Análise e Estratégia

Avaliar criticamente o código existente

O pipeline foi organizado em etapas claras: (1) pose com YOLOv8, (2) extração de features front/side (ex.: `front_shoulder_px`, `side_elbow_span_px`), (3) normalização por altura (`front_height_px`) e (4) regressão (RidgeCV). Detectamos e corrigimos um problema de **contrato de features** na inferência: o modelo salvo esperava exatamente 10 colunas e a predição estava recebendo 11 (inclusão indevida de `front_height_px` após já ter normalizado). Foi criada uma função que **alinha as features de inferência** à lista `feature_names` salva no artefato (`joblib`) e alerta sobre faltantes/extras. Também padronizamos dtypes para `float32`, checamos NaN/inf, e normalizamos o fluxo de métricas. No pré-processamento visual, implementamos a **segmentação por limiar (Otsu)** e a montagem de **máscaras por partes** (peito, cintura, quadril, bíceps, antebraço, coxa, panturrilha), com suavização e "growth" paramétrico por parte. Para medidas, criamos linhas horizontais (tronco/pernas) e **verticais nos braços** (largura normal ao eixo do membro), reduzindo vieses de orientação.

Justificar escolhas técnicas e metodológicas

Mantivemos **YOLOv8-pose** pelo equilíbrio precisão/velocidade e por fornecer 17 keypoints padronizados (COCO). As features geométricas normalizadas por altura são simples, estáveis e explicáveis. Para o modelo, adotamos **RidgeCV** com `StandardScaler` dentro de `Pipeline`, pois: (i) é linear regularizado (reduz overfitting), (ii) tem busca de `alpha` embutida por CV e (iii) é facilmente reprodutível. Como baseline e "sanity check", treinamos **LinearRegression** puro. Validação: criamos função de **5-fold (StratifiedKFold)** estratificando por gênero (quando disponível), e **holdout 80/20** para leitura humana. Métricas: **MAE** e **RMSE** por alvo + **macro-média**. Os resultados mostraram **macro MAE ≈ 3.80** e **macro RMSE ≈ 4.81** em CV; no holdout, erros por medida, por exemplo: peito 5.23, cintura 6.85, abdômen 7.48, panturrilha 1.97 (todos em cm), indicando onde focar melhorias (cintura/abdômen). Houve igualdade entre baseline e RidgeCV — sinal de que as features atuais podem estar limitando ganhos; isso orienta os próximos passos de engenharia de atributos.

Definir plano de testes, validação e próximos passos

Implementamos avaliação programática: funções `evaluate_preds` (MAE/RMSE), `kfold_scores` (CV com agregação média±desvio), **relatórios CSV** por modelo/fold e **artefatos versionados** no `joblib` (modelo, `targets`, `feature_names`, resumo da CV). Para o código de visão, os módulos de máscara/medidas geram **linhas de mensuração** consistentes (braço vertical; tronco/pernas horizontais) e já exportam os valores em px e, opcionalmente, em **cm** via fator `HEIGHT_CM / front_height_px`. Próximos passos, ainda dentro do que está aqui: (1) reforçar **testes unitários (pytest)** para polígonos, máscaras e conversões px→cm; (2) **GroupKFold** (já sinalizado no código, comentado) se houver `person_id`, evitando vazamento entre folds; (3) enriquecer features com **larguras**

dos membros nos perfis definidos (as linhas vermelhas) e usar **side view** de forma mais ativa; (4) registrar ablações curtas (com/sem normalização, diferentes cortes de cintura, bandas alternativas nos braços/pernas).

Considerar escalabilidade, produção e manutenção futura

O treino gera artefatos reprodutíveis (`models/yolo_ridge.joblib`) contendo **modelo + metadados** (targets, nomes de features, sumários de CV). A inferência foi **endurecida**: valida o contrato de features, normaliza tipos e oferece fallback de conversão px→cm. Para produção, o que já está desenhado no código facilita: **módulos separados** (pose→features→modelo→pós), **checagens de sanidade** (NaN/inf, contagem de colunas), logs simples e relatórios salvos em `reports/`. Para manutenção, recomenda-se consolidar **checklists de PR** (padrões de estilo, cobertura mínima de testes, tempo/uso de memória por etapa), além de um notebook de “smoke test” que roda o pipeline em 1–2 amostras e verifica: (i) nomes/ordem das features; (ii) faixas plausíveis de saída; (iii) geração das linhas de medida sem texto. Por fim, mantemos um **fallback por keypoints** (já implementado nas medidas orientadas ao esqueleto) caso a segmentação do corpo falhe, garantindo robustez.