ChatGPT

Deep Learning Approaches to UAV Pose Estimation

Deep learning has been applied to drone localization as an alternative to classical SLAM/pose-graph methods. Pioneering works have shown that convolutional neural networks (CNNs) and recurrent nets can regress full 6-DoF pose directly from sensor inputs. For example, **PoseNet** uses a single RGB image to directly regress 6-DoF camera pose with a deep CNN [1], bypassing any feature matching or graph optimization. Similarly, **DeepVO** treats visual odometry as a sequence-to-sequence problem: it inputs consecutive monocular frames into a CNN+LSTM network to predict the inter-frame motion (learning scale implicitly) [2]. These end-to-end models learn image features and temporal dynamics jointly, showing that scale and orientation can be recovered from training data without hand-engineered features.

Figure 1: Example fusion architecture for visual–inertial odometry [3]. A CNN (e.g. ResNet) extracts features from each RGB frame, while a small LSTM encodes IMU sequences; their outputs are concatenated and fed into a core LSTM that predicts the current 6-DoF pose. This end-to-end network learns feature extraction and temporal fusion jointly.

More recently, **visual–inertial** deep networks explicitly fuse camera and IMU data for UAV odometry. For instance, *VINet* [4] uses a CNN on images plus an LSTM on raw IMU (accelerometer and gyro) readings, combining their latent features in an LSTM that outputs 3D translation and orientation. Baldini *et al.* [3] extended this idea with a two-stream model: a ResNet-based CNN encodes the RGB image while a second LSTM encodes a window of IMU measurements; a larger "core" LSTM then combines these to regress the drone's position and quaternion orientation in real time. Such end-to-end VIO networks are typically trained on datasets like EuRoC MAV (stereo + IMU + ground truth) [5], and have outperformed classical VIO baselines on those benchmarks. Another approach, *VIIONet* [6], converts IMU time-series into an "inertial image" (after denoising) and stacks it with optical-flow frames; features are extracted via Inception-v3 CNNs from the image and inertial channels, and fused (via Gaussian Process regression) to predict 6-DoF motion. In all these fusion models, the deep network **learns its own features** for motion rather than relying on hand-crafted descriptors [7].

Beyond RGB and IMU, incorporating **depth or range sensors** can improve scale and robustness. Liu *et al.* [8] propose an *unsupervised* RGB-D odometry network: two parallel CNN streams (one for RGB, one for depth) feed into a pose regressor. The depth branch supplies absolute scale, allowing the network to recover real-world motion without ground-truth labels. They train it self-supervised on KITTI, achieving lower translation/rotation errors than monocular VO. Analogously, one could feed depth images into a CNN (stacking with RGB or separate-stream) and fuse with IMU via an RNN. (Indeed, DeepFactors and CNN-SLAM replace photometric mapping with learned depth priors, though those still use pose-graph back-ends.) Our UAV has *depth cameras*, so a dual-stream CNN (RGB+D) as in [55], fused with IMU features via an LSTM, would be a natural architecture.

Deep fusion has also been applied to other range sensors. For example, **DeepLIO** [9] fuses LiDAR scans with IMU: one network processes the LiDAR point cloud into features, another ingests IMU, and a learned fusion predicts 6-DoF motion. While LiDAR is denser than sonar, similar ideas apply. In principle, the drone's omnidirectional *sonar* array (giving distance to walls) could be treated like a set of 1D range channels. One could encode sonar readings (e.g. a 1D convolution or MLP on the sonar array at each timestep) and fuse that feature alongside the image and IMU encoders. Such “range images” would help in low-texture scenes. To our knowledge, few published networks explicitly use sonar in odometry; most multimodal work uses RGB, depth or LiDAR. But sensor fusion networks can be extended: concatenate or attentively fuse latent features from each modality, letting the net learn to weight them.

In summary, key deep-learning works include: *PoseNet* [1] (CNN for single-image pose), *DeepVO* [2] (CNN+RNN on video), *VINet* [4] and later variants [3] [6] for visual–inertial fusion, and *RGB-D VO nets* [8] that learn absolute scale from depth. These models typically output a 7-vector (x,y,z translation + quaternion) every few frames, achieving real-time (e.g. 10–60 Hz) inference depending on network size. Importantly, all cited methods estimate full **6-DoF** camera motion (as required) and are designed or evaluated for real-time or online use.

Multimodal Data and Additional Sensors

Your drone already provides RGB images, depth maps, sonar (range) and IMU. To improve accuracy, consider adding or carefully logging other helpful data:

- **Ground truth / calibration**: For supervised training, you need accurate pose ground truth (e.g. via motion-capture or a high-quality SLAM reference). Also record camera intrinsics, extrinsics (to IMU), and timing offsets. IMU biases and scale factors should be calibrated offline.
- **Orientation reference**: A magnetometer (3-axis compass) can give absolute yaw reference, useful if the network can ingest it as extra input. A barometric altimeter provides altitude data to correct vertical drift. These are cheap on most UAVs.
- **Global position**: If outdoors, logging GPS position (even if noisy) can help train the network on large-scale drift patterns, or serve as a supervisory signal if using GPS-denied methods.
- **Additional vision**: If available, stereo cameras or wide FOV cameras add more viewpoint information. Even a second camera (RGB or infrared) can feed parallel CNNs for multi-view fusion.
- **Environment cues**: If in structured environments, consider using fiducial markers (AprilTags) at test time for occasional fixes. These are not learned, but can serve as triggers for recalibration or loop-closure.
- **Velocities**: If you have odometry (e.g. propeller RPM or optical flow sensors), these can be additional inputs or used to generate pseudo-labels during unsupervised training.

In general, **more diverse modalities** give the network redundancy and scale observability [7]. For example, depth from sonar/lidar directly fixes scale like stereo cameras do. Each extra modality simply becomes another input branch: e.g. convert sonar readings into a depth image or use a small CNN/MLP on the raw range vector, then fuse that feature with the visual ones. The key is synchronizing all sensor streams tightly and feeding them into a common network.

Architectural Recommendations

Based on the literature, a **lightweight CNN+RNN fusion network** is a good candidate. For instance:

- **CNN backbone**: Use a compact CNN (ResNet-18, MobileNetV2, or Tiny-ResNet) to encode the RGB image into a feature vector [3] [1]. Include a parallel branch that processes the depth map – either

by concatenating depth as a 4th input channel or via a separate small CNN. This yields an "RGB-D feature" at each timestep.
- **Temporal model**: Feed the image features into a recurrent layer (LSTM or GRU) or a temporal convolution to integrate motion over time [3] [4]. Likewise, feed a short window of IMU readings (linear accel + gyro) into a second LSTM that outputs an "inertial feature vector." Then concatenate the visual and inertial vectors and feed into a **core LSTM/GRU** that regresses the current pose. This is essentially the Baldini *et al.* architecture [3]. For real-time, keep the LSTMs small (e.g. 256–512 units).
- **Sensor fusion**: Instead of simple concatenation, one could experiment with **attention or gating** to let the network weight each modality adaptively [6]. For example, VIIONet showed that boosting visual features with inertial "image" improved accuracy. Learned fusion (via a small MLP after concatenation) can also work.
- **Output representation**: Regress a 7D vector (3 pos + 4-quat) or 6D (pos + rotation vector). Normalize quaternions or use Lie-algebra losses. Many works (PoseNet, DeepVO) train with a loss balancing position vs orientation (possibly learnable weighting).

For **efficiency**, prune the model: use depthwise separable convolutions, quantize or use TensorRT, or distill into a smaller student network. Some works run on embedded GPUs at 30–60 Hz by using e.g. ResNet-18 and a single LSTM. If computation is extremely limited, you could even try 2D convolutional LSTMs (ConvLSTM) on stacks of frames to replace separate CNN+RNN. Transformers are emerging for sequence modeling, but LSTM/GRU remains more parameter-efficient for real-time use.

Deployment Considerations

Finally, ensure real-time performance by **reducing input rate** if needed (e.g. process every 2nd frame). Use fixed-size window updates (sliding window LSTM) rather than very long sequences. For training data, if ground truth is scarce, consider *self-supervised* strategies: e.g. train a photometric reconstruction loss between frames (as in unsupervised depth & pose networks) or use the known sonar/depth to supervise scale.

In summary, prior work shows that end-to-end learned odometry can replace classical SLAM: CNN+RNN architectures (like VINet [4] or Baldini's model [3]) fuse multi-modal inputs and achieve 6-DoF, real-time pose estimation. You should record synchronized, calibrated RGB, depth and inertial data (plus any extra like magnetometer or GPS) and train a small fusion network. This network could mirror published designs (e.g. ResNet18 + LSTMs) but pared down for your hardware. With careful training (possibly mixing supervised and self-supervised losses) and calibration, a deployable deep pose estimator is feasible following these academic precedents [1] [3].

Sources: Research papers on CNN/LSTM-based VIO and 6-DoF pose regression [1] [2] [4] [3] [6] [8] [9]. Each demonstrates end-to-end pose estimation (often 6-DoF) from images, depth, IMU, etc.

---

[1] [1505.07427] PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization
https://arxiv.org/abs/1505.07427

[2] cs.ox.ac.uk
https://www.cs.ox.ac.uk/files/9026/DeepVO.pdf

[3] [5] [7] [1912.04527] Learning Pose Estimation for UAV Autonomous Navigation and Landing Using Visual-Inertial Sensor Data
https://ar5iv.labs.arxiv.org/html/1912.04527

[4] VINet: Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem
https://arxiv.org/pdf/1701.08376

[6] Visual-Inertial Image-Odometry Network (VIIONet): A Gaussian process regression-based deep architecture proposal for UAV pose estimation | Request PDF
https://www.researchgate.net/publication/359393175_Visual-Inertial_Image-Odometry_Network_VIIONet_A_Gaussian_process_regression-based_deep_architecture_proposal_for_UAV_pose_estimation

[8] Unsupervised Deep Learning-Based RGB-D Visual Odometry
https://www.mdpi.com/2076-3417/10/16/5426

[9] isprs-annals.copernicus.org
https://isprs-annals.copernicus.org/articles/V-1-2021/47/2021/isprs-annals-V-1-2021-47-2021.pdf