Research & Strategy: Multi-Camera Synchronization and Recording Protocols

1. Static Camera: Position and Consistency

Do I need the exact same position and angle for all records?

Yes, you must maintain a consistent camera pose.

- **Spatial Consistency:** Behavioral Cloning (BC) models like SmolVLA generally do not learn "camera intrinsics/extrinsics" on the fly from a small dataset (50 episodes). They learn pixel-to-action mappings. If you move the static camera between Episode 1 and Episode 2, the model will struggle to map the pixel coordinates of the target to the robot's physical position.

- **The "Fixed Frame" Assumption:** The static camera provides the "Allocentric" (global) coordinate frame. If this frame shifts, the robot's understanding of "forward" or "left" relative to the screen changes.
- **Practical Tip:** Use a tripod and mark the floor with tape where the tripod legs go. If you must move it, try to realign it as precisely as possible before recording the next batch.

Timeshifting USB vs. Wifi Frames (Critical)

Yes, you must time-shift the USB frames.
This is a classic "Sensor Fusion" problem. Your system has two distinct latencies:

1. **Wifi Camera (Onboard):** High latency (~200ms).
2. **USB Camera (Static):** Low latency (~30ms).

If you naïvely record the "latest available frame" from both cameras at t_now, you create a **temporal mismatch**.

- The **Onboard View** shows the world as it was 200ms ago.
- The **Static View** shows the world as it was 30ms ago.
- *Result:* The static camera might show the robot *already starting to turn*, while the onboard camera shows it *still moving straight*. This contradictory data confuses the training process (conflicting gradients).

The Solution: The "Lookback" Buffer
You need to align the USB frames to the Wifi frames' "capture time," not their "arrival time."

1. **Calculate Offset:** Use the "Flash Test" described previously to measure the *differential* latency.
   - $\Delta t \approx Latency_{Wifi} - Latency_{USB}$ (e.g., $200ms - 30ms =

170ms$).
2. **Buffer the USB Stream:** Maintain a circular buffer (queue) of the last ~1 second of USB frames, timestamped on arrival.
3. **Synchronization Logic:**
   When a Wifi frame arrives at t_now:
   - Calculated "True Capture Time" $\approx t_{now} - Latency_{Wifi}$.
   - Look into your USB buffer.
   - Select the frame with timestamp closest to $(t_{now} - Latency_{Wifi})$.
   - *Effectively:* You are deliberately delaying the USB feed by ~170ms to match the "sluggish" Wifi feed. Both views will now be "200ms old," but they will be **consistent with each other**.

---

2. Recording Frequency

How often should I record?

Record at a fixed Control Frequency, independent of camera frame arrival.
Do not trigger recording "when a frame arrives" because Wifi frames arrive with **jitter** (irregular intervals). Training algorithms (Transformers/Diffusion) expect a constant time-step (dt) between actions (e.g., 0.1s, 0.02s).

Recommended Setup: The 30Hz Heartbeat
Most VLA models (including LeRobot's implementations) are standardizing around **30 Hz** (30 frames per second).
1. **The Control Loop (High Frequency):**
   Run your joystick-reading and command-sending loop as fast as possible (e.g., 50Hz or 100Hz) to keep the robot feeling responsive to your hand.
2. **The Data Logger (Fixed 30Hz):**
   Run a separate timer/thread that triggers exactly 30 times per second.
   - **At every trigger:**
     1. Grab the *latest* Command (V_x, V_y, ω_z).
     2. Grab the *latest* Wifi Frame.
     3. Grab the *buffered* USB Frame (aligned as described above).
     4. Save this tuple to your dataset.
3. **Duplicate Frames are Fine:** If the Wifi camera lags and you record the same frame twice in a row, that is acceptable (and common). It is much better than having variable time-steps.

Summary Implementation Logic

```python
# Pseudo-code for aligned recording
BUFFER_SIZE = 30  # store 1 second of usb frames
usb_buffer = collections.deque(maxlen=BUFFER_SIZE)
offset_seconds = 0.17  # Determined via your latency test

def recording_loop_30hz():
    while recording:
        # 1. Get latest data
        cmd = joystick.get_command()
        wifi_frame = robot.get_latest_frame() # Arrived just now

        # 2. Store current USB frame with receipt time
        usb_frame_now = usb_cam.read()
        usb_buffer.append( (time.time(), usb_frame_now) )

        # 3. Find the ALIGNED USB frame
        # We want the USB frame captured at the same time the Wifi frame was captured.
        # Wifi frame capture time approx = time.time() - wifi_latency
        target_time = time.time() - wifi_latency

        # Search buffer for frame closest to target_time
        aligned_usb_frame = find_closest(usb_buffer, target_time)

        # 4. Save
        save_to_disk(cmd, wifi_frame, aligned_usb_frame)

        time.sleep(1/30.0) # Maintain 30Hz
```

References

Comparison of single-view vs. multi-view for behavioral cloning generalization. Timestamp synchronization strategies for multi-sensor robotics. LeRobot/ALOHA training guidelines regarding fixed-frequency data collection. Latency measurements in networked robotic systems.