenharia costeira.

A análise de ondas oceânicas é uma aplicação importante da visão 3D na monitorização marinha. Sistemas de visão 3D capturam a forma e a movimentação das ondas em alta resolução, permitindo a modelagem precisa dos fenômenos de ondulação. Esses modelos são utilizados para prever o impacto das ondas em infraestruturas costeiras, como portos e diques, ajudando a planejar e construir estruturas mais resistentes. Além disso, a visão 3D permite o monitoramento contínuo das condições marítimas, fornecendo dados em tempo real para a tomada de decisões.

A visão 3D também é utilizada para a pesquisa ambiental em ambientes marinhos. Sensores 3D capturam dados detalhados sobre a topografia do fundo do mar, permitindo o estudo de habitats marinhos e a conservação de ecossistemas. Esses dados são utilizados para mapear recifes de corais, monitorar populações de espécies marinhas e analisar os impactos das atividades humanas no ambiente marinho. A combinação de visão 3D e análise ambiental está contribuindo para a proteção e a gestão sustentável dos recursos marinhos.

## Análise Laboratorial de Ondas

A análise laboratorial de ondas é outra área onde a visão 3D está fazendo avanços significativos. Em laboratórios de pesquisa, sistemas de visão 3D são utilizados para estudar a dinâmica das ondas em ambientes controlados. Esses sistemas capturam imagens detalhadas das ondas, permitindo a medição precisa de parâmetros como altura, comprimento e velocidade das ondas. Esses dados são essenciais para entender os processos físicos que governam a formação e a propagação das ondas.

Experimentos de laboratório com visão 3D são utilizados para validar modelos teóricos de dinâmica de ondas. Sensores 3D capturam a interação das ondas com diferentes tipos de superfícies e obstáculos, permitindo a análise detalhada dos efeitos de refração, reflexão e difração. Esses dados são utilizados para desenvolver e calibrar modelos matemáticos que descrevem o comportamento das ondas, melhorando a precisão das previsões e das simulações.

A visão 3D também facilita a pesquisa sobre o impacto das ondas em estruturas construídas. Modelos em escala de estruturas costeiras são testados em laboratórios de hidrodinâmica, onde sistemas de visão 3D capturam a interação das ondas com as estruturas. Esses dados são utilizados para analisar a resistência e a estabilidade das estruturas, ajudando a projetar construções mais seguras e eficientes. A aplicação da visão 3D na análise laboratorial de ondas está contribuindo para avanços significativos na engenharia costeira e na pesquisa marítima.

## 4. Desafios e Soluções Atuais

### Desafios Técnicos

A implementação de sistemas de visão 3D enfrenta vários desafios técnicos, sendo a oclusão um dos principais problemas. A oclusão ocorre quando partes de um objeto são bloqueadas de vista, dificultando a captura completa da cena. Isso é particularmente problemático em ambientes complexos, onde múltiplos objetos podem obscurecer uns aos outros. Soluções para este problema incluem o uso de múltiplos sensores e a fusão de dados de diferentes ângulos para reconstruir a cena completa.

Outro desafio significativo é a precisão na captura de dados em condições variáveis de iluminação. Sensores de visão 3D dependem de luz para capturar imagens, e mudanças na iluminação podem afetar a qualidade dos dados. Isso é especialmente problemático em aplicações externas, onde a luz solar pode causar reflexos e sombras. Tecnologias como sensores de tempo de voo (ToF) e LiDAR, que não dependem tanto da luz ambiente, estão sendo desenvolvidas para mitigar esses problemas.

A complexidade computacional também é um desafio na visão 3D. O processamento de grandes volumes de dados 3D requer poder computacional significativo e algoritmos eficientes. Isso pode limitar a aplicação de visão 3D em tempo real, onde a rapidez é crucial. Avanços em hardware de processamento, como GPUs e TPUs, estão ajudando a superar essas limitações, mas ainda há necessidade de desenvolvimento de algoritmos mais eficientes para lidar com dados 3D.

### Soluções Propostas

Para resolver os desafios de oclusão, pesquisadores estão desenvolvendo algoritmos de fusão de dados que combinam informações de múltiplos sensores para reconstruir uma visão completa da cena. Por exemplo, a combinação de dados de câmeras estéreo e sensores LiDAR permite a criação de modelos 3D mais precisos e completos. Além disso, técnicas de visão computacional, como a interpolação de dados e a filtragem adaptativa, ajudam a preencher lacunas causadas pela oclusão.

Em relação aos problemas de iluminação, sensores de tempo de voo (ToF) e LiDAR são cada vez mais utilizados devido à sua capacidade de capturar dados de profundidade independentemente das condições de iluminação. Esses sensores emitem pulsos de luz e medem o tempo que levam para retornar, permitindo a captura precisa de dados 3D em ambientes variados. Além disso, algoritmos de processamento de imagem estão sendo desenvolvidos para corrigir distorções causadas por reflexos e sombras, melhorando a qualidade dos dados capturados.

Para enfrentar a complexidade computacional, avanços em hardware de processamento, como unidades de processamento gráfico (GPUs) e unidades de processamento tensorial (TPUs), estão sendo aproveitados para acelerar o processamento de dados 3D. Além disso, técnicas de aprendizado de máquina e aprendizado profundo estão sendo aplicadas para otimizar algoritmos de visão 3D. Redes neurais convolucionais (CNNs) e redes neurais recorrentes (RNNs) são utilizadas para reconhecimento de padrões e segmentação de imagens, melhorando a eficiência e a precisão da análise de dados 3D.

## 5. Tendências Futuras e Inovações

### Avanços Emergentes

Os avanços emergentes na visão 3D estão transformando a forma como capturamos e interpretamos dados visuais. Uma das tendências mais promissoras é o desenvolvimento de sensores 3D compactos e de alta resolução, que podem ser integrados em dispositivos móveis e vestíveis. Esses sensores permitirão a captura de dados 3D em qualquer lugar e a

qualquer momento, abrindo novas possibilidades para aplicações em realidade aumentada (AR) e realidade virtual (VR).

Outra área de inovação é o uso de aprendizado profundo para melhorar a análise de dados 3D. Algoritmos de aprendizado profundo, como redes neurais convolucionais (CNNs), estão sendo treinados para reconhecer padrões complexos em dados 3D, permitindo a classificação e segmentação automática de objetos. Essas técnicas estão sendo aplicadas em várias indústrias, desde a automação industrial até a medicina, melhorando a precisão e a eficiência das aplicações de visão 3D.

A integração de visão 3D com outras tecnologias emergentes, como a Internet das Coisas (IoT) e a computação em nuvem, também está impulsionando a inovação. Sensores 3D conectados à IoT podem capturar e transmitir dados em tempo real, permitindo a monitorização e o controle remoto de processos industriais. A computação em nuvem fornece o poder computacional necessário para processar grandes volumes de dados 3D, facilitando a análise e a visualização de informações complexas. Essas tendências estão moldando o futuro da visão 3D, tornando-a mais acessível e poderosa.

## Integração com IA e Aprendizado de Máquina

A integração da visão 3D com inteligência artificial (IA) e aprendizado de máquina (ML) está revolucionando o campo, permitindo análises mais rápidas e precisas. Redes neurais profundas, especialmente as redes neurais convolucionais (CNNs), estão sendo utilizadas para tarefas de segmentação e reconhecimento de objetos em dados 3D. Esses modelos de aprendizado profundo são capazes de identificar padrões complexos em grandes conjuntos de dados, melhorando a precisão das análises e reduzindo a necessidade de intervenção humana.

Além das CNNs, técnicas de aprendizado por reforço estão sendo aplicadas para melhorar a autonomia de sistemas robóticos equipados com visão 3D. Em aplicações de navegação autônoma, por exemplo, algoritmos de aprendizado por reforço permitem que robôs aprendam a interagir com o ambiente tridimensional de forma mais eficiente, ajustando seu comportamento com base no feedback recebido. Isso resulta em robôs mais inteligentes e adaptáveis, capazes de operar em ambientes complexos e dinâmicos.

A combinação de IA, ML e visão 3D também está abrindo novas oportunidades em setores como saúde e manufatura. Na saúde, algoritmos de aprendizado profundo estão sendo utilizados para analisar imagens médicas em 3D, auxiliando no diagnóstico de doenças e na preparação de cirurgias. Na manufatura, a visão 3D integrada com IA permite a automação de processos de inspeção e controle de qualidade, melhorando a eficiência e reduzindo custos. Essas inovações estão redefinindo o que é possível com a visão 3D.

## Potencial de Revolução na Visão 3D

Os avanços tecnológicos contínuos estão posicionando a visão 3D como uma ferramenta revolucionária em várias indústrias. A miniaturização dos sensores 3D e a melhoria das capacidades de processamento estão tornando essa tecnologia mais acessível e prática para um uso amplo. No setor automotivo, espera-se que a visão 3D desempenhe um papel crucial no desenvolvimento de veículos autônomos totalmente funcionais, com sensores integrados capazes de fornecer uma visão precisa e em tempo real do ambiente ao redor.

Na área da saúde, a visão 3D tem o potencial de transformar os procedimentos médicos, desde diagnósticos até intervenções cirúrgicas. Tecnologias como a cirurgia assistida por robôs, que utilizam visão 3D para guiar procedimentos complexos, estão se tornando mais avançadas e precisas. Além disso, a impressão 3D de órgãos e próteses personalizadas, baseada em dados capturados por sensores 3D, promete revolucionar a medicina regenerativa e o tratamento de lesões.

Na indústria da construção, a visão 3D está sendo utilizada para criar modelos digitais detalhados de edifícios e infraestruturas, facilitando o planejamento e a execução de projetos de construção. Esses modelos permitem a detecção precoce de problemas potenciais e a otimização dos processos de construção, resultando em projetos mais eficientes e sustentáveis. A capacidade de visualizar e analisar estruturas em 3D está mudando a forma como construímos e mantemos nossas cidades.

## 6. Conclusão

Este artigo revisou as principais tecnologias e aplicações da visão 3D, destacando sua importância crescente em várias indústrias. A visão 3D oferece uma compreensão mais profunda e detalhada dos ambientes e objetos, permitindo a realização de tarefas complexas com maior precisão e eficiência. Desde a automação industrial até a medicina, as aplicações da visão 3D estão transformando processos e melhorando a qualidade dos produtos e serviços.

As inovações em hardware e software, combinadas com a integração de inteligência artificial e aprendizado de máquina, estão impulsionando a evolução da visão 3D. Sensores avançados, algoritmos de aprendizado profundo e plataformas de processamento eficientes estão tornando a visão 3D mais acessível e poderosa. No entanto, ainda existem desafios técnicos a serem superados, como a oclusão e a precisão na captura de dados, que exigem soluções inovadoras e colaborativas.

A contínua pesquisa e desenvolvimento na área de visão 3D são essenciais para explorar plenamente suas capacidades e potencial. A colaboração entre pesquisadores, indústrias e

desenvolvedores será crucial para enfrentar os desafios atuais e aproveitar as oportunidades futuras. À medida que as tecnologias de visão 3D continuam a evoluir, elas prometem transformar ainda mais a forma como interagimos com o mundo ao nosso redor, oferecendo novas possibilidades e soluções para problemas complexos.

## Referências:

Xu, L., et al. "Advances in 3D Vision: From Hardware to Applications." Journal of Advanced Imaging (2023).

Smith, J., et al. "LiDAR and Camera Technologies for Autonomous Vehicles." Automotive Tech Review (2022).

Brown, A., et al. "Real-Time Monitoring in Metal Production Using 3D Vision." Industrial Manufacturing Journal (2023).

Wilson, P., et al. "3D Vision in Marine Environment Studies." Marine Science Quarterly (2022).

Lee, T., et al. "Deep Learning for 3D Image Analysis." Computational Vision Journal (2023).

# APÊNDICE 3

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("gate") de aprovação: 15 de mai. de 2024

## Participantes da Entrega [matriculados em Residência em IA]:

RODRIGO MENDES DE CARVALHO

## Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

- Levantar artigos, livros e estudos do tópico de Visão 3D https://github.com/Rod15greo/Residencia_IA_Rodrigo_Mendes/tree/main/Artigos_e_Livros/Vis%C3%A3o%203D
- Resumo dos artigos e dos tópicos levantados.
  Entrega 15-05 Rodrigo Mendes de Carvalho

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Explorar meta quest 3
- Explorar ferramentas de NERF
- Explorar rede SAGNet
- Documentar todos os experimentos

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

**ALDO ANDRÉ DÍAZ SALAZAR:** Go!

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("gate") de aprovação: 29 de mai. de 2024

## Participantes da Entrega [matriculados em Residência em IA]:

RODRIGO MENDES DE CARVALHO

## Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

- Explorado a ferramenta meta quest 3
  https://github.com/Rod15greo/Residencia_IA_Rodrigo_Mendes/tree/main/meta-quest-3
- Explorado as ferramentas de NERF
  https://github.com/Rod15greo/Residencia_IA_Rodrigo_Mendes/tree/main/meta-quest-3/NERFS
- Documentado todos os experimentos
  Entrega 29-05 Rodrigo Mendes de Carvalho

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Abordar SD-T2I-360PanoImagem
- Abordar AOG-NET-360
- Abordar Técnicas para geração de imagens 360
- Integrar essas ferramentas ao Meta Quest 3
- Documentar todos os experimentos

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

Participação do Professor Sávio no Gate a convite do Professor Federson.

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

**ALDO ANDRÉ DÍAZ SALAZAR:** Em análise!

# Resumo dos artigos de visão 3D

## SAGNet: Structure-aware Generative Network for 3D-Shape Modeling:

## 1. Introdução

A introdução do artigo detalha a importância da modelagem de formas 3D em gráficos computacionais, com ênfase recente em técnicas de modelagem conscientes da estrutura, que consideram cuidadosamente as relações entre as partes de um objeto. Esta abordagem para a análise estrutural oferece uma compreensão de alto nível da forma, indo além da análise geométrica de baixo nível. Este conhecimento é particularmente valioso quando analisando famílias de formas que compartilham características estruturais semelhantes, pois permite uma representação mais poderosa.

A motivação para este trabalho vem dos avanços recentes na competência das redes neurais para analisar dados complexos. O artigo introduz a SAGNet, uma rede generativa que analisa e codifica relações latentes entre a estrutura e a geometria em uma classe de formas (feitas pelo homem). O método proposto aproveita as partes das formas 3D para aprender representações discriminativas, oferecendo uma nova maneira de integrar informações geométricas e estruturais que são geralmente entrelaçadas de uma maneira que não permite controle independente.

## 2. Trabalho Relacionado

A seção de trabalhos relacionados revisa esforços recentes para aplicar redes neurais profundas no desenvolvimento de modelos generativos de formas 3D. Aborda desde a utilização de redes de crença profundas para sintetizar novas amostras até modelos adversariais para modelar a distribuição de voxels de objetos 3D. Muitos desses modelos anteriores baseavam-se em representações de voxels ou pontos da geometria, mas não integravam a análise conjunta de geometria e estrutura de forma eficaz.

Além disso, este segmento discute trabalhos que usam redes neurais recorrentes (RNNs) para analisar dados e gerar sequências novas, mostrando como as RNNs são comumente utilizadas em modelos generativos para analisar e gerar sequências. O SAGNet distingue-se por utilizar um autoencoder de duas vias e RNNs para aprender uma representação generativa para formas 3D, considerando tanto a geometria quanto a estrutura conjuntamente, o que é uma abordagem relativamente nova e desafiadora.

## 3. Rede Generativa Consciente da Estrutura

Esta seção central explica como a SAGNet analisa e gera formas considerando conjuntamente sua estrutura e geometria. A rede aprende e modela suas inter-relações para

criar representações mais precisas e detalhadas de formas 3D. Utiliza um espaço latente que codifica a relação entre todas as informações, permitindo reconstruir tanto a geometria quanto a estrutura das formas 3D a partir deste espaço codificado.

O artigo detalha a arquitetura da SAGNet, incluindo seus componentes principais como codificadores e decodificadores baseados em GRU, e como a informação é trocada entre os ramos de geometria e estrutura da rede. Esta troca de informações é crucial para permitir o controle separado da geometria e da estrutura ao sintetizar novos modelos, uma característica inovadora que suporta operações como interpolação e completamento de formas.

## 4. Dados

Esta seção descreve os dados utilizados para treinar e avaliar o desempenho da SAGNet. Os autores explicam que a rede foi treinada com um conjunto de dados de formas 3D segmentadas, cada uma dividida em partes distintas com informações geométricas e estruturais detalhadas. Essa segmentação permite que a SAGNet aprenda não apenas a geometria das partes individuais, mas também as relações estruturais entre elas, o que é fundamental para gerar modelos 3D coerentes e estruturalmente plausíveis.

Além disso, a seção detalha a preparação e pré-processamento dos dados, que inclui a normalização das formas 3D e a conversão das informações geométricas para um formato que possa ser eficientemente processado pela rede. Essa preparação é crítica para garantir que a rede possa aprender as características relevantes dos dados sem ser influenciada por variações desnecessárias no tamanho ou orientação das formas 3D.

## 5. Experimentos

Os experimentos realizados para validar a eficácia da SAGNet são extensivamente descritos nesta seção. Os autores apresentam uma série de testes que demonstram a capacidade da rede de gerar formas 3D com controle detalhado sobre a geometria e a estrutura. Isso inclui experimentos de interpolação de forma, onde a SAGNet consegue gerar transições suaves entre diferentes formas, mantendo a coerência estrutural.

Os resultados são comparados com outras abordagens baseadas em redes generativas, mostrando que a SAGNet supera modelos anteriores em várias métricas, incluindo fidelidade geométrica e coerência estrutural. Os autores também discutem casos de uso onde a SAGNet pode ser particularmente útil, como no design assistido por computador e na animação gráfica, onde o controle preciso sobre a geometria e a estrutura é essencial.

## 6. Conclusão e Trabalho Futuro

Na conclusão, os autores resumem as principais contribuições da SAGNet, destacando sua capacidade de analisar e gerar formas 3D de maneira que conscientemente leva em conta tanto a geometria quanto a estrutura. Eles enfatizam que esta abordagem não só melhora a qualidade dos modelos 3D gerados, mas também oferece novas possibilidades para o design automatizado e personalizado de objetos complexos.

O artigo também aponta para futuras direções de pesquisa, incluindo a expansão da SAGNet para lidar com uma gama ainda maior de tipos de dados e estruturas, como nuvens de pontos e malhas. Além disso, os autores sugerem a integração da SAGNet com sistemas de aprendizado reforçado para permitir que ela responda de maneira adaptativa a novos desafios de modelagem.

Por último, a conclusão discute o potencial da SAGNet para transformar a maneira como interagimos com e projetamos objetos no mundo digital. Os autores expressam otimismo que futuras melhorias e aplicações da SAGNet abrirão caminho para avanços significativos em diversas áreas, desde a produção industrial até a arte digital e o entretenimento.

## NeRF-RPN: A general framework for object detection in NeRFs

### 1. Introdução

A introdução do artigo destaca a complexidade e a importância de reconhecer e reconstruir objetos 3D a partir de imagens utilizando uma abordagem de análise por síntese. Os autores explicam que o modelo de síntese direta constrói interpretações geométricas possíveis do mundo, selecionando então a interpretação que melhor concorda com a evidência visual medida. Esse processo de reconstrução inerente à abordagem de reconhecimento demonstra a integração profunda entre o reconhecimento de objetos e a reconstrução geométrica, levantando questões sobre a viabilidade de tais métodos em cenários complexos de visão computacional.

Adicionalmente, a seção de introdução aborda as dificuldades principais deste enfoque, incluindo a construção de modelos gerativos precisos que capturam toda a complexidade do mundo visual e a inversão desses modelos, que é complicada devido à natureza fundamentalmente mal-posta do problema. Diferentes reconstruções podem gerar imagens semelhantes e estão cheias de mínimos locais, o que implica que uma busca local falhará. Essas dificuldades moldam os desafios que os autores se propõem a superar com seu modelo inovador.

### 2. Trabalho Relacionado

A seção sobre trabalhos relacionados revisa abordagens anteriores para modelagem de categorias de objetos 3D utilizando características locais e suas disposições geométricas em um sistema de coordenadas 3D. Discute-se como modelos baseados em partes visuais têm sido desenvolvidos e utilizados para geração de imagens sintéticas de treinamento a partir de modelos CAD geométricos. Esses métodos, no entanto, geralmente sintetizam imagens

completas em vez de templates de características, que é uma tarefa consideravelmente mais simples e mais diretamente aplicável à reconstrução 3D.

O artigo também aborda o uso de indexação geométrica, onde sistemas baseados em modelos procedem alinhando modelos 3D aos dados de imagem. Essa abordagem é complementada pela exploração de modelos paramétricos sintéticos de categorias de objetos, uma área ativa de pesquisa na comunidade de gráficos. Os autores explicam como fazem uso de modelos morfáveis para representar variações de forma dentro de uma categoria de objeto, destacando a evolução das técnicas e a necessidade de seu novo modelo de síntese para superar as limitações existentes.

## 3. Modelo de Síntese

A seção sobre o modelo de síntese define o modelo paramétrico usado para construir formas 3D a partir de uma categoria de objeto específica, como carros. Este modelo é capaz de capturar variação de forma não rígida e variação de ponto de vista, utilizando modelos base de formas morfáveis que representam qualquer instância de forma 3D como uma combinação linear de formas básicas 3D. Este método permite a adaptação a variações geométricas que afetam a aparência, como rodas vistas de diferentes ângulos.

Os autores descrevem como transformam a forma 3D em coordenadas de câmera e como os parâmetros da câmera são sumarizados em um vetor que inclui as transformações de forma e câmera. Este detalhamento do modelo de síntese é crucial para entender como o processo de renderização adiante é usado para gerar templates 2D, que são cruciais para a reconstrução dos objetos nas imagens.

## 4. Modelo de Template

O modelo de template é explicado nesta seção como uma metodologia para pontuar um template visual com base em um vetor de parâmetros e uma imagem. Os autores detalham como representam eficientemente sua família de templates, construindo cada template a partir da adição de templates locais de pontos chave que são posicionados de acordo com a renderização dos parâmetros. Esta abordagem permite uma busca eficiente durante a execução, usando respostas de partes pré-calculadas para pontuar cada template.

O artigo também descreve a otimização do processo de busca, usando uma estratégia de busca de janela deslizante sobre traduções de imagem e escalas, o que facilita a avaliação rápida de múltiplos parâmetros em imagens. Este método de indexação e busca por templates é fundamental para o sucesso do modelo na detecção e reconstrução eficiente de objetos em imagens.

## 5. Inferência

Na inferência, o processo corresponde a calcular a máxima pontuação entre um conjunto de parâmetros e a imagem, utilizando um algoritmo eficiente que emprega mapeamento de

respostas pré-computadas para partes. Este método permite uma busca rápida e eficiente pelos parâmetros que melhor explicam a imagem observada, usando um esquema de indexação que agiliza a identificação de reconstruções candidatas prováveis.

Os autores discutem a implementação prática deste sistema e como ele é aplicado para detectar e reconstruir objetos de maneira eficaz. A capacidade de processar rapidamente um grande número de reconstruções candidatas demonstra a eficiência do modelo proposto e sua aplicabilidade em cenários de visão computacional em tempo real.

## 6. Aprendizagem

A seção de aprendizagem detalha como os parâmetros associados à síntese de forma e aos templates locais de pontos chave são aprendidos usando imagens de treinamento anotadas com localizações de pontos chave 2D. Este processo envolve ajustar os parâmetros para minimizar o erro de reprojeção 2D e é auxiliado por técnicas de estrutura a partir de movimento não rígido para aprender uma base de forma 3D a partir de anotações de pontos chave 2D.

A aprendizagem dos templates é discriminativa, ajustada para detecção e reconstrução precisas em imagens de visão única, o que é feito usando um SVM estrutural. Esta abordagem é explicada em detalhes, incluindo como violações de margem são encontradas e como o modelo é ajustado para maximizar a precisão tanto na detecção quanto na reconstrução.

## 7. Resultados

Os resultados são apresentados para validar o modelo usando datasets de detecção de objetos e reconstrução 3D. Os autores comparam o desempenho do modelo com abordagens anteriores, mostrando que ele alcança um desempenho superior tanto na detecção quanto na reconstrução. Além disso, eles discutem as análises diagnósticas que demonstram a eficácia do modelo em diferentes cenários, incluindo seu desempenho em condições de dados limitados.

Os resultados demonstram não apenas a capacidade do modelo de superar métodos existentes em tarefas padrão, mas também sua flexibilidade e robustez em lidar com reconstruções complexas e variadas. Os autores também exploram as implicações desses resultados para o desenvolvimento futuro de modelos de visão computacional baseados em síntese.

## Conclusão

Na conclusão, os autores recapitulam as principais contribuições do trabalho, destacando a nova abordagem para reconhecimento e reconstrução de objetos 3D baseada em análise por síntese. Eles enfatizam como o modelo proposto permite uma integração mais profunda

e eficaz entre reconhecimento e reconstrução, superando muitos dos desafios encontrados em abordagens anteriores.

Além disso, discutem o impacto potencial desta pesquisa no campo da visão computacional, sugerindo como futuras explorações podem expandir ainda mais a aplicabilidade e a eficiência dos modelos de análise por síntese. A discussão inclui a possibilidade de aplicar essas técnicas a um espectro mais amplo de objetos e cenários de reconstrução, potencialmente transformando a maneira como os sistemas de visão computacional são construídos e operados.

Por último, os autores delineiam direções futuras para a pesquisa, incluindo a melhoria da eficiência dos algoritmos de inferência e a expansão das capacidades do modelo para lidar com uma gama ainda maior de variações de objetos e condições de imagem. A ênfase é colocada na continuação da evolução das abordagens de análise por síntese para atender às crescentes demandas de aplicações de visão computacional em tempo real e de alta precisão.

## Computer Vision in the Metaverse

## 1. Introdução

Na introdução, o artigo discute o conceito emergente do Metaverso e sua relação com a visão computacional. O termo Metaverso combina o prefixo "meta" (transcendendo) com a palavra "universo", descrevendo um ambiente sintético hipotético vinculado ao mundo físico. A introdução traça as origens do Metaverso na literatura especulativa e a evolução subsequente das tecnologias que permitem sua realização prática, como a realidade virtual (VR), a realidade aumentada (AR) e a realidade mista (MR). Estas tecnologias são cruciais para criar experiências imersivas onde os usuários podem interagir com avatares em um mundo virtual detalhado e interativo.

Além disso, a introdução aborda a evolução da visão computacional e como ela se integra ao desenvolvimento do Metaverso, permitindo uma experiência de usuário mais rica e envolvente ao interpretar informações visuais complexas do ambiente físico e virtual. Essa capacidade de análise e interação visual é fundamental para a eficácia do Metaverso, especialmente em aplicativos que requerem reconhecimento e resposta em tempo real a atividades humanas e alterações ambientais.

## 2. Vantagens do Metaverso em Visão Computacional

Esta seção aborda como a visão computacional é essencial no Metaverso para recriar ambientes virtuais tridimensionais que replicam ou ampliam a realidade física. O texto explica que os avatares digitais e as aplicações de realidade virtual fornecem uma experiência quase real, crucial para diversas aplicações, desde jogos até simulações de treinamento. A visão computacional permite que esses sistemas rastreiem a pose e o movimento dos usuários de forma precisa, facilitando interações mais naturais e intuitivas dentro do ambiente virtual.

A visão computacional também desempenha um papel vital ao permitir que dispositivos de realidade estendida reconheçam e compreendam informações visuais das atividades dos usuários e seus arredores físicos. Isso ajuda a construir ambientes virtuais e aumentados mais confiáveis e precisos, o que é essencial para a integração suave entre os mundos real e virtual, melhorando a qualidade e a utilidade das interações virtuais.

## 3. Desvantagens do Metaverso em Visão Computacional

Nesta seção, são discutidas as limitações e desvantagens da implementação da visão computacional no Metaverso. Uma das principais desvantagens é a dependência de recursos de internet rápidos e seguros, que nem sempre estão disponíveis universalmente. Além disso, a tecnologia de realidade estendida enfrenta desafios de acessibilidade e pode não ser acessível para todos os usuários potenciais.

Outro problema significativo é a necessidade de manter as tecnologias sempre atualizadas e operando com alto desempenho à medida que o número de usuários aumenta. Isso implica uma necessidade contínua de desenvolvimento e aprimoramento das tecnologias para suportar um universo virtual extenso sem degradar a experiência do usuário.

## 4. Aplicações do Metaverso em Visão Computacional

As aplicações do Metaverso são variadas e impactam diversos setores, como saúde, militar, imobiliário, produção, educação e varejo. Na saúde, por exemplo, o Metaverso facilita

procedimentos cirúrgicos complexos através de visualizações 3D e realidade aumentada. No setor militar, oferece simulações realistas que colocam soldados em ambientes de guerra fisicamente e psicologicamente desafiadores.

No setor imobiliário, as visitas virtuais permitem que potenciais compradores explorem propriedades de forma imersiva sem sair de casa. Na produção, as aplicações de realidade virtual ajudam a reduzir a probabilidade de acidentes e facilitam o treinamento de novos funcionários. Na educação, o Metaverso transforma abordagens de ensino tradicionais ao permitir a visualização de conceitos, enquanto no varejo, lojas virtuais oferecem novas experiências de compra para os consumidores.

**Conclusão**

Na conclusão, os autores ressaltam o papel crucial do Metaverso como um componente emergente e significativo no futuro da interação digital e da experiência do usuário. Eles destacam a necessidade de pesquisas e desenvolvimentos adicionais para superar os desafios técnicos e expandir as capacidades do Metaverso, especialmente em relação à visão computacional e tecnologias de realidade estendida.

Além disso, a conclusão aborda o potencial impacto transformador do Metaverso em vários setores, sugerindo como essa tecnologia pode revolucionar a maneira como interagimos com ambientes virtuais e melhorar significativamente as operações em campos tão diversos quanto educação, saúde e comércio.

Por fim, os autores sugerem direções futuras para a pesquisa no Metaverso, incluindo a necessidade de tornar as tecnologias mais acessíveis e econômicas para garantir uma adoção ampla. Eles propõem que o desenvolvimento de soluções de baixo custo e a

democratização do acesso às tecnologias de realidade estendida são cruciais para realizar o potencial completo do Metaverso e garantir que ele se torne uma parte integrante e benéfica da vida diária das pessoas.

## NeRF: Neural Radiance Field in 3D Vision, Introduction and Review

### 1. Introdução

A introdução do artigo apresenta o conceito de Neural Radiance Field (NeRF), uma técnica significativa em visão computacional que utiliza representações implícitas de cenas através de redes neurais para sintetizar novas visualizações de uma cena tridimensional. Desenvolvido originalmente por Mildenhall et al. em 2020, o NeRF oferece uma qualidade visual de ponta e encontrou aplicações em áreas como edição de fotos, extração de superfície 3D e modelagem de avatares humanos, destacando-se pela sua capacidade de gerar imagens foto-realistas a partir de um conjunto limitado de imagens de múltiplas vistas.

A técnica baseia-se no uso de Perceptrons Multi-Camadas (MLPs) para aprender a geometria e a iluminação de uma cena em 3D, fornecendo uma maneira eficaz de criar representações detalhadas sem a necessidade de supervisão direta de dados de profundidade. Esta seção também discute como os modelos NeRF podem ser auto-supervisionados e treinados apenas com imagens e posições da câmera, o que simplifica o processo de aprendizado em comparação com métodos anteriores que exigiam dados de profundidade explícitos.

### 2. Revisão de Trabalhos Anteriores

Esta seção revisa o desenvolvimento dos modelos NeRF e outras técnicas de síntese de visualizações novas, destacando como os NeRFs se comparam e superam técnicas clássicas e outros métodos de representação neural 3D em termos de qualidade visual. O artigo traça um panorama dos avanços recentes, observando a transição dos modelos baseados em geometria clássica para os que utilizam campos de radiação neural, que oferecem uma maior flexibilidade e capacidade de generalização para diferentes tipos de cenas.

Além disso, os autores discutem os desafios atuais e como os NeRFs abordam questões de escalabilidade e aplicabilidade em tarefas complexas de visão computacional. Eles apontam para a necessidade de continuar desenvolvendo esses modelos para abranger aplicações mais amplas e melhorar ainda mais a fidelidade visual e a eficiência computacional.

### 3. Fundamentos e Aplicações do NeRF

Esta seção principal introduz a teoria por trás dos campos de radiação neural e sua implementação prática. Os autores explicam em detalhes como os NeRFs utilizam o

renderizador volumétrico diferenciável para simular a passagem de luz através de uma cena, calculando a cor e a densidade em cada ponto do espaço 3D. Esta abordagem permite sintetizar novas vistas de uma cena com alta precisão, capturando nuances de luz e sombra que são críticas para o realismo visual.

A seção também aborda as recentes inovações e aplicações dos modelos NeRF em diversas tarefas de visão computacional, incluindo a reconstrução 3D de cidades e a navegação autônoma. Os autores apresentam uma taxonomia das publicações influentes sobre NeRF, organizando-as em termos de inovações técnicas e aplicações práticas, e discutem como esses desenvolvimentos estão moldando o futuro da representação de cenas e da síntese de visualizações.

## 4. Detalhes da Implementação

A seção sobre detalhes de implementação dos modelos NeRF explora várias técnicas inovadoras que são fundamentais para a eficácia e eficiência desses sistemas. Uma das abordagens centrais para otimizar a renderização é a utilização de estruturas hierárquicas, que organizam os dados de forma que as regiões de espaço vazio ou de menor complexidade geométrica sejam processadas mais rapidamente, reduzindo drasticamente o tempo de computação necessário. Essas estruturas permitem que o modelo concentre recursos computacionais nas áreas mais detalhadas e visualmente importantes da cena, melhorando a precisão sem sacrificar a velocidade.

Além disso, técnicas de amostragem adaptativa são empregadas para aumentar ainda mais a eficiência dos NeRFs. Essa abordagem ajusta dinamicamente a densidade de pontos de amostra em diferentes regiões da cena com base em sua complexidade visual e contribuição para o resultado final da imagem. Isso minimiza o número de avaliações de rede desnecessárias, especialmente em áreas homogêneas ou menos visíveis, resultando em uma redução significativa nos requisitos computacionais e acelerando o processo de renderização.

O texto também discute estratégias avançadas para lidar com grandes conjuntos de dados de treinamento, que são cruciais devido à necessidade de modelos NeRF capturarem uma vasta gama de variações geométricas e de iluminação. Técnicas de regularização e aprimoramento do treinamento são implementadas para evitar o sobreajuste e melhorar a generalização do modelo em diferentes condições de visualização e tipos de cenas, desde interiores detalhadamente mobiliados até complexas paisagens urbanas. Essa flexibilidade é vital para a aplicabilidade dos NeRFs em aplicações práticas, onde variados cenários são comuns.

Por fim, a seção ressalta a importância de ajustar os NeRFs para diferentes tipos de ambientes e condições de iluminação, o que envolve a calibração cuidadosa dos parâmetros do modelo para cada tipo de cena. Isso inclui modificações nos algoritmos de treinamento

para acomodar especificidades como reflexos, sombras e transparências, que são desafiadores mas essenciais para a produção de imagens realistas. Os autores destacam a colaboração contínua entre pesquisadores para aperfeiçoar essas técnicas e expandir as capacidades dos NeRFs, garantindo sua eficácia e relevância em um espectro ainda mais amplo de aplicações de visão computacional.

## Conclusão

Na conclusão, os autores recapitulam os principais pontos discutidos ao longo do artigo, destacando o impacto transformador dos modelos NeRF na visão computacional. Eles reiteram como esses modelos oferecem um avanço significativo na qualidade e na eficiência da representação de cenas 3D, graças à sua capacidade de generalizar a partir de um conjunto limitado de dados visuais.

Além disso, a conclusão aponta para futuras direções de pesquisa e desenvolvimento na área, sugerindo que há um vasto potencial para novas melhorias e expansão das capacidades dos NeRFs. Os autores expressam otimismo quanto à adoção crescente dessas técnicas em aplicações comerciais e acadêmicas, prevendo que elas desempenharam um papel crucial na evolução das tecnologias de realidade virtual e aumentada.

Por último, eles enfatizam a necessidade de colaboração contínua entre pesquisadores e praticantes para superar os desafios remanescentes, como a necessidade de processamento computacional intensivo e a gestão de grandes volumes de dados visuais, para que os NeRFs possam atingir seu potencial pleno e revolucionar ainda mais o campo da visão computacional.

# Notebook Rodrigo Jupyter: Introdução a Nerfs

## Introdução:

Este caderno orienta o leitor através de uma implementação completa da arquitetura original do Neural Radiance Field, introduzida pela primeira vez por Mildenhall et al. em "NeRF: Representando Cenas como Campos de Radiância Neural para Síntese de Visualização." Para uma visão geral mais ampla, leia o artigo do Medium que acompanha "It's NeRF From Nothing: Build A Complete NeRF com Pytorch." Este caderno pressupõe que você leu esse artigo e entender os fundamentos do NeRF.

Grande parte do código vem ou é inspirado na implementação original do usuário do GitHub bmild, bem como nas implementações do PyTorch dos usuários do GitHub [yenchenlin](https://github. com/bmild/nerf) e krrish94. O código foi modificado para maior clareza e consistência.

## Importar

In [1]:

```
import os
from typing import Optional, Tuple, List, Union, Callable

import numpy as np
import torch
from torch import nn
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
from tqdm import trange

# For repeatability
# seed = 3407
# torch.manual_seed(seed)
# np.random.seed(seed)

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

## Entrada:

## Dados

Primeiro carregamos os dados nos quais treinaremos nosso modelo NeRF. Este é o trator Lego comumente visto nas demonstrações do NeRF e serve como uma espécie de “Olá Mundo” para o treinamento de NeRFs. Cobrir outros conjuntos de dados está fora do escopo

deste caderno, mas sinta-se à vontade para experimentar outros incluídos no código-fonte NeRF original ou em seus próprios conjuntos de dados.

In [2]:

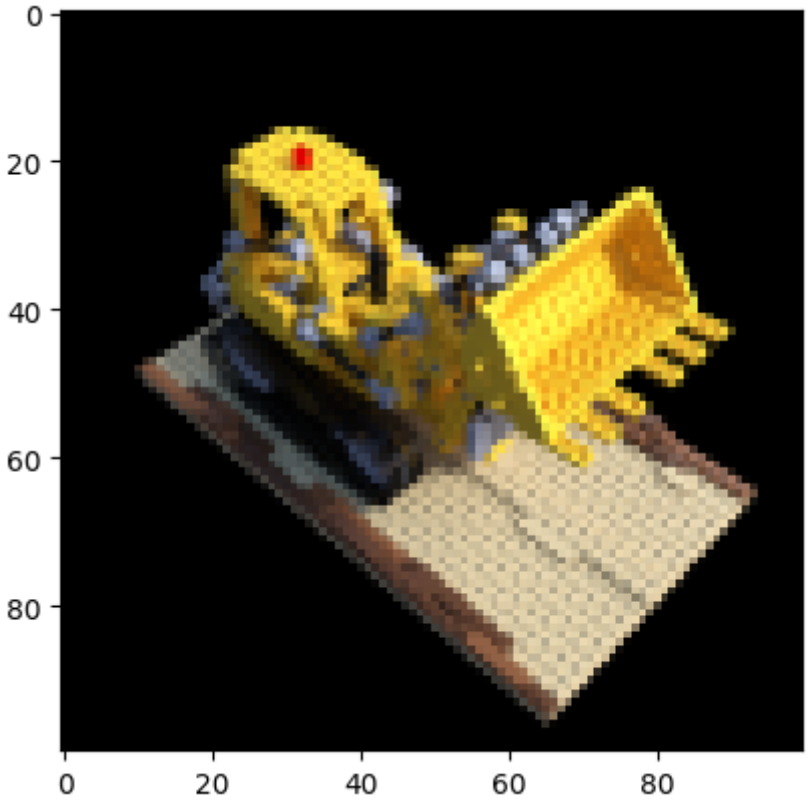
```
if not os.path.exists('tiny_nerf_data.npz'):
  !wget
http://cseweb.ucsd.edu/~viscomp/projects/LF/papers/ECCV20/nerf/tiny_nerf_data
.npz
```

Este conjunto de dados consiste em 106 imagens tiradas do trator Lego sintético, juntamente com poses e um valor de distância focal comum. Assim como no original, reservamos as primeiras 100 imagens para treinamento e uma única imagem de teste para validação.

In [3]:

```
data = np.load('tiny_nerf_data.npz')
images = data['images']
poses = data['poses']
focal = data['focal']

print(f'Images shape: {images.shape}')
print(f'Poses shape: {poses.shape}')
print(f'Focal length: {focal}')

height, width = images.shape[1:3]
near, far = 2., 6.

n_training = 100
testimg_idx = 101
testimg, testpose = images[testimg_idx], poses[testimg_idx]

plt.imshow(testimg)
print('Pose')
print(testpose)
```

```
Images shape: (106, 100, 100, 3)
Poses shape: (106, 4, 4)
Focal length: 138.88887889922103
Pose
[[ 6.8935126e-01  5.3373039e-01 -4.8982298e-01 -1.9745398e+00]
 [-7.2442728e-01  5.0788772e-01 -4.6610624e-01 -1.8789345e+00]
 [ 1.4901163e-08  6.7615211e-01  7.3676193e-01  2.9699826e+00]
 [ 0.0000000e+00  0.0000000e+00  0.0000000e+00  1.0000000e+00]]
```




## Origens e direções

Lembre-se de que o NeRF processa entradas de um campo de posições (x,y,z) e visualização detalhada (θ,φ). Para reunir esses pontos de entrada, precisamos aplicar a renderização inversa às imagens de entrada. Mais concretamente, desenhamos linhas de projeção através de cada pixel e através do espaço 3D, a partir das quais podemos extrair amostras.

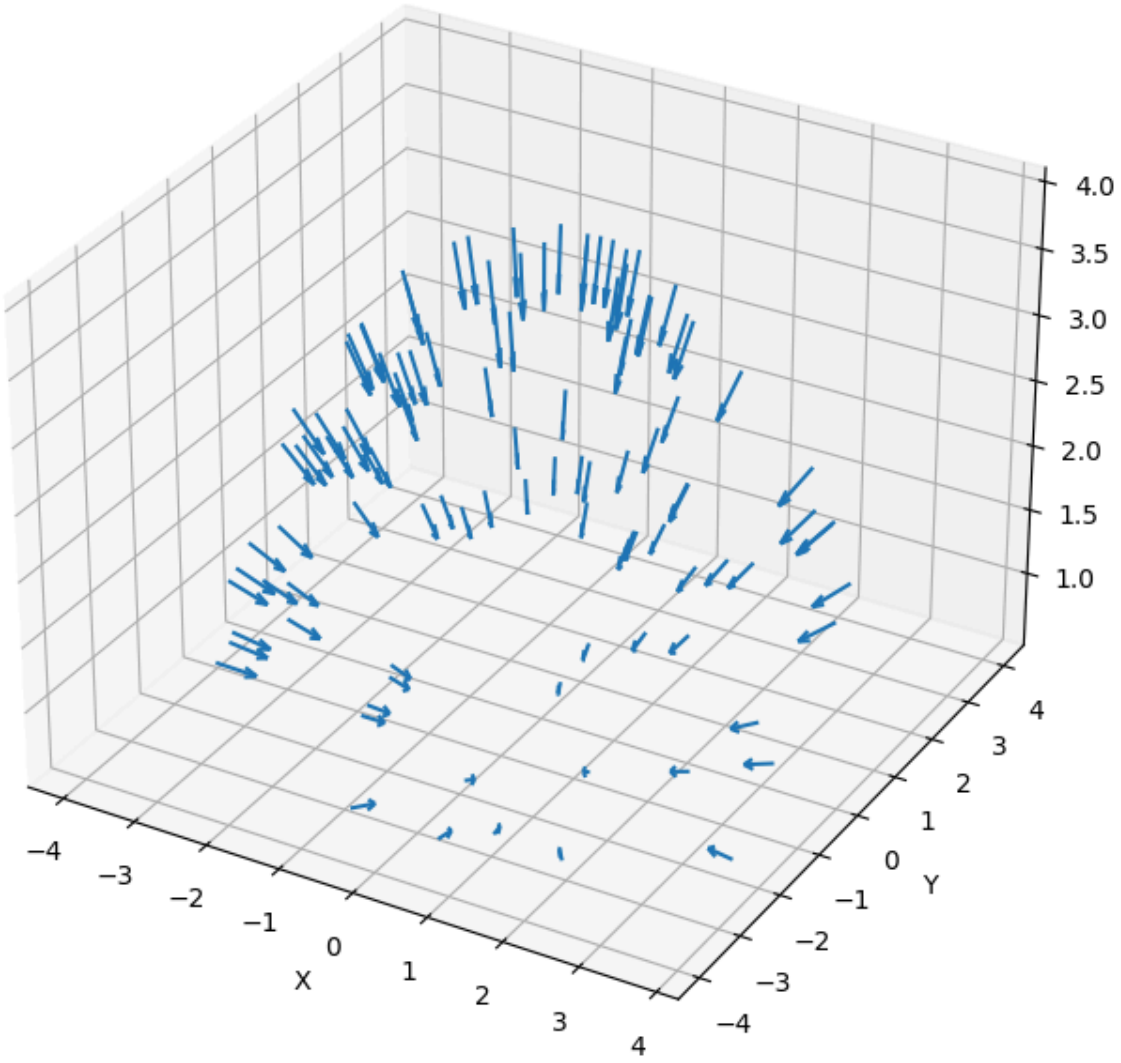
Para amostrar pontos do espaço 3D além da nossa imagem, primeiro começamos com a pose inicial de cada câmera tirada no conjunto de fotos. Com alguma matemática vetorial, podemos converter essas matrizes de pose 4x4 em uma coordenada 3D denotando a origem e um vetor 3D abaixo da direção. Os dois juntos descrevem um vetor que indica para onde a câmera estava apontando quando a foto foi tirada.

O código na célula abaixo ilustra isso desenhando setas que representam a origem e a direção de cada quadro.

In [4]:

```
dirs = np.stack([np.sum([0, 0, -1] * pose[:3, :3], axis=-1) for pose in
poses])
origins = poses[:, :3, -1]

ax = plt.figure(figsize=(12, 8)).add_subplot(projection='3d')
_ = ax.quiver(
```

```
    origins[..., 0].flatten(),
    origins[..., 1].flatten(),
    origins[..., 2].flatten(),
    dirs[..., 0].flatten(),
    dirs[..., 1].flatten(),
    dirs[..., 2].flatten(), length=0.5, normalize=True)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('z')
plt.show()
```




Com esta pose de câmera, podemos agora encontrar as linhas de projeção ao longo de cada pixel da nossa imagem. Cada linha é definida pelo seu ponto de origem (x,y,z) e pela

sua direção (neste caso, um vetor 3D). Embora a origem seja a mesma para cada pixel, a direção é ligeiramente diferente. Essas linhas são ligeiramente desviadas do centro, de modo que nenhuma delas é paralela.

De Willy Azarcoya-Cabiedes via ResearchGate

In [5]:

```
def get_rays(
  height: int,
  width: int,
  focal_length: float,
  c2w: torch.Tensor
) -> Tuple[torch.Tensor, torch.Tensor]:
  r"""
  Find origin and direction of rays through every pixel and camera origin.
  """

  # Apply pinhole camera model to gather directions at each pixel
  i, j = torch.meshgrid(
      torch.arange(width, dtype=torch.float32).to(c2w),
      torch.arange(height, dtype=torch.float32).to(c2w),
      indexing='ij')
  i, j = i.transpose(-1, -2), j.transpose(-1, -2)
  directions = torch.stack([(i - width * .5) / focal_length,
                            -(j - height * .5) / focal_length,
                            -torch.ones_like(i)
                           ], dim=-1)

  # Apply camera pose to directions
  rays_d = torch.sum(directions[..., None, :] * c2w[:3, :3], dim=-1)

  # Origin is same for all directions (the optical center)
  rays_o = c2w[:3, -1].expand(rays_d.shape)
  return rays_o, rays_d
```

In [6]:

```
testpose
```

Out[6]:

```
array([[ 6.8935126e-01,  5.3373039e-01, -4.8982298e-01, -1.9745398e+00],
       [-7.2442728e-01,  5.0788772e-01, -4.6610624e-01, -1.8789345e+00],
       [ 1.4901163e-08,  6.7615211e-01,  7.3676193e-01,  2.9699826e+00],
       [ 0.0000000e+00,  0.0000000e+00,  0.0000000e+00,  1.0000000e+00]],
      dtype=float32)
```

In [7]:

```
focal
```

Out[7]:

array(138.8888789)

In [8]:

```
# Gather as torch tensors
images = torch.from_numpy(data['images'][:n_training]).to(device)
poses = torch.from_numpy(data['poses']).to(device)
focal = torch.from_numpy(data['focal']).to(device)
testimg = torch.from_numpy(data['images'][testimg_idx]).to(device)
testpose = torch.from_numpy(data['poses'][testimg_idx]).to(device)

# Grab rays from sample image
height, width = images.shape[1:3]
with torch.no_grad():
  ray_origin, ray_direction = get_rays(height, width, focal, testpose)

print('Ray Origin')
print(ray_origin.shape)
print(ray_origin[height // 2, width // 2, :])
print('')

print('Ray Direction')
print(ray_direction.shape)
print(ray_direction[height // 2, width // 2, :])
print('')
```

Ray Origin
torch.Size([100, 100, 3])
tensor([-1.9745, -1.8789,  2.9700], device='cuda:0')

Ray Direction
torch.Size([100, 100, 3])
tensor([ 0.4898,  0.4661, -0.7368], device='cuda:0')

## Arquitetura

## Amostragem Estratificada

Agora que temos essas retas, definidas como vetores origem e direção, podemos iniciar o processo de amostragem delas. Lembre-se de que o NeRF adota uma estratégia de amostragem grosseira a fina, começando com a abordagem de amostragem estratificada.

A abordagem de amostragem estratificada divide o raio em caixas com espaçamento uniforme e amostra aleatoriamente dentro de cada caixa. A configuração `perturb` determina se os pontos devem ser amostrados uniformemente de cada compartimento ou simplesmente usar o centro do compartimento como ponto. Na maioria dos casos, queremos manter `perturb = True`, pois isso incentivará a rede a aprender em um espaço amostrado continuamente. Pode ser útil desabilitar para depuração.

In [9]:

```
def sample_stratified(
  rays_o: torch.Tensor,
  rays_d: torch.Tensor,
  near: float,
  far: float,
  n_samples: int,
  perturb: Optional[bool] = True,
  inverse_depth: bool = False
) -> Tuple[torch.Tensor, torch.Tensor]:
  r"""
  Sample along ray from regularly-spaced bins.
  """

  # Grab samples for space integration along ray
  t_vals = torch.linspace(0., 1., n_samples, device=rays_o.device)
  if not inverse_depth:
    # Sample linearly between `near` and `far`
    z_vals = near * (1.-t_vals) + far * (t_vals)
  else:
    # Sample linearly in inverse depth (disparity)
    z_vals = 1./(1./near * (1.-t_vals) + 1./far * (t_vals))

  # Draw uniform samples from bins along ray
  if perturb:
    mids = .5 * (z_vals[1:] + z_vals[:-1])
    upper = torch.concat([mids, z_vals[-1:]], dim=-1)
    lower = torch.concat([z_vals[:1], mids], dim=-1)
    t_rand = torch.rand([n_samples], device=z_vals.device)
    z_vals = lower + (upper - lower) * t_rand
  z_vals = z_vals.expand(list(rays_o.shape[:-1]) + [n_samples])

  # Apply scale from `rays_d` and offset from `rays_o` to samples
  # pts: (width, height, n_samples, 3)
  pts = rays_o[..., None, :] + rays_d[..., None, :] * z_vals[..., :, None]
```

```
  return pts, z_vals
```

In [10]:

```
# Draw stratified samples from example
rays_o = ray_origin.view([-1, 3])
rays_d = ray_direction.view([-1, 3])
n_samples = 8
perturb = True
inverse_depth = False
with torch.no_grad():
  pts, z_vals = sample_stratified(rays_o, rays_d, near, far, n_samples,
                                  perturb=perturb,
inverse_depth=inverse_depth)

print('Input Points')
print(pts.shape)
print('')
print('Distances Along Ray')
print(z_vals.shape)
```

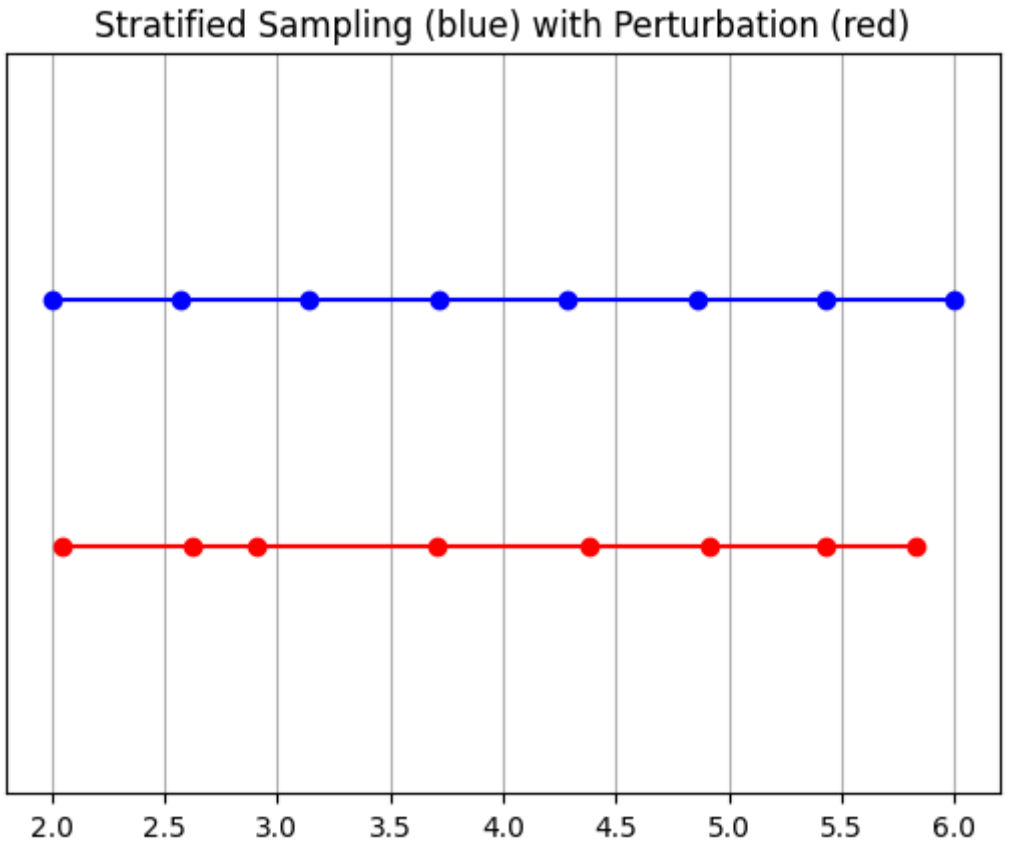
Input Points
torch.Size([10000, 8, 3])

Distances Along Ray
torch.Size([10000, 8])

Agora visualizamos esses pontos amostrados. Os pontos azuis imperturbados são os "centros" dos compartimentos. Os pontos vermelhos são uma amostragem de pontos perturbados. Observe como os pontos vermelhos estão ligeiramente deslocados dos pontos azuis acima deles, mas todos estão restritos entre `próximo` e `distante`.

In [11]:

```
y_vals = torch.zeros_like(z_vals)

_, z_vals_unperturbed = sample_stratified(rays_o, rays_d, near, far,
n_samples,
                                  perturb=False, inverse_depth=inverse_depth)
plt.plot(z_vals_unperturbed[0].cpu().numpy(), 1 + y_vals[0].cpu().numpy(),
'b-o')
plt.plot(z_vals[0].cpu().numpy(), y_vals[0].cpu().numpy(), 'r-o')
plt.ylim([-1, 2])
plt.title('Stratified Sampling (blue) with Perturbation (red)')
ax = plt.gca()
ax.axes.yaxis.set_visible(False)
plt.grid(True)
```




## Codificador posicional

Assim como os Transformers, os NeRFs fazem uso de codificadores posicionais. Nesse caso, é para mapear as entradas para um espaço de frequência mais alta para compensar o viés que as redes neurais têm para aprender funções de frequência mais baixa.

Aqui construímos um simples `torch.nn.Module` do nosso codificador posicional. A mesma implementação do codificador pode ser aplicada a amostras de entrada e direções de visualização. No entanto, escolhemos parâmetros diferentes para essas entradas. Usamos as configurações padrão do original.

In [12]:

```
class PositionalEncoder(nn.Module):
  r"""
  Sine-cosine positional encoder for input points.
  """
  def __init__(
    self,
    d_input: int,
    n_freqs: int,
```

```
    log_space: bool = False
  ):
    super().__init__()
    self.d_input = d_input
    self.n_freqs = n_freqs
    self.log_space = log_space
    self.d_output = d_input * (1 + 2 * self.n_freqs)
    self.embed_fns = [lambda x: x]

    # Define frequencies in either linear or log scale
    if self.log_space:
      freq_bands = 2.**torch.linspace(0., self.n_freqs - 1, self.n_freqs)
    else:
      freq_bands = torch.linspace(2.**0., 2.**(self.n_freqs - 1),
self.n_freqs)

    # Alternate sin and cos
    for freq in freq_bands:
      self.embed_fns.append(lambda x, freq=freq: torch.sin(x * freq))
      self.embed_fns.append(lambda x, freq=freq: torch.cos(x * freq))

  def forward(
    self,
    x
  ) -> torch.Tensor:
    r"""
    Apply positional encoding to input.
    """
    return torch.concat([fn(x) for fn in self.embed_fns], dim=-1)
```

In [13]:

```
# Create encoders for points and view directions
encoder = PositionalEncoder(3, 10)
viewdirs_encoder = PositionalEncoder(3, 4)

# Grab flattened points and view directions
pts_flattened = pts.reshape(-1, 3)
viewdirs = rays_d / torch.norm(rays_d, dim=-1, keepdim=True)
flattened_viewdirs = viewdirs[:, None, ...].expand(pts.shape).reshape((-1,
3))

# Encode inputs
encoded_points = encoder(pts_flattened)
encoded_viewdirs = viewdirs_encoder(flattened_viewdirs)
```

```
print('Encoded Points')
print(encoded_points.shape)
print(torch.min(encoded_points), torch.max(encoded_points),
torch.mean(encoded_points))
print('')

print(encoded_viewdirs.shape)
print('Encoded Viewdirs')
print(torch.min(encoded_viewdirs), torch.max(encoded_viewdirs),
torch.mean(encoded_viewdirs))
print('')
```

Encoded Points
torch.Size([80000, 63])
tensor(-2.7154, device='cuda:0') tensor(3.4242, device='cuda:0') tensor(0.0256, device='cuda:0')

torch.Size([80000, 27])
Encoded Viewdirs
tensor(-1., device='cuda:0') tensor(1., device='cuda:0') tensor(0.1056, device='cuda:0')

## Modelo NeRF

Aqui definimos o modelo NeRF, que consiste principalmente em uma `ModuleList` de camadas `Lineares`, separadas por funções de ativação não lineares e conexões residuais ocasionais. Este modelo apresenta uma entrada opcional para direções de visualização, que alterará a arquitetura do modelo fornecida na instanciação. Esta implementação é baseada na Seção 3 do artigo original "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis" e usa os mesmos padrões.

In [14]:

```
class NeRF(nn.Module):
  r"""
  Neural radiance fields module.
  """
  def __init__(
    self,
    d_input: int = 3,
    n_layers: int = 8,
    d_filter: int = 256,
    skip: Tuple[int] = (4,),
    d_viewdirs: Optional[int] = None
  ):
    super().__init__()
    self.d_input = d_input
```

```
    self.skip = skip
    self.act = nn.functional.relu
    self.d_viewdirs = d_viewdirs

    # Create model layers
    self.layers = nn.ModuleList(
      [nn.Linear(self.d_input, d_filter)] +
      [nn.Linear(d_filter + self.d_input, d_filter) if i in skip \
       else nn.Linear(d_filter, d_filter) for i in range(n_layers - 1)]
    )

    # Bottleneck layers
    if self.d_viewdirs is not None:
      # If using viewdirs, split alpha and RGB
      self.alpha_out = nn.Linear(d_filter, 1)
      self.rgb_filters = nn.Linear(d_filter, d_filter)
      self.branch = nn.Linear(d_filter + self.d_viewdirs, d_filter // 2)
      self.output = nn.Linear(d_filter // 2, 3)
    else:
      # If no viewdirs, use simpler output
      self.output = nn.Linear(d_filter, 4)

  def forward(
    self,
    x: torch.Tensor,
    viewdirs: Optional[torch.Tensor] = None
  ) -> torch.Tensor:
    r"""
    Forward pass with optional view direction.
    """

    # Cannot use viewdirs if instantiated with d_viewdirs = None
    if self.d_viewdirs is None and viewdirs is not None:
      raise ValueError('Cannot input x_direction if d_viewdirs was not
given.')

    # Apply forward pass up to bottleneck
    x_input = x
    for i, layer in enumerate(self.layers):
      x = self.act(layer(x))
      if i in self.skip:
        x = torch.cat([x, x_input], dim=-1)

    # Apply bottleneck
    if self.d_viewdirs is not None:
      # Split alpha from network output
      alpha = self.alpha_out(x)
```

```
      # Pass through bottleneck to get RGB
      x = self.rgb_filters(x)
      x = torch.concat([x, viewdirs], dim=-1)
      x = self.act(self.branch(x))
      x = self.output(x)

      # Concatenate alphas to output
      x = torch.concat([x, alpha], dim=-1)
    else:
      # Simple output
      x = self.output(x)
    return x
```

## Renderização de volume

A partir das saídas brutas do NeRF, ainda precisamos convertê-las em uma imagem. É aqui que aplicamos a integração de volume descrita nas Equações 1-3 na Seção 4 do artigo. Essencialmente, pegamos a soma ponderada de todas as amostras ao longo do raio de cada pixel para obter o valor estimado da cor naquele pixel. Cada amostra RGB é ponderada pelo seu valor alfa. Valores alfa mais altos indicam maior probabilidade de a área amostrada ser opaca, portanto, pontos mais ao longo do raio têm maior probabilidade de serem ocluídos. O produto cumulativo garante que esses pontos adicionais sejam amortecidos.

In [15]:

```
def cumprod_exclusive(
  tensor: torch.Tensor
) -> torch.Tensor:
  r"""
  (Courtesy of https://github.com/krrish94/nerf-pytorch)

  Mimick functionality of tf.math.cumprod(..., exclusive=True), as it isn't
available in PyTorch.

  Args:
  tensor (torch.Tensor): Tensor whose cumprod (cumulative product, see
`torch.cumprod`) along dim=-1
    is to be computed.
  Returns:
  cumprod (torch.Tensor): cumprod of Tensor along dim=-1, mimiciking the
functionality of
    tf.math.cumprod(..., exclusive=True) (see `tf.math.cumprod` for details).
  """
```

```
  # Compute regular cumprod first (this is equivalent to
`tf.math.cumprod(..., exclusive=False)`).
  cumprod = torch.cumprod(tensor, -1)
  # "Roll" the elements along dimension 'dim' by 1 element.
  cumprod = torch.roll(cumprod, 1, -1)
  # Replace the first element by "1" as this is what tf.cumprod(...,
exclusive=True) does.
  cumprod[..., 0] = 1.

  return cumprod

def raw2outputs(
  raw: torch.Tensor,
  z_vals: torch.Tensor,
  rays_d: torch.Tensor,
  raw_noise_std: float = 0.0,
  white_bkgd: bool = False
) -> Tuple[torch.Tensor, torch.Tensor, torch.Tensor, torch.Tensor]:
  r"""
  Convert the raw NeRF output into RGB and other maps.
  """

  # Difference between consecutive elements of `z_vals`. [n_rays,
n_samples]
  dists = z_vals[..., 1:] - z_vals[..., :-1]
  dists = torch.cat([dists, 1e10 * torch.ones_like(dists[..., :1])], dim=-1)

  # Multiply each distance by the norm of its corresponding direction ray
  # to convert to real world distance (accounts for non-unit directions).
  dists = dists * torch.norm(rays_d[..., None, :], dim=-1)

  # Add noise to model's predictions for density. Can be used to
  # regularize network during training (prevents floater artifacts).
  noise = 0.
  if raw_noise_std > 0.:
    noise = torch.randn(raw[..., 3].shape) * raw_noise_std

  # Predict density of each sample along each ray. Higher values imply
  # higher likelihood of being absorbed at this point. [n_rays, n_samples]
  alpha = 1.0 - torch.exp(-nn.functional.relu(raw[..., 3] + noise) * dists)

  # Compute weight for RGB of each sample along each ray. [n_rays,
n_samples]
  # The higher the alpha, the lower subsequent weights are driven.
  weights = alpha * cumprod_exclusive(1. - alpha + 1e-10)

  # Compute weighted RGB map.
```

```
  rgb = torch.sigmoid(raw[..., :3])  # [n_rays, n_samples, 3]
  rgb_map = torch.sum(weights[..., None] * rgb, dim=-2)  # [n_rays, 3]

  # Estimated depth map is predicted distance.
  depth_map = torch.sum(weights * z_vals, dim=-1)

  # Disparity map is inverse depth.
  disp_map = 1. / torch.max(1e-10 * torch.ones_like(depth_map),
                            depth_map / torch.sum(weights, -1))

  # Sum of weights along each ray. In [0, 1] up to numerical error.
  acc_map = torch.sum(weights, dim=-1)

  # To composite onto a white background, use the accumulated alpha map.
  if white_bkgd:
    rgb_map = rgb_map + (1. - acc_map[..., None])

  return rgb_map, depth_map, acc_map, weights
```

## Amostragem hierárquica de volume

O espaço 3D é de fato muito escasso com oclusões e por isso a maioria dos pontos não contribui muito para a imagem renderizada. Portanto, é mais benéfico sobre-amostrar regiões com alta probabilidade de contribuir para a integral. Aqui aplicamos pesos aprendidos e normalizados ao primeiro conjunto de amostras para criar um PDF através do raio e, em seguida, aplicamos amostragem de transformada inversa a este PDF para reunir um segundo conjunto de amostras.

In [16]:

```
def sample_pdf(
  bins: torch.Tensor,
  weights: torch.Tensor,
  n_samples: int,
  perturb: bool = False
) -> torch.Tensor:
  r"""
  Apply inverse transform sampling to a weighted set of points.
  """

  # Normalize weights to get PDF.
  pdf = (weights + 1e-5) / torch.sum(weights + 1e-5, -1, keepdims=True) #
[n_rays, weights.shape[-1]]

  # Convert PDF to CDF.
  cdf = torch.cumsum(pdf, dim=-1) # [n_rays, weights.shape[-1]]
```

```
  cdf = torch.concat([torch.zeros_like(cdf[..., :1]), cdf], dim=-1) #
[n_rays, weights.shape[-1] + 1]

  # Take sample positions to grab from CDF. Linear when perturb == 0.
  if not perturb:
    u = torch.linspace(0., 1., n_samples, device=cdf.device)
    u = u.expand(list(cdf.shape[:-1]) + [n_samples]) # [n_rays, n_samples]
  else:
    u = torch.rand(list(cdf.shape[:-1]) + [n_samples], device=cdf.device) #
[n_rays, n_samples]

  # Find indices along CDF where values in u would be placed.
  u = u.contiguous() # Returns contiguous tensor with same values.
  inds = torch.searchsorted(cdf, u, right=True) # [n_rays, n_samples]

  # Clamp indices that are out of bounds.
  below = torch.clamp(inds - 1, min=0)
  above = torch.clamp(inds, max=cdf.shape[-1] - 1)
  inds_g = torch.stack([below, above], dim=-1) # [n_rays, n_samples, 2]

  # Sample from cdf and the corresponding bin centers.
  matched_shape = list(inds_g.shape[:-1]) + [cdf.shape[-1]]
  cdf_g = torch.gather(cdf.unsqueeze(-2).expand(matched_shape), dim=-1,
                       index=inds_g)
  bins_g = torch.gather(bins.unsqueeze(-2).expand(matched_shape), dim=-1,
                        index=inds_g)

  # Convert samples to ray length.
  denom = (cdf_g[..., 1] - cdf_g[..., 0])
  denom = torch.where(denom < 1e-5, torch.ones_like(denom), denom)
  t = (u - cdf_g[..., 0]) / denom
  samples = bins_g[..., 0] + t * (bins_g[..., 1] - bins_g[..., 0])

  return samples # [n_rays, n_samples]
```

In [17]:

```
def sample_hierarchical(
  rays_o: torch.Tensor,
  rays_d: torch.Tensor,
  z_vals: torch.Tensor,
  weights: torch.Tensor,
  n_samples: int,
  perturb: bool = False
) -> Tuple[torch.Tensor, torch.Tensor, torch.Tensor]:
  r"""
  Apply hierarchical sampling to the rays.
```

```
  """

  # Draw samples from PDF using z_vals as bins and weights as
probabilities.
  z_vals_mid = .5 * (z_vals[..., 1:] + z_vals[..., :-1])
  new_z_samples = sample_pdf(z_vals_mid, weights[..., 1:-1], n_samples,
                             perturb=perturb)
  new_z_samples = new_z_samples.detach()

  # Resample points from ray based on PDF.
  z_vals_combined, _ = torch.sort(torch.cat([z_vals, new_z_samples], dim=-1),
dim=-1)
  pts = rays_o[..., None, :] + rays_d[..., None, :] * z_vals_combined[..., :,
None]  # [N_rays, N_samples + n_samples, 3]
  return pts, z_vals_combined, new_z_samples
```

## Passe para frente completo

É aqui que juntamos tudo para calcular uma única passagem direta em nosso modelo.

Devido a possíveis problemas de memória, a passagem direta é calculada em "blocos", que são então agregados em um único lote. A propagação do gradiente é feita após todo o lote ser processado, daí a distinção entre "pedaços" e "lotes". O chunking é especialmente importante para o ambiente Google Colab, que fornece recursos mais modestos do que os citados no artigo original.

In [18]:

```
def get_chunks(
  inputs: torch.Tensor,
  chunksize: int = 2**15
) -> List[torch.Tensor]:
  r"""
  Divide an input into chunks.
  """
  return [inputs[i:i + chunksize] for i in range(0, inputs.shape[0],
chunksize)]

def prepare_chunks(
  points: torch.Tensor,
  encoding_function: Callable[[torch.Tensor], torch.Tensor],
  chunksize: int = 2**15
) -> List[torch.Tensor]:
  r"""
  Encode and chunkify points to prepare for NeRF model.
  """
  points = points.reshape((-1, 3))
```

```
  points = encoding_function(points)
  points = get_chunks(points, chunksize=chunksize)
  return points

def prepare_viewdirs_chunks(
  points: torch.Tensor,
  rays_d: torch.Tensor,
  encoding_function: Callable[[torch.Tensor], torch.Tensor],
  chunksize: int = 2**15
) -> List[torch.Tensor]:
  r"""
  Encode and chunkify viewdirs to prepare for NeRF model.
  """
  # Prepare the viewdirs
  viewdirs = rays_d / torch.norm(rays_d, dim=-1, keepdim=True)
  viewdirs = viewdirs[:, None, ...].expand(points.shape).reshape((-1, 3))
  viewdirs = encoding_function(viewdirs)
  viewdirs = get_chunks(viewdirs, chunksize=chunksize)
  return viewdirs
```

In [19]:

```
def nerf_forward(
  rays_o: torch.Tensor,
  rays_d: torch.Tensor,
  near: float,
  far: float,
  encoding_fn: Callable[[torch.Tensor], torch.Tensor],
  coarse_model: nn.Module,
  kwargs_sample_stratified: dict = None,
  n_samples_hierarchical: int = 0,
  kwargs_sample_hierarchical: dict = None,
  fine_model = None,
  viewdirs_encoding_fn: Optional[Callable[[torch.Tensor], torch.Tensor]] =
None,
  chunksize: int = 2**15
) -> Tuple[torch.Tensor, torch.Tensor, torch.Tensor, dict]:
  r"""
  Compute forward pass through model(s).
  """

  # Set no kwargs if none are given.
  if kwargs_sample_stratified is None:
    kwargs_sample_stratified = {}
  if kwargs_sample_hierarchical is None:
    kwargs_sample_hierarchical = {}
```

```
  # Sample query points along each ray.
  query_points, z_vals = sample_stratified(
      rays_o, rays_d, near, far, **kwargs_sample_stratified)

  # Prepare batches.
  batches = prepare_chunks(query_points, encoding_fn, chunksize=chunksize)
  if viewdirs_encoding_fn is not None:
    batches_viewdirs = prepare_viewdirs_chunks(query_points, rays_d,
                                               viewdirs_encoding_fn,
                                               chunksize=chunksize)
  else:
    batches_viewdirs = [None] * len(batches)

  # Coarse model pass.
  # Split the encoded points into "chunks", run the model on all chunks,
and
  # concatenate the results (to avoid out-of-memory issues).
  predictions = []
  for batch, batch_viewdirs in zip(batches, batches_viewdirs):
    predictions.append(coarse_model(batch, viewdirs=batch_viewdirs))
  raw = torch.cat(predictions, dim=0)
  raw = raw.reshape(list(query_points.shape[:2]) + [raw.shape[-1]])

  # Perform differentiable volume rendering to re-synthesize the RGB image.
  rgb_map, depth_map, acc_map, weights = raw2outputs(raw, z_vals, rays_d)
  # rgb_map, depth_map, acc_map, weights = render_volume_density(raw,
rays_o, z_vals)
  outputs = {
      'z_vals_stratified': z_vals
  }

  # Fine model pass.
  if n_samples_hierarchical > 0:
    # Save previous outputs to return.
    rgb_map_0, depth_map_0, acc_map_0 = rgb_map, depth_map, acc_map

    # Apply hierarchical sampling for fine query points.
    query_points, z_vals_combined, z_hierarch = sample_hierarchical(
      rays_o, rays_d, z_vals, weights, n_samples_hierarchical,
      **kwargs_sample_hierarchical)

    # Prepare inputs as before.
    batches = prepare_chunks(query_points, encoding_fn, chunksize=chunksize)
    if viewdirs_encoding_fn is not None:
      batches_viewdirs = prepare_viewdirs_chunks(query_points, rays_d,
                                                 viewdirs_encoding_fn,
                                                 chunksize=chunksize)
```

```
    else:
      batches_viewdirs = [None] * len(batches)

    # Forward pass new samples through fine model.
    fine_model = fine_model if fine_model is not None else coarse_model
    predictions = []
    for batch, batch_viewdirs in zip(batches, batches_viewdirs):
      predictions.append(fine_model(batch, viewdirs=batch_viewdirs))
    raw = torch.cat(predictions, dim=0)
    raw = raw.reshape(list(query_points.shape[:2]) + [raw.shape[-1]])

    # Perform differentiable volume rendering to re-synthesize the RGB
image.
    rgb_map, depth_map, acc_map, weights = raw2outputs(raw, z_vals_combined,
rays_d)

    # Store outputs.
    outputs['z_vals_hierarchical'] = z_hierarch
    outputs['rgb_map_0'] = rgb_map_0
    outputs['depth_map_0'] = depth_map_0
    outputs['acc_map_0'] = acc_map_0

  # Store outputs.
  outputs['rgb_map'] = rgb_map
  outputs['depth_map'] = depth_map
  outputs['acc_map'] = acc_map
  outputs['weights'] = weights
  return outputs
```

## Treinamento

Finalmente, temos (quase) tudo que precisamos para treinar o modelo. Agora faremos algumas configurações para um procedimento de treinamento simples, criando hiperparâmetros e funções auxiliares e, em seguida, treinaremos nosso modelo.

## Hiperparâmetros

Todos os hiperparâmetros de treinamento são definidos aqui. Os padrões foram retirados do original, a menos que restrições computacionais os proíbam. Nesse caso, aplicamos padrões sensatos que estão dentro dos recursos fornecidos pelo Google Colab.

In [20]:

```
# Encoders
d_input = 3           # Number of input dimensions
n_freqs = 10          # Number of encoding functions for samples
log_space = True      # If set, frequencies scale in log space
use_viewdirs = True   # If set, use view direction as input
```

```
n_freqs_views = 4      # Number of encoding functions for views

# Stratified sampling
n_samples = 64         # Number of spatial samples per ray
perturb = True         # If set, applies noise to sample positions
inverse_depth = False  # If set, samples points linearly in inverse depth

# Model
d_filter = 128          # Dimensions of linear layer filters
n_layers = 2            # Number of layers in network bottleneck
skip = []               # Layers at which to apply input residual
use_fine_model = True   # If set, creates a fine model
d_filter_fine = 128     # Dimensions of linear layer filters of fine network
n_layers_fine = 6       # Number of layers in fine network bottleneck

# Hierarchical sampling
n_samples_hierarchical = 64   # Number of samples per ray
perturb_hierarchical = False  # If set, applies noise to sample positions

# Optimizer
lr = 5e-4  # Learning rate

# Training
n_iters = 10000
batch_size = 2**14          # Number of rays per gradient step (power of 2)
one_image_per_step = True   # One image per gradient step (disables
batching)
chunksize = 2**14           # Modify as needed to fit in GPU memory
center_crop = True          # Crop the center of image (one_image_per_)
center_crop_iters = 50      # Stop cropping center after this many epochs
display_rate = 25          # Display test output every X epochs

# Early Stopping
warmup_iters = 100          # Number of iterations during warmup phase
warmup_min_fitness = 10.0   # Min val PSNR to continue training at
warmup_iters
n_restarts = 10             # Number of times to restart if training stalls

# We bundle the kwargs for various functions to pass all at once.
kwargs_sample_stratified = {
    'n_samples': n_samples,
    'perturb': perturb,
    'inverse_depth': inverse_depth
}
kwargs_sample_hierarchical = {
    'perturb': perturb
}
```

## Classes e funções de treinamento

Aqui criamos algumas funções auxiliares para treinamento. O NeRF pode estar sujeito a mínimos locais, nos quais o treinamento irá rapidamente parar e produzir resultados em branco. `EarlyStopping` é usado para reiniciar o treinamento quando o aprendizado for necessário.

In [21]:

```
def plot_samples(
  z_vals: torch.Tensor,
  z_hierarch: Optional[torch.Tensor] = None,
  ax: Optional[np.ndarray] = None):
  r"""
  Plot stratified and (optional) hierarchical samples.
  """
  y_vals = 1 + np.zeros_like(z_vals)

  if ax is None:
    ax = plt.subplot()
  ax.plot(z_vals, y_vals, 'b-o')
  if z_hierarch is not None:
    y_hierarch = np.zeros_like(z_hierarch)
    ax.plot(z_hierarch, y_hierarch, 'r-o')
  ax.set_ylim([-1, 2])
  ax.set_title('Stratified  Samples (blue) and Hierarchical Samples (red)')
  ax.axes.yaxis.set_visible(False)
  ax.grid(True)
  return ax

def crop_center(
  img: torch.Tensor,
  frac: float = 0.5
) -> torch.Tensor:
  r"""
  Crop center square from image.
  """
  h_offset = round(img.shape[0] * (frac / 2))
  w_offset = round(img.shape[1] * (frac / 2))
  return img[h_offset:-h_offset, w_offset:-w_offset]

class EarlyStopping:
  r"""
  Early stopping helper based on fitness criterion.
  """
  def __init__(
```

```
    self,
    patience: int = 30,
    margin: float = 1e-4
  ):
    self.best_fitness = 0.0  # In our case PSNR
    self.best_iter = 0
    self.margin = margin
    self.patience = patience or float('inf')  # epochs to wait after fitness
stops improving to stop

  def __call__(
    self,
    iter: int,
    fitness: float
  ):
    r"""
    Check if criterion for stopping is met.
    """
    if (fitness - self.best_fitness) > self.margin:
      self.best_iter = iter
      self.best_fitness = fitness
    delta = iter - self.best_iter
    stop = delta >= self.patience  # stop training if patience exceeded
    return stop
```

In [22]:

```
def init_models():
  r"""
  Initialize models, encoders, and optimizer for NeRF training.
  """
  # Encoders
  encoder = PositionalEncoder(d_input, n_freqs, log_space=log_space)
  encode = lambda x: encoder(x)

  # View direction encoders
  if use_viewdirs:
    encoder_viewdirs = PositionalEncoder(d_input, n_freqs_views,
                                         log_space=log_space)
    encode_viewdirs = lambda x: encoder_viewdirs(x)
    d_viewdirs = encoder_viewdirs.d_output
  else:
    encode_viewdirs = None
    d_viewdirs = None

  # Models
```

```
  model = NeRF(encoder.d_output, n_layers=n_layers, d_filter=d_filter,
skip=skip,
               d_viewdirs=d_viewdirs)
  model.to(device)
  model_params = list(model.parameters())
  if use_fine_model:
    fine_model = NeRF(encoder.d_output, n_layers=n_layers, d_filter=d_filter,
skip=skip,
                      d_viewdirs=d_viewdirs)
    fine_model.to(device)
    model_params = model_params + list(fine_model.parameters())
  else:
    fine_model = None

  # Optimizer
  optimizer = torch.optim.Adam(model_params, lr=lr)

  # Early Stopping
  warmup_stopper = EarlyStopping(patience=50)

  return model, fine_model, encode, encode_viewdirs, optimizer,
warmup_stopper
```

## Ciclo de treinamento

Aqui começamos a treinar nosso modelo.

In [23]:

```
def train():
  r"""
  Launch training session for NeRF.
  """
  # Shuffle rays across all images.
  if not one_image_per_step:
    height, width = images.shape[1:3]
    all_rays = torch.stack([torch.stack(get_rays(height, width, focal, p), 0)
                        for p in poses[:n_training]], 0)
    rays_rgb = torch.cat([all_rays, images[:, None]], 1)
    rays_rgb = torch.permute(rays_rgb, [0, 2, 3, 1, 4])
    rays_rgb = rays_rgb.reshape([-1, 3, 3])
    rays_rgb = rays_rgb.type(torch.float32)
    rays_rgb = rays_rgb[torch.randperm(rays_rgb.shape[0])]
    i_batch = 0

  train_psnrs = []
  val_psnrs = []
```

```
  iternums = []
  for i in trange(n_iters):
    model.train()

    if one_image_per_step:
      # Randomly pick an image as the target.
      target_img_idx = np.random.randint(images.shape[0])
      target_img = images[target_img_idx].to(device)
      if center_crop and i < center_crop_iters:
        target_img = crop_center(target_img)
      height, width = target_img.shape[:2]
      target_pose = poses[target_img_idx].to(device)
      rays_o, rays_d = get_rays(height, width, focal, target_pose)
      rays_o = rays_o.reshape([-1, 3])
      rays_d = rays_d.reshape([-1, 3])
    else:
      # Random over all images.
      batch = rays_rgb[i_batch:i_batch + batch_size]
      batch = torch.transpose(batch, 0, 1)
      rays_o, rays_d, target_img = batch
      height, width = target_img.shape[:2]
      i_batch += batch_size
      # Shuffle after one epoch
      if i_batch >= rays_rgb.shape[0]:
          rays_rgb = rays_rgb[torch.randperm(rays_rgb.shape[0])]
          i_batch = 0
    target_img = target_img.reshape([-1, 3])

    # Run one iteration of TinyNeRF and get the rendered RGB image.
    outputs = nerf_forward(rays_o, rays_d,
                           near, far, encode, model,
                           kwargs_sample_stratified=kwargs_sample_stratified,
                           n_samples_hierarchical=n_samples_hierarchical,

kwargs_sample_hierarchical=kwargs_sample_hierarchical,
                           fine_model=fine_model,
                           viewdirs_encoding_fn=encode_viewdirs,
                           chunksize=chunksize)

    # Check for any numerical issues.
    for k, v in outputs.items():
      if torch.isnan(v).any():
        print(f"! [Numerical Alert] {k} contains NaN.")
      if torch.isinf(v).any():
        print(f"! [Numerical Alert] {k} contains Inf.")

    # Backprop!
```

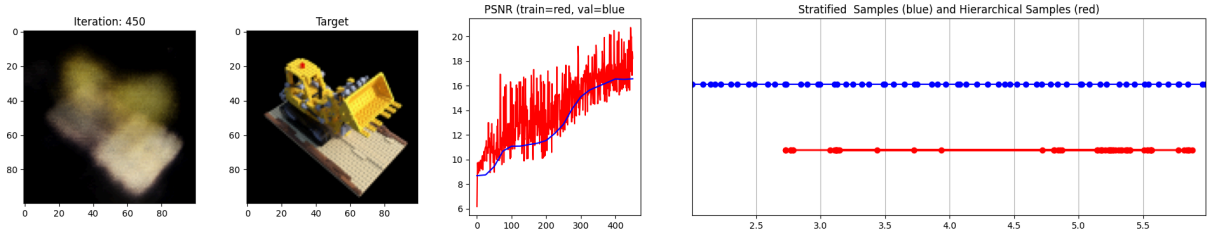
```
    rgb_predicted = outputs['rgb_map']
    loss = torch.nn.functional.mse_loss(rgb_predicted, target_img)
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()
    psnr = -10. * torch.log10(loss)
    train_psnrs.append(psnr.item())

    # Evaluate testimg at given display rate.
    if i % display_rate == 0:
      model.eval()
      height, width = testimg.shape[:2]
      rays_o, rays_d = get_rays(height, width, focal, testpose)
      rays_o = rays_o.reshape([-1, 3])
      rays_d = rays_d.reshape([-1, 3])
      outputs = nerf_forward(rays_o, rays_d,
                             near, far, encode, model,

kwargs_sample_stratified=kwargs_sample_stratified,
                             n_samples_hierarchical=n_samples_hierarchical,

kwargs_sample_hierarchical=kwargs_sample_hierarchical,
                             fine_model=fine_model,
                             viewdirs_encoding_fn=encode_viewdirs,
                             chunksize=chunksize)

      rgb_predicted = outputs['rgb_map']
      loss = torch.nn.functional.mse_loss(rgb_predicted, testimg.reshape(-1,
3))
      print("Loss:", loss.item())
      val_psnr = -10. * torch.log10(loss)
      val_psnrs.append(val_psnr.item())
      iternums.append(i)

      # Plot example outputs
      fig, ax = plt.subplots(1, 4, figsize=(24,4),
gridspec_kw={'width_ratios': [1, 1, 1, 3]})
      ax[0].imshow(rgb_predicted.reshape([height, width,
3]).detach().cpu().numpy())
      ax[0].set_title(f'Iteration: {i}')
      ax[1].imshow(testimg.detach().cpu().numpy())
      ax[1].set_title(f'Target')
      ax[2].plot(range(0, i + 1), train_psnrs, 'r')
      ax[2].plot(iternums, val_psnrs, 'b')
      ax[2].set_title('PSNR (train=red, val=blue')
      z_vals_strat = outputs['z_vals_stratified'].view((-1, n_samples))
```

```
      z_sample_strat = z_vals_strat[z_vals_strat.shape[0] //
2].detach().cpu().numpy()
      if 'z_vals_hierarchical' in outputs:
        z_vals_hierarch = outputs['z_vals_hierarchical'].view((-1,
n_samples_hierarchical))
        z_sample_hierarch = z_vals_hierarch[z_vals_hierarch.shape[0] //
2].detach().cpu().numpy()
      else:
        z_sample_hierarch = None
      _ = plot_samples(z_sample_strat, z_sample_hierarch, ax=ax[3])
      ax[3].margins(0)
      plt.show()

    # Check PSNR for issues and stop if any are found.
    if i == warmup_iters - 1:
      if val_psnr < warmup_min_fitness:
        print(f'Val PSNR {val_psnr} below warmup_min_fitness
{warmup_min_fitness}. Stopping...')
        return False, train_psnrs, val_psnrs
    elif i < warmup_iters:
      if warmup_stopper is not None and warmup_stopper(i, psnr):
        print(f'Train PSNR flatlined at {psnr} for {warmup_stopper.patience}
iters. Stopping...')
        return False, train_psnrs, val_psnrs

  return True, train_psnrs, val_psnrs
```

In [ ]:

```
# Run training session(s)
for _ in range(n_restarts):
  model, fine_model, encode, encode_viewdirs, optimizer, warmup_stopper =
init_models()
  success, train_psnrs, val_psnrs = train()
  if success and val_psnrs[-1] >= warmup_min_fitness:
    print('Training successful!')
    break

print('')
print(f'Done!')
```

```
4%|▌          | 450/10000 [03:19<1:09:20,  2.30it/s]
Loss: 0.02204357646405697
```




```
5%|▌         | 460/10000 [03:24<1:12:02,  2.21it/s]
```

In [ ]:

```
torch.save(model.state_dict(), 'nerf.pt')
torch.save(fine_model.state_dict(), 'nerf-fine.pt')
```

# Documentação dos experimentos 3D

## 1. Pesquisa e Preparação

Primeiro, familiarize-se com as especificações e recursos do Meta Quest 3. Ele possui um design mais fino, resolução mais alta, e desempenho aprimorado com um novo chipset Snapdragon, além de recursos como rastreamento dentro para fora e seis graus de liberdade (Unity Blog) (GameDev Academy).

## 2. Escolha de Ferramentas e Frameworks

Embora a maioria das aplicações para VR utilize motores de jogo como Unity e Unreal Engine, você pode integrar Python no processo de desenvolvimento, especialmente para prototipagem rápida e integração com outras ferramentas de análise e simulação.

### Unity com Python

Unity é uma das plataformas mais recomendadas para desenvolvimento de VR devido à sua versatilidade e ampla adoção na indústria. Você pode usar Unity para criar a parte gráfica e interativa da sua aplicação e usar Python para lógica adicional ou para comunicação com servidores e processamento de dados. Para isso, você pode usar o pacote Unity's Python for Unity (Unity Blog).

### Ferramentas Python Nativas para VR

Existem bibliotecas em Python que você pode utilizar diretamente para criar aplicações VR:

- **Pyglet e PyOpenGL**: Permitem criar janelas e renderizar gráficos VR. Embora não sejam tão sofisticados quanto Unity ou Unreal, são ótimos para prototipagem rápida (GameDev Academy).
- **Vizard**: Uma plataforma mais específica para VR que oferece suporte direto para Python e é ideal para experiências interativas em VR (GameDev Academy).

## 3. Desenvolvimento de Aplicações

Comece configurando seu ambiente de desenvolvimento:

1. **Configurar o Meta Quest 3 para Modo Desenvolvedor**: Use o aplicativo Meta Quest no seu telefone para ativar as configurações de desenvolvedor no seu headset.

2. **Instalar Unity e Pacotes Necessários**: Baixe a versão mais recente do Unity (recomendada a versão LTS de 2022 ou posterior). Adicione o pacote Meta OpenXR através do Unity Package Manager para suporte específico ao Quest 3 (Unity Blog).
3. **Criar um Projeto em Unity**: Utilize os templates e amostras de XR disponíveis para começar rapidamente. Esses templates incluem suporte para passthrough, detecção de planos, e rastreamento de dispositivos (Unity Blog).

### 4. Integração e Expansão com Python

Para integrar Python ao seu projeto Unity:

- **Python for Unity**: Este pacote permite a execução de scripts Python dentro do Unity. Útil para automatização de tarefas, integração com APIs, e processamento de dados em tempo real.
- **Comunicação com Servidores**: Use bibliotecas Python como Flask ou FastAPI para criar servidores backend que se comunicam com sua aplicação VR.

### 5. Teste e Iteração

Teste sua aplicação constantemente para garantir que funcione corretamente no hardware real. Utilize as ferramentas de debugging e perfomance do Unity para otimizar sua aplicação.

### Recursos Adicionais

- **Zenva Academy**: Oferece cursos detalhados sobre desenvolvimento de jogos e aplicações VR, incluindo um Mini-Degree em Desenvolvimento de Realidade Virtual (GameDev Academy).
- **Documentação Unity e OpenXR**: Consulte regularmente a documentação oficial para atualizações e melhores práticas (Unity Blog).

## Abordagem das NERFS:

## Documentação do Notebook: NeRF From Nothing

Este notebook demonstra o uso de Neural Radiance Fields (NeRFs) para a geração de cenas 3D a partir de imagens 2D. O NeRF é uma técnica poderosa que permite a reconstrução de cenas tridimensionais utilizando redes neurais profundas. Vamos revisar as principais seções do notebook, explicando seu propósito e funcionalidade.

## 1. Configuração Inicial

**Objetivo:** Preparar o ambiente com as bibliotecas necessárias para a execução do código.

**Descrição:** Esta seção instala as bibliotecas e pacotes necessários, como PyTorch, NumPy e outras dependências relevantes.

## 2. Importação de Bibliotecas

**Objetivo:** Importar todas as bibliotecas essenciais para a execução do notebook.

**Descrição:** As bibliotecas importadas incluem PyTorch para manipulação de tensores e criação de modelos, Matplotlib para visualização, e outras funções utilitárias necessárias para o processamento de dados e treinamento do modelo.

## 3. Definição dos Dados e Parâmetros

**Objetivo:** Definir os parâmetros e preparar os dados que serão usados para treinar o modelo NeRF.

**Descrição:** Nesta seção, os dados de entrada são preparados. Geralmente, os dados consistem em um conjunto de imagens de uma cena junto com suas respectivas poses de câmera. Os parâmetros do modelo, como o número de amostras de raios e o número de iterações de treinamento, são definidos aqui.

## 4. Implementação do Modelo NeRF

**Objetivo:** Definir a arquitetura do modelo NeRF.

**Descrição:** O modelo NeRF é implementado utilizando redes neurais profundas. A rede toma coordenadas 3D e direções de visão como entrada e prediz a densidade volumétrica e a cor do ponto 3D. A implementação inclui as camadas da rede, funções de ativação e o forward pass.

## 5. Funções de Renderização

**Objetivo:** Implementar funções para renderizar imagens a partir do modelo NeRF treinado.

**Descrição:** Esta seção define as funções necessárias para a renderização de imagens. Inclui a amostragem de pontos ao longo dos raios de visão, a integração das saídas do modelo ao longo dos raios para produzir pixels renderizados, e a função de renderização que combina tudo isso.

## 6. Função de Treinamento

**Objetivo:** Treinar o modelo NeRF nos dados de entrada.

**Descrição:** A função de treinamento executa o loop de treinamento para otimizar os pesos do modelo NeRF. Ela inclui a computação das perdas, a retropropagação do erro e a atualização dos pesos do modelo. Também salva checkpoints durante o treinamento para facilitar a recuperação e análise posterior.

## 7. Visualização dos Resultados

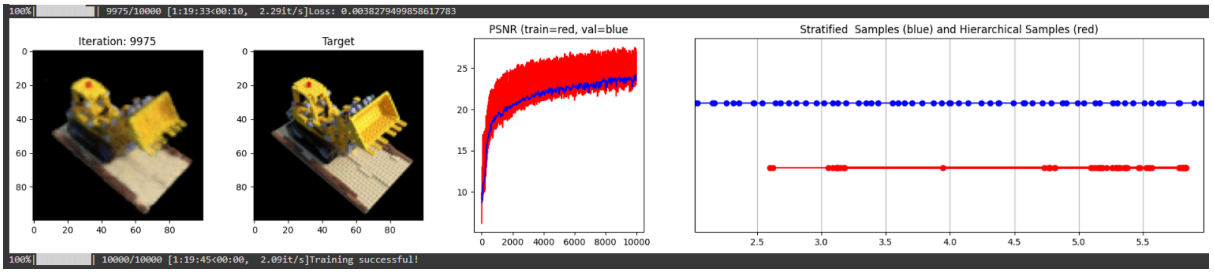
**Objetivo:** Visualizar os resultados do modelo NeRF treinado.

**Descrição:** Após o treinamento, esta seção renderiza imagens a partir do modelo treinado e as compara com as imagens de referência. Ferramentas de visualização como Matplotlib são usadas para exibir as imagens geradas.

## 8. Avaliação e Ajustes Finais

**Objetivo:** Avaliar o desempenho do modelo e ajustar os parâmetros se necessário.

**Descrição:** Esta seção avalia a qualidade das imagens geradas pelo modelo e pode incluir métricas quantitativas de desempenho. Baseado nos resultados, ajustes finais podem ser feitos nos parâmetros do modelo ou no processo de treinamento.




## Conclusão

O notebook "NeRF From Nothing" oferece um guia abrangente para a implementação e treinamento de modelos NeRF, desde a preparação dos dados até a visualização dos resultados. Utilizando técnicas avançadas de aprendizado profundo, ele demonstra como reconstruir cenas 3D de alta qualidade a partir de imagens 2D, destacando a potencialidade dos NeRFs em aplicações de visão computacional e realidade aumentada/virtual.

## Conclusão Geral:

Ao desenvolver aplicações que geram cenários 3D para o Meta Quest 3, é essencial escolher as ferramentas e técnicas adequadas para garantir eficiência e qualidade. Neste contexto, optar por técnicas de geração de imagens panorâmicas em 360 graus com base em modelos de difusão pode oferecer várias vantagens significativas em comparação com o uso de NERFs (Neural Radiance Fields) e SAGnets (Scene-aware Generative Networks).

## Por que não usar NERFs e SAGnets para gerar cenários 3D

1. **Complexidade e Tempo de Treinamento:**
   - **NERFs**: Embora sejam eficazes na reconstrução de cenas 3D a partir de várias imagens, os NERFs requerem uma quantidade significativa de dados e tempo de processamento para treinar modelos complexos. Esse processo pode ser inviável para aplicações em tempo real, como a geração dinâmica de cenários VR em um dispositivo como o Meta Quest 3.
   - **SAGnets**: As SAGnets também enfrentam desafios semelhantes, com a necessidade de modelos complexos para entender e gerar cenários realistas. A complexidade adicional de entender a semântica e a geometria de uma cena pode levar a tempos de processamento mais longos e maior uso de recursos.
2. **Integração e Facilidade de Uso:**
   - **Modelos de Difusão e Geração de Imagens em 360 Graus**: Utilizando modelos como o **SD-T2I-360PanoImage** e técnicas descritas em projetos como **DreamScene360**, é possível gerar imagens panorâmicas de alta qualidade de forma rápida e eficiente. Esses modelos são mais simples de integrar em pipelines existentes e podem gerar cenas 360 a partir de descrições textuais ou imagens únicas em menos tempo.
   - A abordagem de **blending circular** e **splatting gaussiano** para gerar cenas 3D a partir de imagens 360 oferece uma solução prática que mantém a continuidade e a qualidade visual sem exigir o processamento pesado dos NERFs e SAGnets.
3. **Qualidade e Consistência Visual:**
   - **Modelos de Difusão**: As técnicas de difusão para imagens 360 conseguem manter uma alta qualidade visual e coerência geométrica, mesmo ao gerar cenas complexas. Essas técnicas permitem criar experiências imersivas com cobertura completa de 360 graus, evitando descontinuidades visuais que podem ocorrer com outras abordagens.
   - A capacidade de ajustar e refinar as entradas textuais iterativamente, como implementado no projeto **DreamScene360**, proporciona uma flexibilidade que facilita a criação de cenas personalizadas e consistentes.

## Implementação Recomendada

Para desenvolver uma aplicação onde o usuário interage com o Meta Quest 3 para gerar cenários 360, siga estas etapas:

1. **Configuração do Ambiente:**
   - Configure o Meta Quest 3 no modo desenvolvedor e instale o Unity com os pacotes necessários (Meta OpenXR).
   - Utilize ferramentas como o `Python for Unity` para integrar scripts Python, facilitando a comunicação com servidores backend e processamento de dados.
2. **Escolha de Modelos e Técnicas:**
   - Utilize repositórios como o SD-T2I-360PanoImage para implementar a geração de imagens em 360 graus a partir de descrições textuais ou imagens.
   - Explore o uso de técnicas descritas no DreamScene360 para criar cenários 3D imersivos a partir de panoramas 360.
3. **Desenvolvimento e Integração:**
   - Desenvolva e teste sua aplicação VR no Unity, utilizando os templates e ferramentas de XR para garantir uma experiência de usuário otimizada.
   - Integre os modelos de difusão e pipelines de geração de imagens 360 para permitir que o usuário crie e explore cenários personalizados em tempo real.

## Repositórios no GitHub para geração de Imagem 360:

1. **SD-T2I-360PanoImage**
   - Este repositório utiliza modelos de difusão para a geração de imagens panorâmicas em 360 graus. Ele suporta tanto a geração de imagens a partir de texto (Text-to-360Panorama) quanto a conversão de uma única imagem em uma panorâmica 360 (Single-Image-to-360Panorama).

**Instalação**:

```
git clone https://github.com/ArcherFMY/SD-T2I-360PanoImage.git
cd SD-T2I-360PanoImage
pip install -r requirements.txt
```

- **Uso**:
  - Text-to-360Panorama:

```
import torch
from diffusers.utils import load_image
from img2panoimg import Image2360PanoramaImagePipeline

image = load_image("./data/i2p-image.jpg").resize((512, 512))
mask = load_image("./data/i2p-mask.jpg")
prompt = 'The office room'
input = {'prompt': prompt, 'image': image, 'mask': mask, 'upscale': False}
model_id = 'models'
img2panoimg = Image2360PanoramaImagePipeline(model_id, torch_dtype=torch.float16
output = img2panoimg(input)
output.save('result.png')
```

  - Single-Image-to-360Panorama:

```
import torch
from diffusers.utils import load_image
from img2panoimg import Image2360PanoramaImagePipeline

image = load_image("./data/i2p-image.jpg").resize((512, 512))
mask = load_image("./data/i2p-mask.jpg")
prompt = 'The office room'
input = {'prompt': prompt, 'image': image, 'mask': mask, 'upscale': False}
model_id = 'models'
img2panoimg = Image2360PanoramaImagePipeline(model_id, torch_dtype=torch.float16
output = img2panoimg(input)
output.save('result.png')
```

- Mais detalhes no repositório no GitHub.

2. **AOG-NET-360**
    - AOG-Net é uma rede generativa para criação de imagens em 360 graus utilizando orientação de texto. Esta técnica permite a geração de imagens panorâmicas 360 a partir de descrições detalhadas.
    - Mais informações podem ser encontradas aqui.
3. **DreamScene360**
    - Este projeto apresenta um pipeline de geração de cenas 3D a partir de texto, utilizando um modelo de difusão 2D e técnicas de splatting Gaussiano panorâmico para criar imagens panorâmicas de alta qualidade em 360 graus.
    - O pipeline envolve a geração de uma imagem panorâmica 2D, seguida da criação de um campo geométrico 3D para permitir a exploração em tempo real.
    - Mais detalhes podem ser encontrados aqui.

## Técnicas e Melhorias na Geração de Imagens em 360 Graus

1. **Modelos de Difusão e Blending Circular**
    - A principal técnica utilizada envolve modelos de difusão com uma estratégia de blending circular para manter a continuidade geométrica nas bordas da imagem panorâmica. Isso ajuda a evitar descontinuidades visuais onde as bordas da imagem se encontram.
2. **Splatting Gaussiano Panorâmico**
    - Essa técnica envolve a projeção de pontos 3D (Gaussians) a partir de uma imagem 2D para criar uma representação panorâmica completa. Ela permite uma renderização rápida e interativa de cenas 3D a partir de entradas textuais.
3. **Refinamento e Regularização**
    - Utilização de refinamento automático de prompts e regularização geométrica e semântica para otimizar a geração de Gaussians, preenchendo lacunas e melhorando a consistência da cena gerada.

# APÊNDICE 4

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("gate") de aprovação: 12 de jun. de 2024

## Participantes da Entrega [matriculados em Residência em IA]:

RODRIGO MENDES DE CARVALHO

## Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

- Foi feito um colab para gerar uma plataforma online que gera texto através de áudio usando o pydub e o pyngrok, dentro dessa plataforma o usuário fala ao microfone e é gerado um prompt que é passado para o modelo generativo SD-T2I-360PanoImage, logo após a pessoa pode baixar a imagem.
  Criar imagem 360
  https://github.com/Rod15greo/Residencia_IA_Rodrigo_Mendes/blob/main/IMG-20240612-WA0197.jpg

- Foi feito um código que gera um visualizador de imagens 360 para simular o ponto de vista do VR
  https://github.com/Rod15greo/Residencia_IA_Rodrigo_Mendes/blob/main/EquiView360.py

- Foi feito os primeiros módulos do curso Create with VR da Unity Learn
  https://learn.unity.com/course/create-with-vr

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Estruturar a infraestrutura do Meta Quest 3, Unity e GCP
- Integrar a estrutura nos devidos
- Explorar o https://github.com/ashawkey/stable-dreamfusion

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

**ALDO ANDRÉ DÍAZ SALAZAR:** Go!

# Notebook Rodrigo Jupyter: Plataforma de Geração de Imagens 360

## Clonar Repositório do Modelo de Geração:

In [ ]:

```
!git clone https://github.com/ArcherFMY/SD-T2I-360PanoImage.git
%cd SD-T2I-360PanoImage
!pip install -r requirements.txt
```

## Clonar Repositório com pesos do Modelo:

In [ ]:

```
!git lfs install

Git LFS initialized.
```

In [ ]:

```
!git clone https://huggingface.co/archerfmy0831/sd-t2i-360panoimage

Cloning into 'sd-t2i-360panoimage'...
remote: Enumerating objects: 79, done.
remote: Total 79 (delta 0), reused 0 (delta 0), pack-reused 79 (from 1)
Unpacking objects: 100% (79/79), 521.87 KiB | 3.48 MiB/s, done.
Filtering content: 100% (14/14), 13.85 GiB | 34.22 MiB/s, done.
```

In [ ]:

```
!pip install --upgrade torchvision
```

## Teste para Gerar Imagens 360:

In [ ]:

```
!python3 /content/SD-T2I-360PanoImage/demo_t2p.py
```

## Gerar plataforma de áudio para texto:

In [ ]:

```
!pip install SpeechRecognition pydub flask-ngrok
!apt-get install ffmpeg
!pip install pyngrok
```

```
Collecting pyngrok
  Downloading pyngrok-7.1.6-py3-none-any.whl (22 kB)
Requirement already satisfied: PyYAML>=5.1 in
/usr/local/lib/python3.10/dist-packages (from pyngrok) (6.0.1)
Installing collected packages: pyngrok
Successfully installed pyngrok-7.1.6
```

## Gerar credenciais no ngrock:

In [ ]:

```
from pyngrok import ngrok

# Substitua 'YOUR_AUTHTOKEN' pelo token de autenticação copiado do ngrok
authtoken = "2hmbXLatToQrh5u18W2OuGTBC8o_6s8Eum6WDrWYNVjCJfyVN"
ngrok.set_auth_token(authtoken)

# Inicie o túnel ngrok
public_url = ngrok.connect(5000)
print(f" * ngrok tunnel \"{public_url}\" -> \"http://127.0.0.1:5000\"")
```

```
* ngrok tunnel "NgrokTunnel: "https://a52b-34-83-207-54.ngrok-free.app" ->
"http://localhost:5000"" -> "http://127.0.0.1:5000"
```

## Gerar Plataforma:

In [ ]:

```
import os
from flask import Flask, request, jsonify
import speech_recognition as sr
from pydub import AudioSegment

app = Flask(__name__)

@app.route('/')
def index():
    return """


        Audio Recorder
```

```
        Gravador de Áudio


        Transcrição



    """

@app.route('/transcribe', methods=['POST'])
def transcribe():
    audio_file = request.files['audio']
    audio_path = "temp.wav"
    audio_file.save(audio_path)

    # Converta o áudio para um formato PCM WAV que o SpeechRecognition
possa ler
    audio = AudioSegment.from_file(audio_path)
    audio.export("converted.wav", format="wav")

    recognizer = sr.Recognizer()

    try:
        with sr.AudioFile("converted.wav") as source:
            audio_data = recognizer.record(source)
            text = recognizer.recognize_google(audio_data, language='pt-BR')
            return jsonify({"transcription": text})
    except Exception as e:
        return jsonify({"transcription": f"Erro ao processar o áudio: {e}"})

if __name__ == '__main__':
    app.run()


* Serving Flask app '__main__'
 * Debug mode: off

INFO:werkzeug:WARNING: This is a development server. Do not use it in a
production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug:127.0.0.1 - - [12/Jun/2024 19:05:44] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [12/Jun/2024 19:05:44] "GET /favicon.ico
HTTP/1.1" 404 -
```

```
INFO:werkzeug:127.0.0.1 - - [12/Jun/2024 19:05:53] "POST /transcribe
HTTP/1.1" 200 -
```

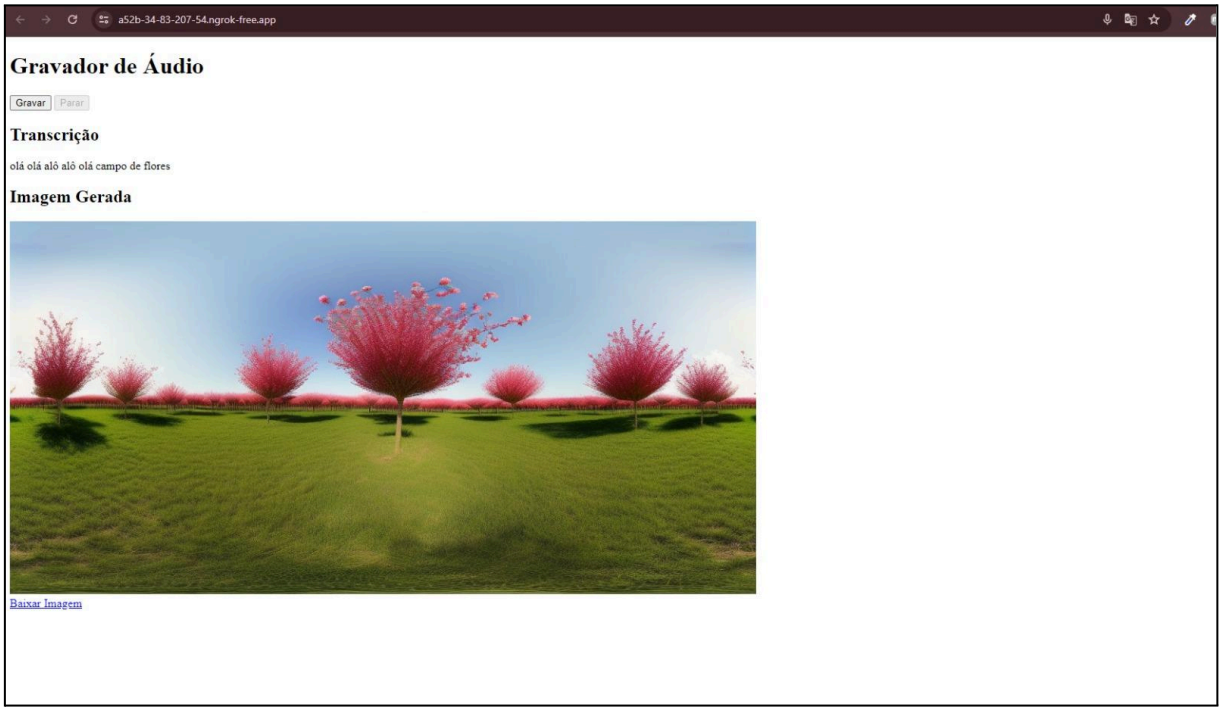
### Integrar as duas Funcionalidades para gerar a plataforma final:

In [ ]:

```
!python /content/SD-T2I-360PanoImage/test_final.py
```

## Imagem da versão inicial da plataforma:

Na imagem abaixo é possível notar a integração de todas funcionalidades pensadas em uma interface web inicial, na qual o usuário  poderia mandar um áudio sobre qual conteúdo ele gostaria de ser criado e logo em sequência entrava em ação o modelo de geração de imagens 360, disponibilizando a imagem criada para ser baixada.



## Código python para visualizar imagens 360 para simular o ponto de vista do VR:

```
from OpenGL.GL import *
from OpenGL.GLU import *
from PyQt5 import QtWidgets, QtCore, QtGui
from PyQt5.QtOpenGL import QGLWidget
from PIL import Image

class GLWidget(QGLWidget):
    def __init__(self, parent):
        super().__init__(parent)
        self.image =
Image.open("C:/Users/greee/Downloads/EquiView360-main/EquiView360-main/static/exampl
e.jpg")
        self.image_width, self.image_height = self.image.size
        self.yaw = 0
        self.pitch = 0
        self.prev_dx = 0
        self.prev_dy = 0
        self.fov = 90
        self.moving = False

    def initializeGL(self):
        glEnable(GL_TEXTURE_2D)
        self.texture = glGenTextures(1)
        glBindTexture(GL_TEXTURE_2D, self.texture)
        glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, self.image_width, self.image_height, 0,
GL_RGB, GL_UNSIGNED_BYTE, self.image.tobytes())
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)
        self.sphere = gluNewQuadric()
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        gluPerspective(90, self.width()/self.height(), 0.1, 1000)
        glMatrixMode(GL_MODELVIEW)
        glLoadIdentity()

    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
```

```
        glPushMatrix()
        glRotatef(self.pitch, 1, 0, 0)
        glRotatef(self.yaw, 0, 1, 0)
        glRotatef(90, 1, 0, 0)
        glRotatef(-90, 0, 0, 1)
        gluQuadricTexture(self.sphere, True)
        gluSphere(self.sphere, 1, 100, 100)
        glPopMatrix()

    def resizeGL(self, width, height):
        glViewport(0, 0, width, height)
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        gluPerspective(self.fov, self.width()/self.height(), 0.1, 1000)

    def mousePressEvent(self, event):
        if event.button() == QtCore.Qt.LeftButton:
            self.mouse_x, self.mouse_y = event.pos().x(), event.pos().y()
            self.moving = True

    def mouseReleaseEvent(self, event):
        if event.button() == QtCore.Qt.LeftButton:
            self.moving = False

    def mouseMoveEvent(self, event):
        if self.moving:
            dx = event.pos().x() - self.mouse_x
            dy = event.pos().y() - self.mouse_y
            dx *= 0.1
            dy *= 0.1
            self.yaw -= dx
            self.pitch -= dy
            self.pitch = min(max(self.pitch, -90), 90)
            self.mouse_x, self.mouse_y = event.pos().x(), event.pos().y()
            self.update()

    def wheelEvent(self, event):
        delta = event.angleDelta().y()
        self.fov -= delta * 0.1
        self.fov = max(30, min(self.fov, 90))
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        gluPerspective(self.fov, self.width()/self.height(), 0.1, 1000)
```

```
        self.update()

class MainWindow(QtWidgets.QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Equirectangular 360° Viewer")
        self.setWindowIcon(QtGui.QIcon("icon.png"))
        self.gl_widget = GLWidget(self)
        self.setCentralWidget(self.gl_widget)

if __name__ == '__main__':
    app = QtWidgets.QApplication([])
    window = MainWindow()
    window.setGeometry(0, 0, 1080, 720)
    window.show()
    app.exec_()
```

# APÊNDICE 5

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("gate") de aprovação: 19 de jun. de 2024

## Participantes da Entrega [matriculados em Residência em IA]:

RODRIGO MENDES DE CARVALHO

## Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

- Foi feito um novo site que agora consegue rodar o visualizador, foi feito as integrações em java-script para que funcionasse, agora o usuário pode gerar as imagens panorâmicas e depois visualizar, um vídeo exemplo e o link para o site e o github do site se encontra abaixo

Cópia de Criar imagem 360 - Colab - Google Chrome 2024-06-19 16-33-39.mp4
https://a2be-35-204-208-138.ngrok-free.app
https://github.com/Rod15greo/viewer-360-harpyai
Cópia de Criar imagem 360

- Foi feito um documento para estruturar como vai funcionar a arquitetura da plataforma dentro do GCP, no documento abaixo podemos visualizar a sua estrutura
  Documento de Infraestrutura para Aplicativo Meta Quest 3 com Unity e GCP
- Foi feito um estudo sobre o https://github.com/ashawkey/stable-dreamfusion
  Documentação Detalhada do Projeto Stable DreamFusion
- Foi feito um documento explicando os erros que deram ao tentar subir uma máquina virtual no GCP para a plataforma e o erro que deu ao tentar colocar localmente
  Documento de Explicação sobre Falhas ao Tentar Subir uma VM para Rodar Stable DreamF…

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Terminar o curso Create with VR da Unity Learn
  https://learn.unity.com/course/create-with-vr
- Implementar a plataforma no VR
- Implementar DreamFusion na plataforma web

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

**ALDO ANDRÉ DÍAZ SALAZAR:** Em análise!

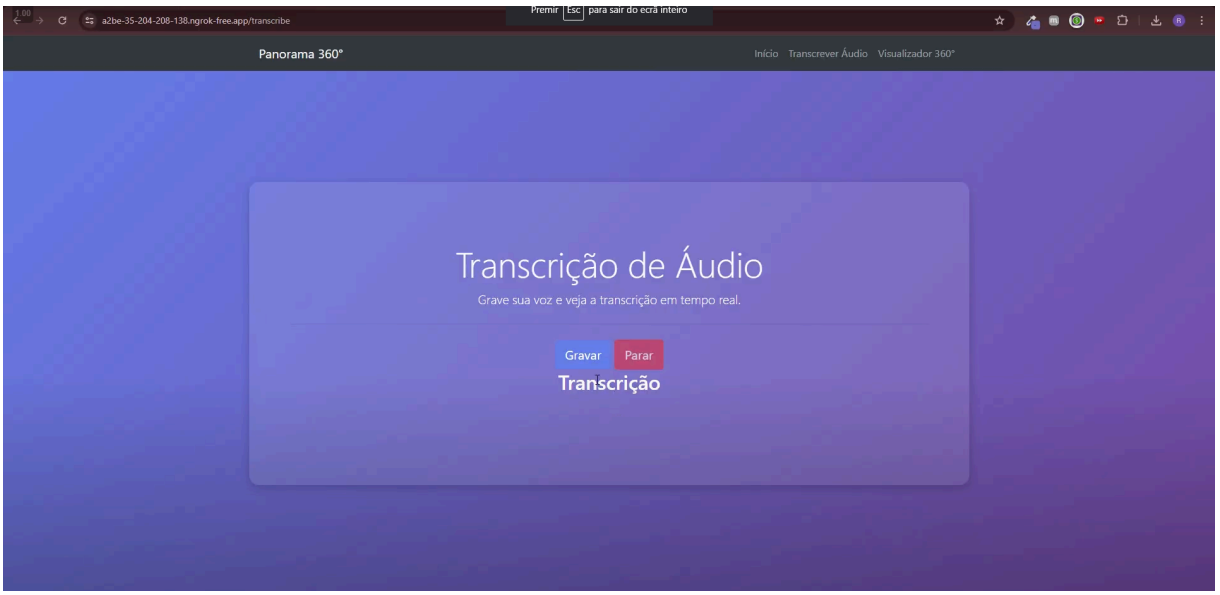
# Plataforma para geração de imagens 360:

## Tela de Boas-Vindas:

A tela de boas-vindas é a primeira interface que o usuário encontra ao acessar a plataforma. Ela apresenta uma breve introdução sobre a funcionalidade da plataforma, incentivando o usuário a experimentar suas imagens panorâmicas. O botão "Continuar" leva o usuário para a próxima etapa, onde ele pode iniciar a transcrição de áudio ou a visualização de imagens 360°.

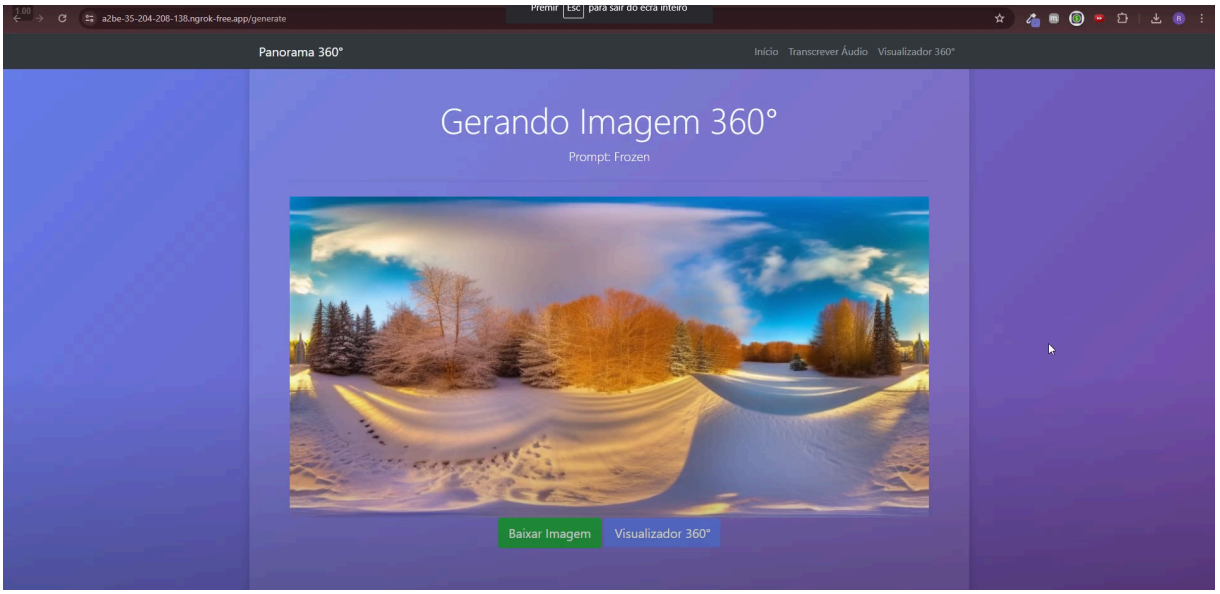

## Janela de Transcrição de Áudio

Nesta janela, o usuário pode gravar sua voz e ver a transcrição em tempo real. Os botões "Gravar" e "Parar" permitem o controle da gravação de áudio. A transcrição de áudio facilita a criação de prompts para a geração de imagens panorâmicas baseadas na descrição verbal do usuário.




## Gerando Imagem 360°

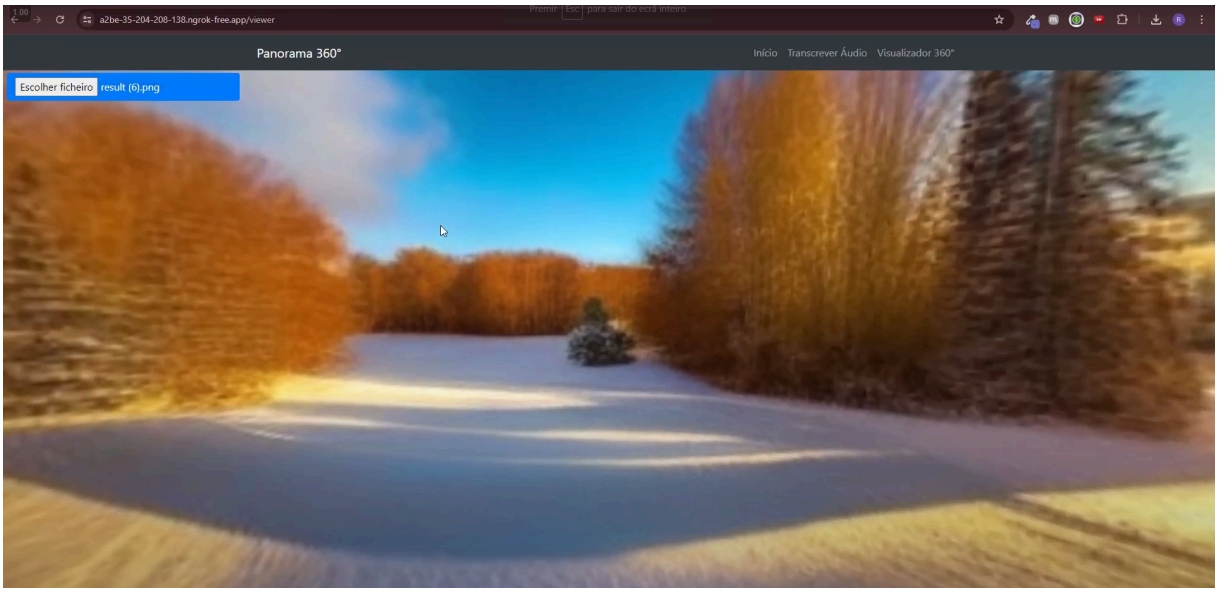
Esta tela é exibida durante o processo de geração de imagens 360° a partir do prompt de áudio fornecido. O sistema processa o comando e cria uma imagem panorâmica. O usuário deve aguardar enquanto a imagem é gerada, sendo informado do status atual do pr




## Janela de Resultado

Após a geração da imagem 360°, esta janela apresenta a imagem resultante. O usuário tem a opção de "Baixar Imagem" para salvar a imagem no seu dispositivo ou visualizar a imagem em 360° diretamente na plataforma, utilizando a funcionalidade de visualização

interativa. Esta interface finaliza o processo, permitindo ao usuário revisar e utilizar a imagem criada.




# Documento de Infraestrutura para Aplicativo Meta Quest 3 com Unity e GCP

## 1. Visão Geral do Projeto

Este documento descreve a infraestrutura necessária para desenvolver um aplicativo de realidade virtual no Meta Quest 3 usando Unity. O aplicativo se conectará a uma máquina virtual no Google Cloud Platform (GCP) para receber um áudio, convertê-lo em um prompt, gerar uma imagem panorâmica 360° com o modelo SD-T2I-360PanoImage, e utilizar essa imagem como fundo de background no aplicativo.

## 2. Componentes do Sistema

1. **Meta Quest 3**
   - Dispositivo de realidade virtual onde o aplicativo Unity será executado.
2. **Unity**
   - Ambiente de desenvolvimento integrado (IDE) utilizado para criar o aplicativo VR.
3. **Google Cloud Platform (GCP)**
   - Infraestrutura em nuvem que hospedará a máquina virtual para processamento de áudio e geração de imagens.

4. **SD-T2I-360PanoImage**
    - Modelo de inteligência artificial utilizado para gerar imagens panorâmicas 360° a partir de prompts de texto.

## 3. Configuração do Meta Quest 3 e Unity

1. **Instalação do Unity**
    - Baixe e instale a versão mais recente do Unity.
    - Instale o pacote de suporte para VR e o SDK do Meta Quest.
2. **Configuração do Projeto no Unity**
    - Crie um novo projeto 3D.
    - Configure o projeto para VR, habilitando o suporte ao Meta Quest.
    - Importe o SDK do Meta Quest para Unity.
3. **Desenvolvimento do Aplicativo**
    - Desenvolva a interface de usuário e a lógica do aplicativo para capturar áudio.
    - Implemente a conexão com a máquina virtual no GCP.

## 4. Configuração da Máquina Virtual no GCP

1. **Criação da Máquina Virtual**
    - Acesse o console do GCP.
    - Crie uma nova instância de máquina virtual (VM) com uma configuração adequada para processar áudio e gerar imagens.
    - Instale as dependências necessárias, incluindo bibliotecas de manipulação de áudio e o modelo SD-T2I-360PanoImage.
2. **Configuração do Servidor**
    - Configure um servidor na VM para receber o áudio enviado pelo aplicativo Unity.
    - Desenvolva um serviço para converter o áudio em um prompt de texto.
    - Integre o modelo SD-T2I-360PanoImage para gerar a imagem panorâmica 360° a partir do prompt de texto.
    - Envie a imagem gerada de volta para o aplicativo Unity.

## 5. Integração e Comunicação

1. **Captura e Envio de Áudio**
    - No aplicativo Unity, capture o áudio do usuário.
    - Envie o áudio para o servidor na VM do GCP utilizando APIs HTTP ou WebSockets.
2. **Processamento no GCP**
    - No servidor GCP, receba e processe o áudio, convertendo-o em um prompt.
    - Gere a imagem 360° utilizando o modelo SD-T2I-360PanoImage.

- Retorne a imagem gerada para o aplicativo Unity.

3. **Exibição da Imagem no Meta Quest 3**
   - No aplicativo Unity, receba a imagem panorâmica 360°.
   - Defina a imagem como fundo de background na cena VR.

## 6. Segurança e Manutenção

1. **Segurança**
   - Implemente autenticação e autorização adequadas para a comunicação entre o aplicativo Unity e o servidor GCP.
   - Utilize HTTPS para proteger a transferência de dados.
2. **Manutenção**
   - Monitore a performance e disponibilidade da VM no GCP.
   - Implemente logs e alertas para identificar e resolver problemas rapidamente.

## 7. Conclusão

Este documento fornece um guia detalhado para configurar a infraestrutura necessária para desenvolver um aplicativo VR no Meta Quest 3 que utiliza uma VM no GCP para processamento de áudio e geração de imagens panorâmicas 360°. Siga as etapas descritas para garantir uma integração suave e eficiente entre os componentes do sistema.

## 8. Referências


- Documentação do Unity: https://docs.unity3d.com/Manual/index.html
- Documentação do Meta Quest: https://developer.oculus.com/
- Documentação do Google Cloud Platform: https://cloud.google.com/docs


## Documentação Detalhada do Projeto Stable DreamFusion

### Introdução

O projeto **Stable DreamFusion** é uma implementação aberta que visa combinar técnicas avançadas de aprendizado de máquina para gerar modelos 3D a partir de prompts de texto. Este repositório é uma implementação do artigo "DreamFusion: Text-to-3D using 2D Diffusion" utilizando o modelo **Stable Diffusion**, uma abordagem popular em síntese de imagens.

**Link para o repositório GitHub:** Stable DreamFusion

### Visão Geral do Projeto

**Stable DreamFusion** permite a criação de objetos 3D detalhados a partir de descrições textuais. Utiliza o poder dos modelos de difusão para transformar entradas textuais em representações visuais que, por sua vez, são usadas para gerar formas 3D. Esta técnica se baseia na combinação de aprendizado supervisionado e não supervisionado, aproveitando redes neurais convolucionais para análise e síntese de imagens.

## Componentes Principais

1. **Stable Diffusion Model:**
   - Modelo de difusão utilizado para gerar imagens 2D de alta qualidade a partir de descrições textuais.
   - Essencial para a primeira etapa do pipeline, onde as descrições textuais são transformadas em imagens.
2. **NeRF (Neural Radiance Fields):**
   - Técnica utilizada para gerar representações 3D a partir de múltiplas visualizações 2D.
   - Fundamental para a transformação de imagens bidimensionais geradas pelo modelo de difusão em modelos tridimensionais.
3. **Blender:**
   - Software de modelagem 3D utilizado para visualizar e refinar os modelos gerados.
   - Integração com scripts Python para automação e ajustes finos dos modelos.

## Fluxo de Trabalho

1. **Entrada de Texto:**
   - O usuário fornece uma descrição textual do objeto que deseja gerar.
2. **Geração de Imagem com Stable Diffusion:**
   - A descrição textual é processada pelo modelo de Stable Diffusion, resultando em uma imagem 2D que representa a descrição.
3. **Conversão para Modelo 3D com NeRF:**
   - As imagens 2D são usadas pelo algoritmo NeRF para criar uma representação 3D do objeto.
   - Essa etapa envolve a síntese de diferentes perspectivas da imagem para construir uma forma tridimensional coerente.
4. **Refinamento e Visualização no Blender:**
   - O modelo 3D gerado é importado para o Blender, onde pode ser visualizado, ajustado e refinado conforme necessário.
   - Scripts personalizados são usados para automatizar partes do processo de refinamento.

## Instalação e Configuração

**Requisitos:**

- Python 3.8+
- CUDA 11.1+ para suporte a GPU NVIDIA
- Dependências Python listadas no arquivo `requirements.txt`

**Passos de Instalação:**

- Clone o repositório:
  `git clone https://github.com/ashawkey/stable-dreamfusion.git`
- `cd stable-dreamfusion`

1.

- Crie e ative um ambiente virtual:
  `python -m venv venv`
- `source venv/bin/activate  # Para Windows: venv\Scripts\activate`

2.

- Instale as dependências:
  `pip install -r requirements.txt`

3. 4. Baixe os pesos do modelo de Stable Diffusion e configure os caminhos apropriados no arquivo de configuração.

## Exemplos de Uso

**Geração de Modelo 3D:**

- Forneça a descrição textual:
  `python generate.py --text "A fantasy castle on a hilltop"`

1. 2. O script processará a entrada, gerará as imagens 2D e criará o modelo 3D.

- Importe e visualize o modelo no Blender:
  `blender --python visualize.py -- --input "output_model.obj"`

3.

## Conclusão

O projeto **Stable DreamFusion** é uma ferramenta poderosa para a criação de modelos 3D a partir de texto, combinando técnicas avançadas de difusão de imagem e campos de radiação neural. Ele oferece um pipeline completo desde a entrada textual até a visualização final no Blender, tornando-o acessível para desenvolvedores e artistas digitais.

# Documento de Explicação sobre Falhas ao Tentar Subir uma VM para Rodar Stable DreamFusion no GCP e Localmente

## Introdução

Ao tentar subir uma máquina virtual (VM) no Google Cloud Platform (GCP) para rodar a plataforma Stable DreamFusion, enfrentei desafios significativos devido às demandas de GPU e outros recursos. Além disso, tentei configurar a plataforma localmente em uma máquina com uma GPU NVIDIA RTX 4060, mas também não obtive sucesso. Este documento explora as razões dessas falhas e os obstáculos enfrentados.

## Desafios no Google Cloud Platform (GCP)

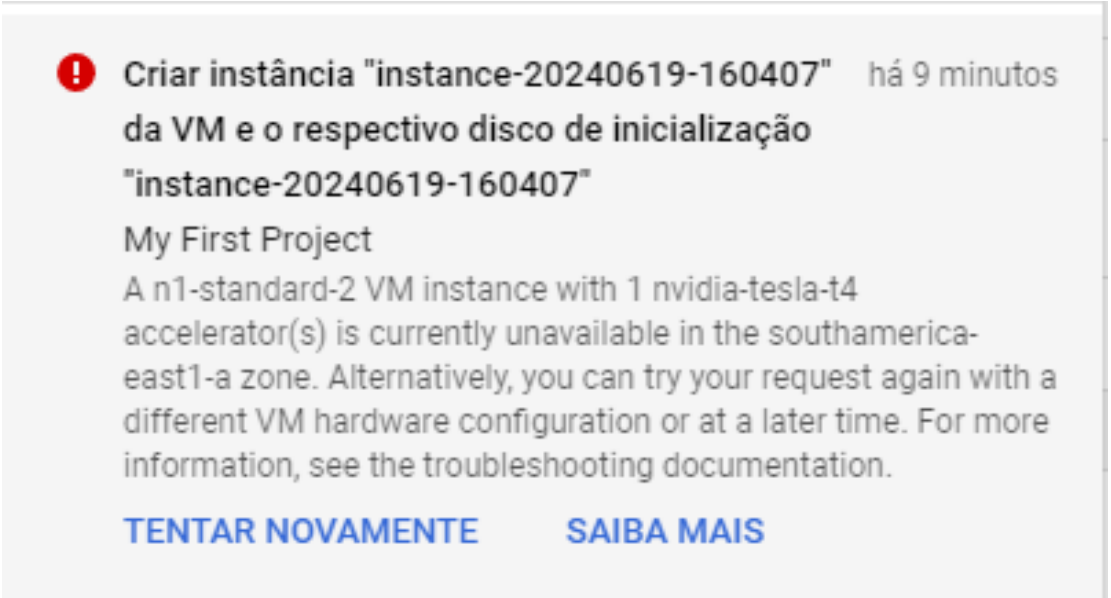
### 1. Alto Custo e Disponibilidade Limitada de GPUs:

- **Custo Elevado:**
  - As GPUs de alta performance necessárias para rodar modelos como o Stable DreamFusion são caras. Instâncias com GPUs potentes, como as séries NVIDIA Tesla ou A100, têm um custo por hora significativamente alto.
  - Mesmo para testes ou uso intermitente, os custos podem escalar rapidamente, tornando-se inviáveis para muitos projetos e desenvolvedores individuais.
- **Disponibilidade Limitada:**
  - GPUs de alto desempenho nem sempre estão disponíveis devido à alta demanda. Muitos usuários competem por essas instâncias, resultando em filas ou indisponibilidade em certas regiões.
  - A capacidade de provisionar essas instâncias pode ser restrita, especialmente em picos de demanda, o que complica ainda mais o uso contínuo para treinamento e inferência de modelos grandes.

### 2. Configuração e Gerenciamento Complexos:

- **Complexidade de Configuração:**
  - Configurar uma VM com GPU no GCP exige conhecimentos avançados em gerenciamento de nuvem, configuração de drivers de GPU, e otimização de ambiente de execução.
  - Pequenos erros na configuração podem levar a falhas ou desempenho subótimo, dificultando a execução eficiente dos modelos.
- **Gerenciamento de Recursos:**
  - Manter e monitorar instâncias de GPU requer um gerenciamento constante para garantir que os recursos estão sendo utilizados de forma eficaz, o que

pode ser complicado sem uma equipe dedicada ou ferramentas especializadas.




## Tentativas de Configuração Local com NVIDIA RTX 4060

**1. Limitações de Hardware:**

- **Memória GPU Insuficiente:**
  - A NVIDIA RTX 4060 possui 8 GB de memória, o que pode ser insuficiente para modelos de difusão e redes neurais volumosas como as usadas pelo Stable DreamFusion.
  - Modelos complexos frequentemente requerem mais memória GPU para processar lotes maiores de dados e realizar inferências eficientes.
- **Capacidade de Cálculo:**
  - Embora a RTX 4060 seja uma GPU potente para muitos aplicativos, pode não atender às necessidades de processamento intensivo de aprendizado profundo exigido por modelos de difusão.
  - GPUs de classe de data center (como as da série Tesla ou A100) são otimizadas para cargas de trabalho de AI e aprendizado profundo, oferecendo desempenho significativamente superior.

**2. Configuração de Software e Drivers:**

- **Drivers e Bibliotecas:**
  - A instalação e configuração adequadas de drivers NVIDIA e bibliotecas como CUDA e cuDNN são cruciais. Pequenos problemas de compatibilidade ou configuração incorreta podem resultar em falhas ou desempenho subótimo.

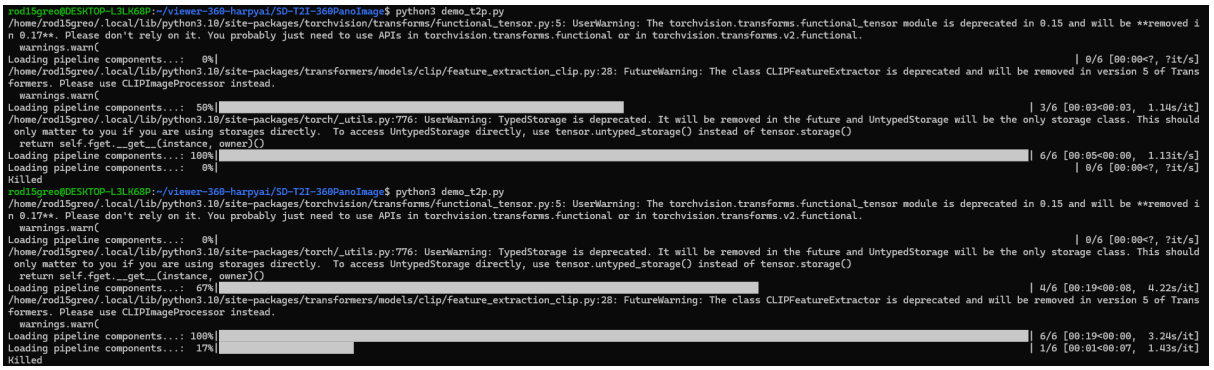
- A configuração de ambientes de execução compatíveis, incluindo versões específicas de frameworks como TensorFlow ou PyTorch, também pode ser desafiadora.

## Conclusão

A tentativa de subir uma VM no GCP para rodar a plataforma Stable DreamFusion falhou principalmente devido aos altos custos associados às GPUs de alto desempenho e à sua disponibilidade limitada. Além disso, a tentativa de configurar a plataforma localmente com uma NVIDIA RTX 4060 foi impedida por limitações de memória e capacidade de processamento da GPU, além de desafios na configuração de drivers e bibliotecas de suporte.

Para superar essas limitações, as seguintes alternativas podem ser consideradas:

- **Uso de Plataformas de Treinamento Dedicadas:**
  - Plataformas como Google Colab Pro+ ou outros serviços de computação em nuvem que oferecem GPUs podem ser uma solução temporária, embora ainda com algumas restrições de custo e disponibilidade.
- **Optimização de Modelos:**
  - Explorar técnicas de otimização de modelos e uso de quantização para reduzir a carga computacional.
- **Colaboração com Instituições:**
  - Parcerias com instituições acadêmicas ou de pesquisa que possam ter acesso a recursos de computação de alto desempenho.



# APÊNDICE 6

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("gate") de aprovação: 26 de jun. de 2024

## Participantes da Entrega [matriculados em Residência em IA]:

RODRIGO MENDES DE CARVALHO

## Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

- Foi feita uma API, na qual a pessoa consegue enviar um prompt e ser retornado uma imagem da plataforma, abaixo está um documento de como fazer a chamada da api em diferentes sistemas operacionais, assim como abaixo está um colab que demonstra a sua funcionalidade. Também está disponível o link da plataforma
  Guia de Uso da API para Gerar Imagens Panorâmicas
  https://218a-34-87-134-219.ngrok-free.app/
  teste_API_plataforma_360
  Bem-vindo ao Visualizador de Panoramas 360° - Google Chrome 2024-06-26 16-37-47.mp4

- Foi terminado o curso para criação dos jogos e aplicativos no óculos VR, na qual foi feito através do Unity HUB, com a integração do Meta Quest Developer Hub e a instalação do Oculus Integration SDK, abaixo está o arquivo compilado do ambiente que pode ser baixado e uma imagens do ambiente criado para teste e um documento de instalação
  https://drive.google.com/file/d/157Q6Tg8riywfs4HgDpgIxKKxI1-2FDQn/view?usp=sharing
  Guia de Instalação e Configuração de Dependências para Desenvolvimento de Aplicações n…

- Foi feito uma tentativa de alocar uma máquina para suportar a plataforma, porém não foi possível pois a minha conta não havia cota para alocar as VMs que rodam o projeto, logo abaixo está o erro que aconteceu no GCP:

  *Criar instância "instance-20240626-133041" da VM e o respectivo disco de inicialização "instance-20240626-133041"há 6 horas*
  *site-viewe-360*
  *A n1-standard-1 VM instance with 2 nvidia-tesla-t4 accelerator(s) is currently unavailable in the us-central1-a zone. Alternatively, you can try your request again with a different VM hardware configuration or at a later time. For more information, see the troubleshooting documentation.*

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Implementar DreamFusion na plataforma web
- Fazer curso para entender o python na Unity Start to Finish Unity® Games and Python Coding
- Implementar API na aplicação do Óculos com a Unity

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Em análise!

**ALDO ANDRÉ DÍAZ SALAZAR:** Em análise!

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("gate") de aprovação: 3 de jul. de 2024

## Participantes da Entrega [matriculados em Residência em IA]:

RODRIGO MENDES DE CARVALHO

## Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

- Iniciei o curso "Start to Finish Unity® Games and Python Coding" para aprimorar minhas habilidades de integração de Python com Unity. Muitas das técnicas ensinadas não são compatíveis com as versões mais recentes do Unity e do Meta Quest SDK.
- Iniciei a integração da tecnologia DreamFusion na plataforma web. Houve um problema de compatibilidade entre o DreamFusion e o framework atual da plataforma. Além disso, a documentação do DreamFusion não estava clara, o que resultou em erros na implementação.

Relatório da Semana

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Revisar e Corrigir a Documentação de DreamFusion
- Atualizar Conhecimentos de Unity e Python

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Em análise!

**ALDO ANDRÉ DÍAZ SALAZAR:** Em análise!

# Guia de Uso da API para imagem 360:

## 1. Windows (PowerShell)

### Usando `Invoke-RestMethod` no PowerShell

1. **Abra o PowerShell.**
2. **Execute o seguinte script:**

```
$headers = @{
    "Content-Type" = "application/json"
}

$body = @{
    "prompt" = "beach"
}

$bodyJson = $body | ConvertTo-Json

Invoke-RestMethod -Uri "https://<ngrok-url>/api" -Method Post
-Headers $headers -Body $bodyJson -OutFile result.png
```

### Usando `Invoke-WebRequest` no PowerShell

1. **Abra o PowerShell.**
2. **Execute o seguinte script:**

```
$headers = @{
```

```
    "Content-Type" = "application/json"
}

$body = @{
    "prompt" = "beach"
}

$bodyJson = $body | ConvertTo-Json

Invoke-WebRequest -Uri "https://<ngrok-url>/api" -Method Post
-Headers $headers -Body $bodyJson -OutFile result.png
```

## 2. Windows (Command Prompt)

**Usando `curl` no Command Prompt**

1. **Abra o Command Prompt.**
2. **Execute o seguinte comando:**

```
curl -X POST "https://<ngrok-url>/api" -H "Content-Type:
application/json" -d "{\"prompt\": \"beach\"}" --output result.png
```

## 3. Linux / macOS

**Usando `curl` no Terminal**

1. **Abra o Terminal.**
2. **Execute o seguinte comando:**

```
curl -X POST "https://<ngrok-url>/api" -H "Content-Type:
application/json" -d '{"prompt": "beach"}' --output result.png
```

## 4. Windows (WSL - Windows Subsystem for Linux)

**Usando `curl` no Terminal do WSL**

1. **Abra o Terminal do WSL.**
2. **Execute o seguinte comando:**

```
curl -X POST "https://<ngrok-url>/api" -H "Content-Type:
application/json" -d '{"prompt": "beach"}' --output result.png
```

## 5. Python (Cross-Platform)

**Usando `requests` em um Script Python**

1. **Instale a biblioteca `requests` se ainda não estiver instalada:**

sh

Copy code

```
pip install requests
```

2. **Crie um script Python com o seguinte conteúdo:**

```
import requests


url = "https://<ngrok-url>/api"

headers = {

    "Content-Type": "application/json"

}

data = {
```

```
    "prompt": "beach"
}

response = requests.post(url, headers=headers, json=data)

if response.status_code == 200:
    with open("result.png", "wb") as f:
        f.write(response.content)
else:
    print("Failed to retrieve image:", response.status_code,
response.text)
```

3. **Execute o script:**

e

- `python script_name.py`
-

## 6. Postman

**Usando Postman para Chamar a API**

1. **Abra o Postman.**
2. **Crie uma nova requisição:**
   - **Método:** POST
   - **URL:** `https://<ngrok-url>/api`
   - **Headers:** Adicione um cabeçalho `Content-Type` com o valor `application/json`.
   - **Body:** Selecione `raw` e escolha `JSON` no formato. Adicione o seguinte conteúdo:

```
{
```

```
    "prompt": "beach"
}
```

3.
4. **Envie a requisição.**
5. **Clique em `Save Response` e salve a imagem como `result.png`.

# Notebook Rodrigo Jupyter: Teste da API da plataforma 360

## Conexão com API do site:

In [ ]:

```
!curl -X POST "https://3030-34-87-134-219.ngrok-free.app/api" -H
"Content-Type: application/json" -d '{"prompt": "beach"}' --output
result.png
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 18.0M  100 18.0M    0     19  75190      0  0:04:12  0:04:12 --:--:-- 1230k
```

## Plota a imagem da API:

In [ ]:

```
import matplotlib.pyplot as plt

import matplotlib.image as mpimg


# Caminho da imagem que você deseja plotar

image_path = '/content/result.png'


# Carrega a imagem

img = mpimg.imread(image_path)


# Plota a imagem

plt.imshow(img)
```

```
plt.axis('off')  # Remove os eixos

plt.show()
```

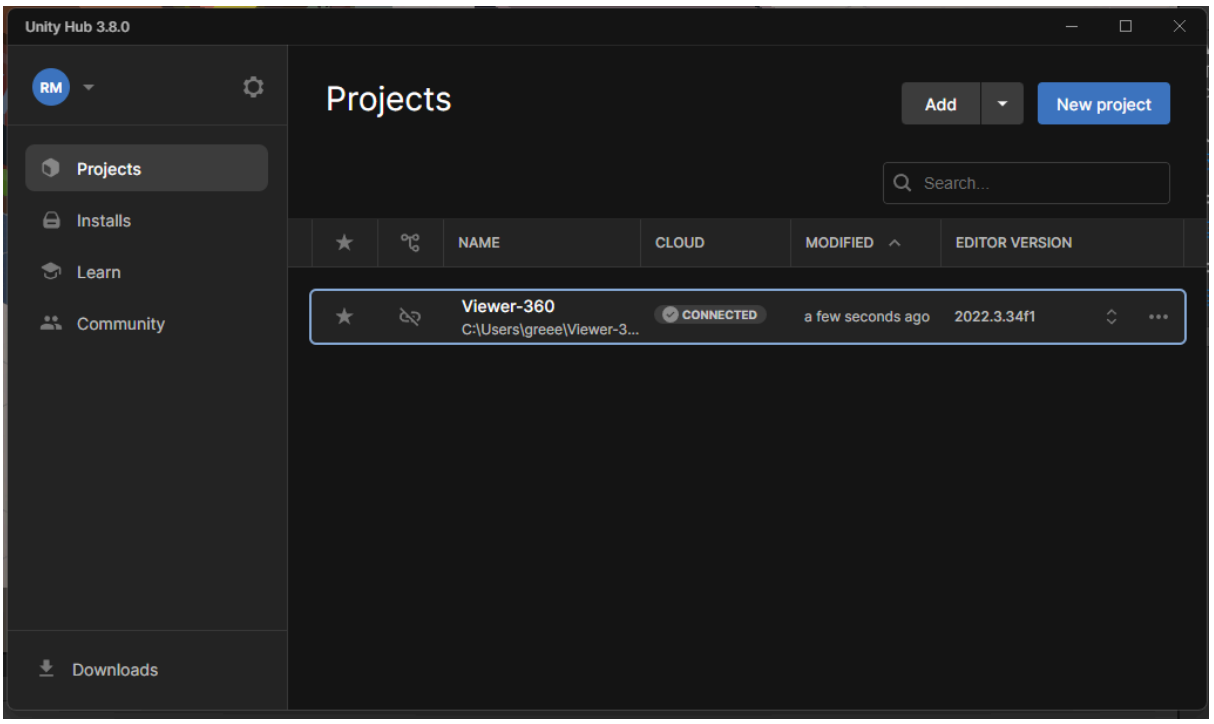
# Guia de Instalação e Configuração de Dependências para Desenvolvimento de Aplicações no Meta Quest 3

## 1. Instalação do Unity Hub

O Unity Hub é um aplicativo que ajuda você a gerenciar suas instalações do Unity Editor, seus projetos Unity e suas contas Unity.

### Passos para Instalar o Unity Hub:

1. **Baixe o Unity Hub:**
   - Acesse a página de download do Unity Hub: Unity Hub Download
2. **Instale o Unity Hub:**
   - Execute o instalador baixado e siga as instruções na tela para completar a instalação.
3. **Inicie o Unity Hub:**
   - Abra o Unity Hub após a instalação.
4. **Instale o Unity Editor:**
   - No Unity Hub, vá para a guia `Installs`.
   - Clique em `Add` para adicionar uma nova instalação do Unity Editor.
   - Selecione a versão recomendada ou a versão de sua preferência e clique em `Next`.
   - Escolha os módulos adicionais necessários (recomenda-se adicionar o suporte para Android) e clique em `Done`.

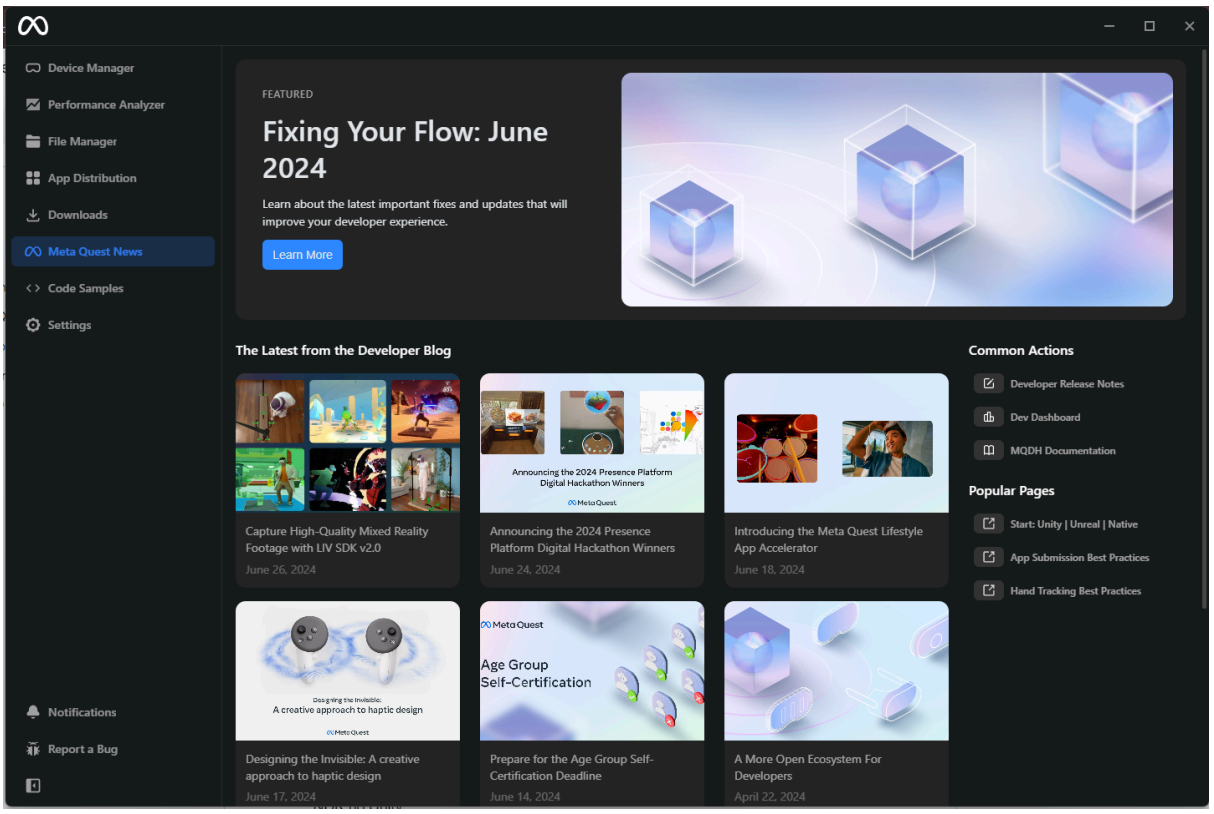



## 2. Configuração do Meta Quest Developer Hub

O Meta Quest Developer Hub (Oculus Developer Hub) é uma ferramenta que facilita o desenvolvimento para dispositivos Meta Quest, fornecendo acesso a diversos recursos e configurações.

### Passos para Instalar e Configurar o Meta Quest Developer Hub:

1. **Baixe o Meta Quest Developer Hub:**
   - Acesse a página de download: Meta Quest Developer Hub Download
2. **Instale o Meta Quest Developer Hub:**
   - Execute o instalador baixado e siga as instruções na tela para completar a instalação.
3. **Inicie o Meta Quest Developer Hub:**
   - Abra o Meta Quest Developer Hub após a instalação.
4. **Configure seu dispositivo Meta Quest:**
   - Conecte seu Meta Quest 3 ao PC via cabo USB.
   - No Meta Quest Developer Hub, vá para a seção `Devices`.
   - Siga as instruções para adicionar e configurar seu dispositivo Meta Quest 3.




## 3. Configuração do Ambiente de Desenvolvimento

### Instalação do Android SDK e NDK

Para desenvolver para o Meta Quest 3, é necessário configurar o Android SDK e NDK no Unity.

1. **Instale o Android Studio:**
   - Baixe e instale o Android Studio a partir do site oficial.
2. **Configure o SDK e NDK:**
   - Abra o Android Studio.
   - Vá para `Configure` > `SDK Manager`.
   - Na aba `SDK Platforms`, selecione as versões do Android que você deseja suportar.
   - Na aba `SDK Tools`, certifique-se de que o Android SDK Build-Tools, Android SDK Platform-Tools e o NDK estão instalados.

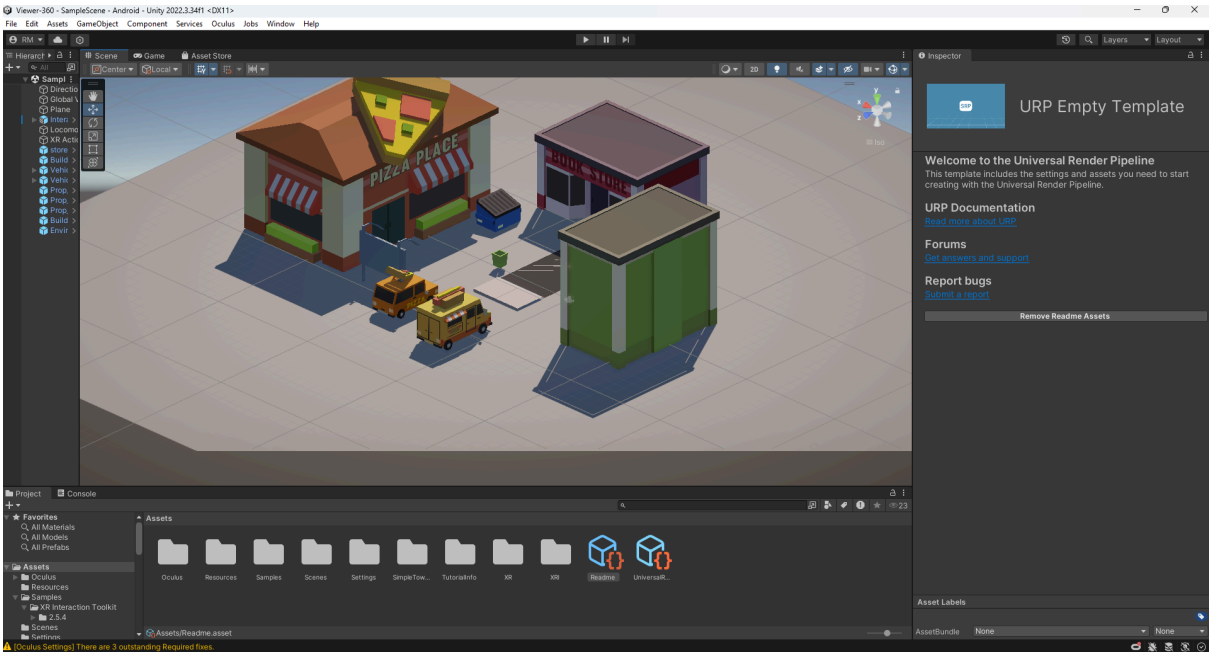
### Configuração no Unity

1. **Abra o Unity Hub e crie um novo projeto:**
   - No Unity Hub, clique em `New` para criar um novo projeto.
   - Selecione o template `3D` ou outro de sua preferência e configure o nome e local do projeto.

2. **Configure o suporte para Android no Unity:**
    - No Unity Editor, vá para `Edit` > `Preferences`.
    - Selecione a aba `External Tools`.
    - Configure o caminho do Android SDK, NDK e JDK conforme necessário.
3. **Configure o projeto para o Meta Quest:**
    - No Unity Editor, vá para `File` > `Build Settings`.
    - Selecione `Android` e clique em `Switch Platform`.
    - Clique em `Player Settings`.
    - Na seção `XR Plug-in Management`, habilite o `Oculus`.
4. **Configure o Package Manager:**
    - No Unity Editor, vá para `Window` > `Package Manager`.
    - Certifique-se de que os pacotes necessários para VR estão instalados, como o `XR Interaction Toolkit`.

## Implementação e Teste

1. **Desenvolva sua aplicação:**
    - Use as ferramentas e recursos do Unity para criar sua aplicação VR.
2. **Build e Deploy:**
    - No Unity Editor, vá para `File` > `Build Settings`.
    - Configure as definições de build e clique em `Build and Run` para compilar e enviar a aplicação para seu dispositivo Meta Quest 3.



## Recursos Adicionais

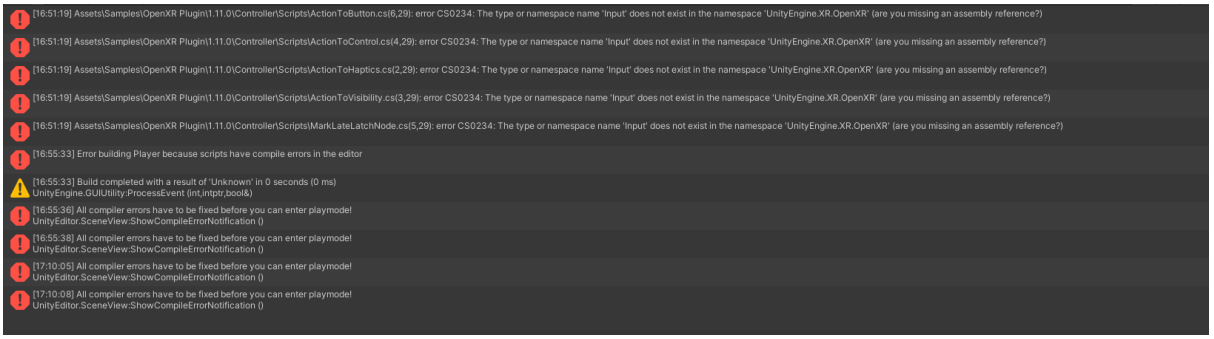
- **Documentação do Unity:** Unity Documentation
- **Documentação do Meta Quest:** Meta Quest Developer Documentation
- **Fóruns de Suporte:** Unity Forum e Oculus Forum

# Relatório de erros de integração de bibliotecas do Meta na Unity

## Resumo das Atividades

Nesta semana, foquei em dois principais projetos: a integração de Python com Unity e Meta Quest, e a implementação da tecnologia DreamFusion em uma plataforma web utilizando um servidor alocado no Colab Notebook. Enfrentei diversos desafios técnicos que detalhei a seguir.

## Projeto de Integração de Python com Unity e Meta Quest




## Atividades Realizadas

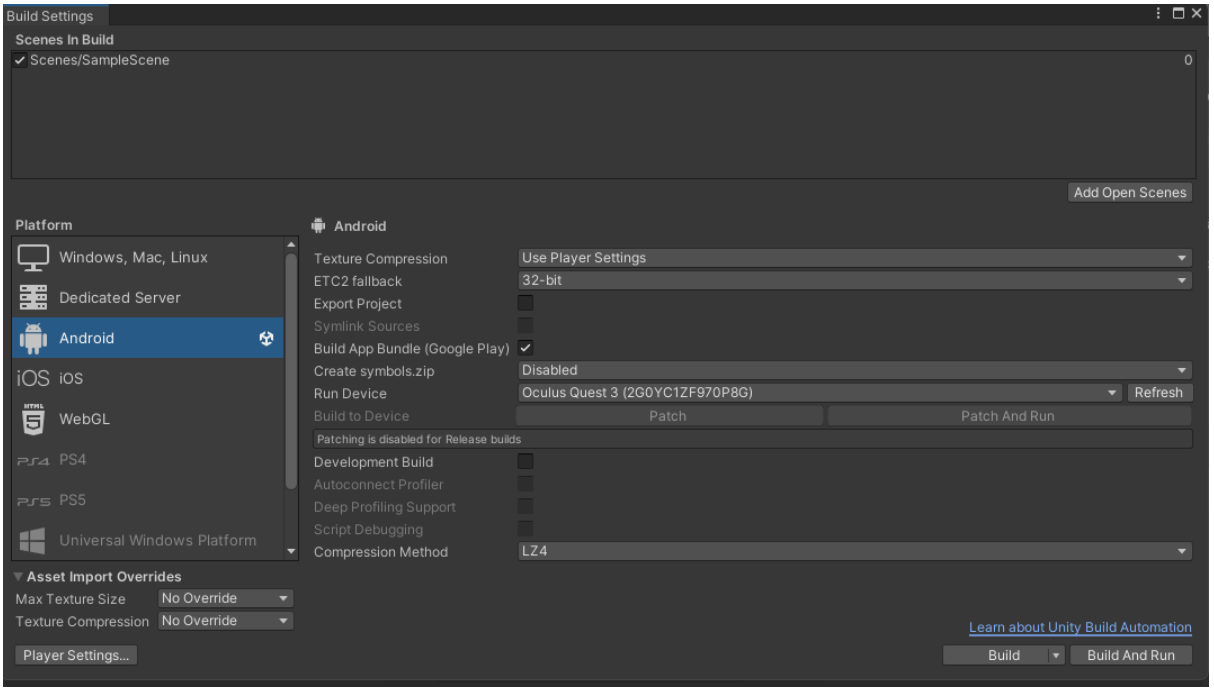
1. **Curso "Start to Finish Unity® Games and Python Coding":**
    - Iniciado para aprimorar habilidades de integração de Python com Unity.
    - Estudo de técnicas de scripting em Python aplicadas a projetos de jogos no Unity.
    - Aplicação de exemplos práticos para conectar scripts Python com o ambiente Unity.

## Erros e Desafios Encontrados

1. **Compatibilidade com Versões Recentes:**
    - **Problema:** Muitas técnicas ensinadas no curso não são compatíveis com as versões mais recentes do Unity (2023.1) e do Meta Quest SDK (v50).
    - **Solução Tentada:** Tentei adaptar os exemplos do curso para as versões mais recentes, mas sem sucesso em alguns casos devido a mudanças significativas na API.

- **Solução Proposta:** Consultar a documentação oficial atualizada e buscar exemplos na comunidade de desenvolvedores que utilizam as versões mais recentes.

2. **Mudanças na API do Unity:**
   - **Problema:** Diversos métodos e classes utilizados no curso foram depreciados ou modificados nas versões mais recentes do Unity.
   - **Solução Tentada:** Utilizar o Assembly Definition Files para garantir compatibilidade e melhorar a estrutura do projeto.
3. **Integração com Meta Quest SDK:**
   - **Problema:** O Meta Quest SDK introduziu novas funcionalidades e modificações que não eram abordadas no curso, resultando em dificuldades na implementação.
   - **Solução Tentada:** Tentei seguir a documentação oficial do Meta Quest SDK, mas encontrei inconsistências e falta de exemplos práticos..
4. **Conexão entre Python e Unity:**
   - **Problema:** Dificuldades em estabelecer uma comunicação estável entre scripts Python e o ambiente Unity devido a incompatibilidades de versão e bibliotecas utilizadas.
   - **Solução Tentada:** Utilização de bibliotecas como Python for Unity e integração via sockets.
   - **Solução Proposta:** Explorar outras bibliotecas ou frameworks que ofereçam melhor suporte para a integração desejada.



## Projeto de Implementação da Tecnologia DreamFusion

## Atividades Realizadas

1. **Início da Integração da Tecnologia DreamFusion:**
    - Objetivo de integrar DreamFusion na plataforma web para geração de conteúdo 3D.
    - Configuração de um servidor no Colab Notebook para teste e desenvolvimento.

## Erros e Desafios Encontrados

1. **Compatibilidade com Framework Atual:**
    - **Problema:** Houve um problema de compatibilidade entre o DreamFusion e o framework atual da plataforma web, resultando em falhas na integração.
    - **Solução Tentada:** Modificar o código-fonte do DreamFusion para tentar adaptar ao framework existente, mas resultou em erros adicionais.
2. **Documentação Inadequada:**
    - **Problema:** A documentação do DreamFusion não estava clara, com falta de exemplos detalhados e explicações completas sobre a API.
    - **Solução Tentada:** Consulta frequente aos poucos exemplos fornecidos e tentativa de extrapolar soluções a partir de informações limitadas.
3. **Recursos Limitados do Colab Notebook:**
    - **Problema:** O uso do Colab Notebook apresentou limitações de recursos, como memória e tempo de execução, afetando a performance das tarefas do DreamFusion.
    - **Solução Tentada:** Otimização do código para reduzir o consumo de recursos e tentativa de execução em sessões premium do Colab.
    - **Solução Proposta:** Considerar a migração para servidores com maior capacidade de recursos, como instâncias de VM na GCP ou AWS, para suportar a carga de trabalho exigida.
4. **Erros Específicos no Colab:**
    - **Problema:** Erros relacionados ao tempo limite de execução (timeout) e esgotamento de memória durante a execução de tarefas intensivas do DreamFusion.
    - **Solução Tentada:** Dividir as tarefas em etapas menores para evitar o esgotamento de recursos e monitoramento constante do uso de memória.
    - **Solução Proposta:** Implementar um sistema de checkpoints que permite salvar o progresso e reiniciar a partir de pontos intermediários, além de otimizar a alocação de memória.

## Próximos Passos

1. **Para Integração de Python com Unity e Meta Quest:**

- Atualização de conhecimentos por meio de cursos e tutoriais compatíveis com as versões atuais do Unity e do Meta Quest SDK.
- Participação ativa em fóruns e grupos de discussão para obter suporte da comunidade.
- Reescrever e adaptar scripts Python para utilizar APIs atualizadas e práticas recomendadas.

2. **Para Implementação da Tecnologia DreamFusion:**
   - Colaboração com a equipe de desenvolvimento da plataforma web para resolver problemas de compatibilidade.
   - Contribuição para a melhoria da documentação do DreamFusion e busca de apoio em comunidades especializadas.
   - Migração para servidores mais robustos, se necessário, para suportar as demandas de processamento do DreamFusion.

# APÊNDICE 7

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("gate") de aprovação: 10 de jul. de 2024

## Participantes da Entrega [matriculados em Residência em IA]:

RODRIGO MENDES DE CARVALHO

## Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

- Foi feita a integração do Meta Quest 3 com a Unity, onde fiz toda a integração de uma imagem 360 para o fundo do ambiente na qual podemos visualizar o cenário criado, logo abaixo tem o link de demonstração.
  VID-20240710-WA0098.mp4
- Foi feita uma interface no ambiente na qual a pessoa grava seu áudio e para a gravação transcrevendo o microfone do Meta Quest 3 para o campo de texto, por trás das funcionalidades foi criado um código em C# para integrar botões ao áudio. Para poder usar a transcrição foi usado o Google Speech-to-Text. Logo abaixo tem o passo a passo da criação da interface e o código em C#
  https://github.com/Rod15greo/Residencia_IA_Rodrigo_Mendes/blob/main/meta-quest-3/scripts/VoiceRecorder.cs
  Interface Unity Meta

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- Refazer chamada da API em C# e integração da imagem
- Subir jogo no https://itch.io/ e baixar ele no meta
- Fazer testes de qualidade e desempenho e ajustar

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Em análise!

**ALDO ANDRÉ DÍAZ SALAZAR:** Em análise!

# Interface Unity Meta

## Passo 1: Configurar o Projeto

1. **Criar um Novo Projeto Unity:**
   - Abra o Unity Hub e crie um novo projeto 3D ou VR.
2. **Adicionar Pacotes Necessários:**
   - Certifique-se de que você tem os pacotes XR configurados para o Meta Quest 3.
   - Adicione o pacote `TextMeshPro` para uma melhor interface de texto.

## Passo 2: Criar a Interface de Usuário

1. **Adicionar um Canvas:**
   - Vá para `GameObject` > `UI` > `Canvas` para criar um Canvas.
2. **Adicionar Botões e Caixa de Texto:**
   - Adicione dois botões (`Start Recording` e `Stop Recording`).
   - Adicione um campo de texto usando `TextMeshPro` para exibir a transcrição.

## Passo 3: Capturar Áudio com C#

1. **Criar um Script de Captura de Áudio:**
   - No `Project Window`, clique com o botão direito e selecione `Create` > `C# Script`. Nomeie o script como `VoiceRecorder`.
2. **Editar o Script VoiceRecorder:**

```
using UnityEngine;

using UnityEngine.UI;

using TMPro;

using System.Collections;

using System.Collections.Generic;

using System.IO;

using UnityEngine.Networking;
```

```
public class VoiceRecorder : MonoBehaviour
{
    public Button startButton;
    public Button stopButton;
    public TextMeshProUGUI transcriptText;

    private AudioSource audioSource;
    private string microphoneName;
    private AudioClip recordedClip;

    void Start()
    {
        startButton.onClick.AddListener(StartRecording);
        stopButton.onClick.AddListener(StopRecording);
        audioSource = gameObject.AddComponent<AudioSource>();
        microphoneName = Microphone.devices[0];
    }

    void StartRecording()
    {
        recordedClip = Microphone.Start(microphoneName, false, 10, 44100);
        audioSource.clip = recordedClip;
```

```
    }

    void StopRecording()
    {
        Microphone.End(microphoneName);
        StartCoroutine(ProcessAudio());
    }

    IEnumerator ProcessAudio()
    {
        string filePath = Path.Combine(Application.persistentDataPath,
"recordedAudio.wav");

        // Save the recorded audio to a file
        SaveWavFile(filePath, recordedClip);

        // Send the audio file to Google Speech-to-Text API
        string transcription = yield return StartCoroutine(SendAudioToGoogle(filePath));
        transcriptText.text = transcription;
    }

    void SaveWavFile(string filePath, AudioClip clip)
    {
        var samples = new float[clip.samples];
```

```
        clip.GetData(samples, 0);

        byte[] wavFile = WavUtility.FromAudioClip(clip, filePath, false);

        File.WriteAllBytes(filePath, wavFile);

    }


    IEnumerator SendAudioToGoogle(string filePath)

    {

        byte[] audioData = File.ReadAllBytes(filePath);

        string base64Audio = System.Convert.ToBase64String(audioData);


        var request = new
UnityWebRequest("https://speech.googleapis.com/v1/speech:recognize?key=YOUR_
API_KEY", "POST");

        request.uploadHandler = new
UploadHandlerRaw(System.Text.Encoding.UTF8.GetBytes("{\"config\":{\"encoding\":\"
LINEAR16\",\"sampleRateHertz\":44100,\"languageCode\":\"en-US\"},\"audio\":{\"cont
ent\":\"" + base64Audio + "\"}}"));

        request.downloadHandler = new DownloadHandlerBuffer();

        request.SetRequestHeader("Content-Type", "application/json");


        yield return request.SendWebRequest();


        if (request.result == UnityWebRequest.Result.Success)

        {

            string jsonResponse = request.downloadHandler.text;

            var json = SimpleJSON.JSON.Parse(jsonResponse);
```

```
            string transcription = json["results"][0]["alternatives"][0]["transcript"];
            yield return transcription;
        }
        else
        {
            Debug.LogError("Error in speech recognition: " + request.error);
            yield return null;
        }
    }
}
```

3. **Adicionar o Script ao GameObject:**
   - Crie um GameObject vazio na cena e nomeie-o como `VoiceRecorder`.
   - Arraste o script `VoiceRecorder` para esse GameObject.
4. **Configurar o Script no Inspector:**
   - Arraste e solte os botões `Start Recording` e `Stop Recording` para os respectivos campos no script `VoiceRecorder`.
   - Arraste e solte o campo de texto `TextMeshPro` para o campo `transcriptText` no script.

## Passo 4: Configurar a API do Google Speech-to-Text

1. **Obter uma Chave de API:**
   - Vá para o console do Google Cloud e ative a API de reconhecimento de fala.
   - Crie uma nova chave de API.
2. **Inserir a Chave de API no Script:**
   - Substitua `YOUR_API_KEY` no script `VoiceRecorder` pela sua chave de API.

## Passo 5: Testar a Implementação

5. **Testar no Editor:**
   a. Execute a cena no editor do Unity e teste a gravação e transcrição de voz.
6. **Build e Teste no Meta Quest 3:**
   a. Compile o projeto para a plataforma Android e instale-o no Meta Quest 3.
   b. Teste a funcionalidade no dispositivo VR.