Perfeito — faz todo o sentido. Vou deixar um **resumo técnico, estruturado e "transportável"**, para poderes colar num novo chat e continuar exatamente do mesmo ponto, sem perder contexto.

📌 Resumo do Trabalho Realizado (IACV – Calibração e Retificação)

🎯 Objetivo Geral

Reconstruir a geometria 3D de um lançamento de basquetebol a partir de **uma única câmara não calibrada** (Samsung Galaxy S23), usando:

- deteção da bola (YOLOv8),
- anotação manual de pontos e retas,
- geometria projetiva (homografias, pontos de fuga),
- retificação métrica de planos (chão e plano vertical),
- posterior calibração da câmara.

1️⃣ Deteção e tracking da bola

- Foi treinado um modelo **YOLOv8** que:
    - deteta a bola em cada frame,
    - desenha a bounding box,
    - estima o centro da bola,
    - constrói a trajetória temporal.
- As coordenadas ((u,v)) do centro e da bbox são guardadas em CSV.

2️⃣ Definição do sistema de referência no mundo

- Foi criado um **referencial cartesiano fixo no campo**:
    - Origem: ponto O
    - Eixos:
        - (X): ao longo da largura do campo
        - (Y): profundidade
        - (Z): vertical
- Foram medidos **25 pontos no mundo real (em metros)**:
- O, A1–A4, B1–B5, C1–C4,
- P1–P3, Q1–Q2,

- V1–V5
- Estes pontos foram guardados em:
- outputs/world_points_coord.csv

3️⃣ Ferramenta interativa ("Court Wizard") para anotação na imagem

Conceito central

- Foi implementado um **state-driven interactive wizard** usando OpenCV.
- O estado é mantido num dicionário Python (state), atualizado progressivamente.

➡️ **Padrão usado**:
Não é um pattern formal "clássico", mas combina:

- *State Machine*
- *Command-style UI*
- *Model-View separation* (state vs draw)

É uma abordagem **comum e muito usada** em CV experimental e ferramentas de anotação.

Etapas sequenciais (sem modos livres)

STEP 1 – Pontos diretos

Selecionados manualmente na imagem:

A1, A2, A3, A4, V1, V2

STEP 2 – Retas manuais (2 cliques por reta)

Desenhadas em ordem fixa:

LP1, LP2, LP3

LQ1, LQ2, LQ3

LB1, LB2, LB3, LB4

Cada reta é armazenada como:

- segmento (p1, p2)
- reta homogénea (a,b,c)

4️⃣ Interseções diretas (pontos derivados)

Calculadas automaticamente:

O = LP1 × LQ1

C1 = LP1 × LQ2

C2 = LP1 × LQ3

C3 = LQ2 × LP3

C4 = LQ3 × LP3

B2 = LP2 × LB1

B3 = LP2 × LB2

B4 = LP2 × LB3

B5 = LP2 × LB4

5️⃣ Pontos de fuga (Vanishing Points)

- As famílias de paralelas no mundo são:

LP → VP_P

LQ → VP_Q

LB → VP_B

- Cada VP é estimado via **SVD**:
  [
  l_i^\top v = 0
  ]
- Implementado por:

estimate_vanishing_point_svd()

6️⃣ Retas projetadas (direções no mundo)

Reta projetada = ponto × ponto de fuga

Exemplo:

LQ4 = (LQ on V1)

7️⃣ Pontos derivados por projeção + interseção

Exemplos:

P1 = (LQ on A1) × LP1

P2 = (LQ on V1) × LP1

P3 = (LQ on A2) × LP1

V5 = (LQ on V1) × LP3

⚠️ O ponto V4 foi corrigido depois:

V4 = (LB on V3) × LQ4

→ **tem de ser calculado depois do conic fit**, porque depende de V3.

8️⃣ Conic fit no aro (ponto V3)

Estratégia

- Selecionar manualmente ~7–8 pontos no aro.
- Ajustar uma cónica geral:
  [
  ax^2 + bxy + cy^2 + dx + ey + f = 0
  ]

Implementação

- Método de **Fitzgibbon (least squares)**.
- O centro do círculo/ellipse é obtido resolvendo:
  [
  \begin{bmatrix}
  2a & b \
  b & 2c
  \end{bmatrix}
  \begin{bmatrix}
  x_c \ y_c
  \end{bmatrix}
  =
  \begin{bmatrix}
  -d \ -e
  \end{bmatrix}
  ]
- Resultado:

V3 = centro da cónica

9️⃣ Retificação métrica do plano do chão (z = 0)

Dados

- Correspondências:
- (X,Y) ↔ (u,v)

para pontos com z = 0.

Método

- **Homografia via DLT** (com normalização de Hartley).
- Opção de usar:

cv2.findHomography(..., RANSAC)

Resultado

- Homografia (H_{ground})
- Imagem retificada top-down (rectified_floor.png)
- Erro médio de reprojeção verificado (sanity check).

🔟 **Retificação do plano vertical (plano de lançamento)**

Plano

[
x = 6.58
]

Ideia

- Usar o **mesmo código da homografia**, mas:
    - filtrar pontos com x == 6.58,
    - usar coordenadas do mundo ((y,z)),
    - gerar uma imagem retificada do plano vertical.

➡️ Mesma pipeline, **apenas muda o filtro e os eixos**.

1️⃣1️⃣**Próximo passo lógico (não feito ainda)**

- Combinar:
    - homografia do chão,
    - homografia do plano vertical,
    - pontos de fuga,
- para:
    - obter a **matriz intrínseca da câmara (K)**,
    - reconstruir a trajetória 3D da bola,
    - validar com reprojeção.

🧠 **Conceitos-chave usados**

- Geometria projetiva
- Homografias (DLT)
- Pontos de fuga (SVD)
- Retificação métrica (não apenas afim)
- Conic fitting
- State machine interativa para anotação

Se quiseres, no **novo chat**, podes começar com algo como:

"Tenho um sistema interativo em Python/OpenCV que anota pontos e retas de um campo de basquete, calcula pontos de fuga, faz conic fit no aro e já consigo retificar o plano do chão via homografia DLT. Quero agora avançar para …"

Quando quiseres, eu acompanho-te exatamente daí 🚀