Sistema de Estimativa de Profundidade usando MiDaS com Interface Gráfica em Tkinter

Autor do Trabalho

1 Introdução

A visão computacional busca extrair informações tridimensionais a partir de imagens bidimensionais. Em particular, a estimação de profundidade monocular consiste em prever valores de profundidade para cada pixel de uma imagem única, tarefa fundamental para aplicações como realidade aumentada, robótica e reconstrução 3D. Entretanto, essa tarefa é intrinsecamente ambígua, pois a informação de escala está ausente na imagem monocular:contentReference[oaicite:0]index=0. Modelos modernos de aprendizado profundo, como o MiDaS (Monocular Depth Estimation via a Mixture of Datasets), vêm sendo desenvolvidos para produzir mapas de profundidade relativos robustos e generalizáveis. Ranftl et al. (2022) demonstram que a utilização de múltiplos conjuntos de dados complementares e uma função de perda invariante à escala e ao deslocamento melhora significativamente a acurácia e a robustez da estimativa monocular de profundidade:contentReference[oaicite:1]index=1. Nesse contexto, este trabalho descreve o desenvolvimento de um sistema completo que integra o modelo MiDaS a uma interface gráfica em Python, permitindo estimar profundidade de imagens estáticas e de vídeo em tempo real, além de oferecer ferramentas de medição interativa e visualização 3D dos resultados.

2 Objetivos

Os principais objetivos deste projeto incluem: - Implementar um sistema capaz de carregar imagens ou capturar vídeo por webcam e estimar mapas de profundidade monocular usando o modelo MiDaS pré-treinado. - Desenvolver uma interface gráfica usando Tkinter que permita ao usuário interagir facilmente com as funcionalidades (carregar imagem, iniciar câmera, visualizar

resultados, etc.). - Fornecer ferramentas de visualização dos mapas de profundidade resultantes, incluindo apresentação em escalas de cor, visualização 3D de nuvens de pontos e gráficos auxiliares. - Incluir uma funcionalidade de medição interativa: o usuário poderá selecionar pontos na imagem e obter estimativas de distância relativa baseadas no mapa de profundidade. - Operar em tempo real: o sistema deverá suportar captura de vídeo via webcam, aplicando o modelo MiDaS frame a frame para estimativa de profundidade dinâmica.

3 Fundamentação Teórica

3.1 Visão Computacional e Estimativa de Profundidade

Visão computacional é um campo que trata da interpretação automática de imagens e vídeos para extrair informações úteis sobre o ambiente. Uma aplicação crucial é a recuperação da profundidade (ou geometria 3D) de uma cena a partir de imagens 2D. Para imagens estéreo ou multivistas, a profundidade pode ser obtida por correspondência de pixels; porém, no caso de imagens monoculares essa tarefa é menos direta, pois falta disparidade estereoscópica. Eigen et al. (2014) observaram que a previsão de profundidade de uma única imagem requer integração de informações globais e locais e ainda enfrenta ambiguidade de escala:contentReference[oaicite:2]index=2. Redes neurais convolucionais (CNNs) multi-escala foram propostas para resolver essa questão, combinando predições grosseiras de profundidade global com refinamentos locais. Apesar da ambiguidade inerente, técnicas de aprendizado profundo têm alcançado resultados de estado da arte ao explorar grandes quantidades de dados e arquiteturas sofisticadas.

3.2 Modelo MiDaS

O modelo MiDaS (Monocular Depth Estimation via a Mixture of Datasets) é uma abordagem avançada para estimar profundidade relativa a partir de imagens monoculares. Conforme descrito por Ranftl et al., a versão original do MiDaS foi treinada combinando diversos conjuntos de dados heterogêneos e utilizando funções de perda invariantes à escala e deslocamento:contentReference[oaicite:3]index=3. Isso permite que o modelo aprenda relações de profundidade relativas robustas, sem depender de uma escala métrica fixa. Birkl et al. (2023) apresentam a coleção de modelos MiDaS 3.1, explicando que o MiDaS opera em espaço de disparidade (profundi-

dade inversa até uma constante de escala) e que cada nova versão integra diferentes *backbones* (como ResNet, EfficientNet ou Transformers) na arquitetura encoder-decoder padrão:contentReference[oaicite:4]index=4. De modo geral, o MiDaS fornece mapas de profundidade relativa com qualidade e generalização superiores em ambientes diversos, conforme evidenciado por resultados de zero-shot em múltiplos benchmarks:contentReference[oaicite:5]index=5:contentReference[oaicite:6

3.3 GUI com Tkinter

Tkinter é a biblioteca padrão do Python para desenvolvimento de interfaces gráficas (GUI). Ela fornece bindings em Python para o toolkit Tcl/Tk e está disponível na distribuição padrão do Python em várias plataformas. Conforme descrito na documentação oficial, "o pacote `tkinter` é a interface padrão do Python para o toolkit Tcl/Tk":contentReference[oaicite:7]index=7. Tkinter oferece widgets básicos (botões, labels, canvas, etc.) que permitem criar janelas interativas com relativa facilidade. Por ser parte da biblioteca padrão, Tkinter é de fácil acesso e não requer instalações adicionais, o que o torna uma escolha comum para prototipagem rápida de GUIs em aplicações de visão computacional e aprendizado de máquina.

3.4 Processamento de Imagens com OpenCV

OpenCV (Open Source Computer Vision Library) é uma das bibliotecas mais populares para processamento de imagens e visão computacional. Desenvolvida inicialmente em C/C++, possui interfaces em Python que permitem realizar operações de baixo nível (leitura, escrita, transformações de pixel) e de alto nível (detecção de bordas, segmentação, transformações geométricas, calibração de câmeras, etc.). Bradski e Kaehler (2008) apresentam o OpenCV como uma biblioteca de código aberto para visão computacional ampla:contentReference[oaicite:8]index=8. Em projetos de estimativa de profundidade, o OpenCV é utilizado tanto para pré-processar imagens (como redimensionamento e normalização) quanto para capturar vídeo em tempo real e exibir os mapas de profundidade resultantes em janelas nativas. Além disso, o OpenCV permite lidar diretamente com eventos de mouse ou teclado, facilitando a implementação de ferramentas interativas.

4 Metodologia

4.1 Arquitetura da Aplicação

A arquitetura da aplicação foi projetada de forma modular, separando responsabilidades em componentes distintos. Em geral, foram criados os seguintes módulos principais: (*i*) *Interface Gráfica* (GUI) em Tkinter, responsável pela interação com o usuário (botões, visualização de imagens, etc.); (*ii*) *Processamento de Imagens* baseado em OpenCV, para tarefas como carregar imagem, capturar quadros da câmera e converter formatos; (*iii*) *Estimador de Profundidade* usando o modelo MiDaS, que recebe uma imagem e retorna o mapa de profundidade correspondente; (*iv*) *Visualização de Resultados*, utilizando Matplotlib e Open3D para apresentar os mapas de profundidade e nuvens de pontos 3D. Essa divisão modular facilita a manutenção e permite testar cada parte independentemente.

4.2 Pipeline de Estimativa de Profundidade

O pipeline para estimativa de profundidade segue passos sequenciais: primeiro, a imagem de entrada é carregada (ou capturada da webcam) e pré-processada (redimensionamento, normalização e conversão para formato tensor). Em seguida, essa imagem é passada ao modelo MiDaS (rodando em PyTorch) que realiza inferência e produz um mapa de profundidade relativa. O mapa de profundidade bruto, normalmente em escala arbitrária, é então pós-processado: normalizado para valores de intensidade adequados e convertido em imagem (por exemplo, escala de cinza ou mapa de calor). Finalmente, o mapa de profundidade é exibido na interface do usuário, associado à imagem original. Todo esse fluxo é executado tanto para imagens estáticas quanto para cada quadro capturado em tempo real da webcam, garantindo a atualização contínua do resultado.

4.3 Funcionalidades da Interface do Usuário

A interface gráfica desenvolvida em Tkinter inclui botões e áreas de visualização para todas as funcionalidades principais. O usuário pode: carregar uma imagem de arquivo local; iniciar e parar a captura de vídeo da webcam; visualizar a imagem original e o mapa de profundidade estimado lado a lado; alternar para o modo de medição; e acionar a visualização em 3D. Cada funcionalidade está associada a controles claros (botões ou menus) e exibida em um `Canvas` ou `Label` do Tkinter. Também foram adicionados indicadores de status (por exemplo, mostrando quando o modelo está processando uma

imagem) para melhorar a experiência. Em resumo, a GUI integra de forma intuitiva o fluxo de entrada (imagem/câmera), processamento (estimativa de profundidade) e saída (visualização) do sistema.

4.4 Ferramenta de Medição

A ferramenta de medição interativa permite ao usuário clicar em dois pontos arbitrários do mapa de profundidade exibido. Ao detectar os cliques (por meio de eventos de mouse do Tkinter ou OpenCV), o sistema obtém os valores de profundidade relativos nos dois pixels selecionados. Supondo dados de calibração ou adotando-se uma profundidade média de referência, é possível estimar a distância entre esses pontos no espaço tridimensional. Essa estimativa é exibida ao usuário em tempo real, mostrando por exemplo a distância em unidades arbitrárias (ou metros, caso os parâmetros da câmera sejam conhecidos). Em suma, a ferramenta utiliza o mapa de disparidade fornecido pelo MiDaS para calcular distâncias relativas entre pontos da cena.

4.5 Processamento em Tempo Real com Câmera

Para suportar vídeo em tempo real, utilizamos o módulo `VideoCapture` do OpenCV para ler quadros da webcam em uma taxa fixa. Um loop principal contínuo capta cada quadro, executa o pipeline de estimação de profundidade descrito acima e atualiza a GUI com os resultados. Como a inferência do modelo MiDaS pode ser custosa, aplicamos técnicas de otimização (por exemplo, redimensionamento para tamanhos menores ou processamento em threads separadas) para manter a taxa de quadros aceitável. O resultado é que o sistema exibe, continuamente, o feed de vídeo original e o correspondente mapa de profundidade em tempo quase real, permitindo ao usuário mover a câmera e observar as mudanças de profundidade instantaneamente.

4.6 Ferramentas de Visualização

Os mapas de profundidade estimados são visualizados usando duas abordagens complementares. Primeiro, utilizamos o Matplotlib para exibir imagens 2D estáticas: por exemplo, como imagens em escala de cinza ou coloridas para facilitar a interpretação visual. O Matplotlib, biblioteca consolidada para gráficos em Python:contentReference[oaicite:9]index=9, permite renderizar facilmente os mapas de profundidade e até plotar histogramas ou perfis de profundidade se necessário. Segundo, para visualização tridimensional, empregamos o Open3D. Com ele, convertemos o mapa de profundidade em uma nuvem de pontos 3D (calculando as coordenadas

x, y, z a partir das coordenadas de pixel e da profundidade) e usamos a função `draw_geometries` do Open3D para renderizar interativamente a cena 3D:contentReference[oaicite:10]index=10. Essa visualização 3D permite ao usuário girar e aproximar a nuvem de pontos, obtendo uma percepção espacial dos dados de profundidade estimados.

5 Plano de Trabalho

5.1 Levantamento de Requisitos

- Identificar os requisitos funcionais junto aos stakeholders (ex.: tipos de imagens suportadas, precisão desejada, necessidade de operação em tempo real). - Definir requisitos não funcionais, como desempenho mínimo (FPS na câmera), tempo de resposta da interface e restrições de hardware. - Especificar as funcionalidades essenciais (cálculo de profundidade, medição interativa, visualização 3D) e critérios de aceite para cada uma.

5.2 Escolha de Tecnologias e Modelos

- Decidir utilizar Python como linguagem principal, pela disponibilidade das bibliotecas desejadas (PyTorch, Tkinter, OpenCV, Open3D, Matplotlib). - Selecionar o modelo MiDaS (versão mais recente disponível) para a inferência de profundidade. - Justificar o uso de Tkinter para interface (disponibilidade padrão, simplicidade) em contraposição a outras opções de GUI. - Escolher ferramentas de visualização (Open3D para 3D, Matplotlib para 2D) baseando-se em exemplos de sucesso na literatura.

5.3 Implementação Modular

- Estruturar o código em módulos ou classes: e.g. módulo de interface, módulo de inferência de profundidade, módulo de utilidades (processamento de imagem), módulo de visualização. - Implementar cada módulo separadamente, criando funções/pipelines distintos para teste isolado: por exemplo, testar a carga de imagem e exibição sem incluir o modelo, ou validar o modelo MiDaS em um script separado. - Utilizar boas práticas de programação, como documentação inline e controle de versão, para manter o código organizado.

5.4 Integração de Componentes

- Integrar progressivamente os módulos desenvolvidos: por exemplo, conectar a função de carga de imagem da GUI ao pipeline de inferência de profundidade. - Garantir que a interface permita executar cada função (carregar imagem, iniciar câmera, acionar medição, visualizar 3D) de forma sequencial e coesa. - Implementar mecanismos de comunicação entre threads ou callbacks para assegurar que a execução em tempo real (loop de vídeo) não bloqueie a interface gráfica.

5.5 Testes Funcionais e de Usuário

- Realizar testes unitários e funcionais em cada parte do sistema (ex.: verificar se um ponto clicado no mapa de profundidade corresponde à profundidade esperada). - Validar o desempenho do sistema com imagens reais e streaming de webcam, confirmando a taxa de atualização e a qualidade do mapa de profundidade. - Conduzir testes de usabilidade básicos com potenciais usuários, para garantir que a interface e as ferramentas (carregar, medir, visualizar) sejam intuitivas.

5.6 Otimizações e Ajustes

- Otimizar parâmetros do modelo MiDaS e do pipeline (por exemplo, escolha de resolução de entrada) para equilibrar qualidade de profundidade versus desempenho. - Ajustar a interface para melhor resposta: eliminar gambiarras visuais, reduzir sobreposições de janelas. - Aplicar otimizações de código (vetorização, uso de GPU se disponível) onde necessário para acelerar o processamento de imagem em tempo real.

5.7 Documentação Técnica e Finalização

- Elaborar documentação técnica do sistema, explicando a arquitetura, o uso e os requisitos para execução. - Comentar o código-fonte, descrevendo as responsabilidades de cada módulo. - Preparar a versão final do software e validar todas as funcionalidades segundo os critérios definidos, fechando o ciclo de desenvolvimento.

6 Bibliografia

Referências

[1] Ranftl, R.; Lasinger, K.; Hafner, D.; Schindler, K.; Koltun, V. *Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer*. IEEE Trans. Pattern Anal. Mach. Intell., 2022.

[2] Birkl, R.; Wofk, D.; Müller, M. *MiDaS v3.1 – A Model Zoo for Robust Monocular Relative Depth Estimation*. arXiv:2307.14460, 2023.

[3] Zhou, Q.-Y.; Park, J.; Koltun, V. *Open3D: A Modern Library for 3D Data Processing*. arXiv:1801.09847, 2018.

[4] Eigen, D.; Puhrsch, C.; Fergus, R. *Depth Map Prediction from a Single Image using a Multi-Scale Deep Network*. Adv. Neural Inf. Process. Syst. 27 (NIPS), 2014.

[5] Hunter, J.D. *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, vol.9, no.3, pp.90-95, 2007.

[6] Bradski, G.; Kaehler, A. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.

[7] Lutz, M. *Programming Python*. 4th ed., O'Reilly Media, 2010. (Capítulo sobre Tkinter)