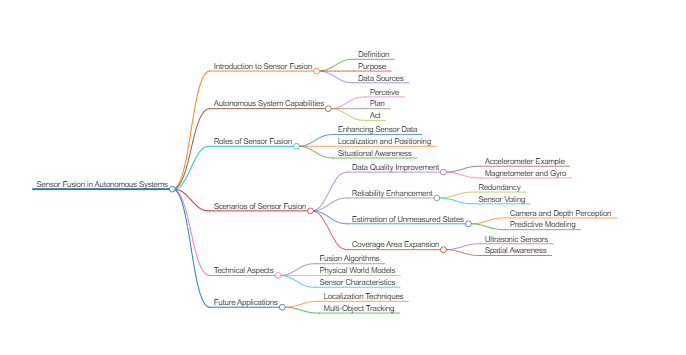
TASK 30

## DEEP DAS

Sensor fusion is the process of combining data from multiple sensors to achieve a more accurate, consistent, and dependable understanding of the system and its environment. It is integral to the design of autonomous systems such as self-driving cars, where it helps in improving the quality of data by reducing noise and uncertainty. The script illustrates sensor fusion as a method to enhance the system's ability to perceive its surroundings and make informed decisions.



#### 1. Improving Data Quality

One of the primary benefits of sensor fusion is its ability to improve data quality. When integrating measurements from different types of sensors, uncorrelated noise from each sensor can be averaged out, leading to more accurate and reliable data. For instance, consider the use of accelerometers in an autonomous system. Each accelerometer may have its own noise characteristics, but by fusing the data from multiple accelerometers, the system can average out the noise and obtain a more accurate representation of acceleration.

**Example:** In an autonomous vehicle, accelerometers are used to measure acceleration. Each accelerometer may introduce some noise due to vibrations, mechanical imperfections, or other factors. By using sensor fusion to combine data from multiple accelerometers, the system can reduce the overall noise, providing a more precise measurement of the vehicle's acceleration.

#### 2. Reducing Noise

Sensor fusion also helps in significantly reducing noise in sensor readings. By combining data from various sensors, the system can filter out random noise that might affect individual sensors. This is particularly important in environments with a lot of interference or when sensors have varying levels of accuracy.

**Example:** In self-driving cars, lidar and radar sensors are used for object detection. Each sensor might be affected by different types of noise—lidar might struggle with rain, while radar might be affected by metal objects. By fusing data from both sensors, the system can reduce the impact of these noises and provide a clearer picture of the surroundings.

#### 3. Increasing Reliability

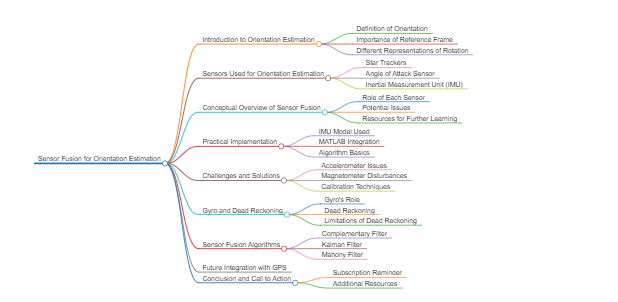
By integrating measurements from multiple sensors, sensor fusion enhances the overall reliability of the system. This is especially valuable in scenarios where individual sensors may fail or give inconsistent readings. The redundancy provided by multiple sensors ensures that the system can continue to operate reliably even if one or more sensors experience issues.

**Example:** In an IoT network for home automation, multiple temperature sensors might be deployed in different rooms. If one sensor fails or gives an outlier reading due to malfunction, the system can rely on data from other sensors to maintain an accurate temperature profile of the house.

#### 4. Measuring Unmeasured States

Sensor fusion enables the estimation of unmeasured states by leveraging data from multiple sensors. This is crucial for applications where direct measurement is challenging or impossible. By combining different types of sensor data, the system can infer states that are not directly observable.

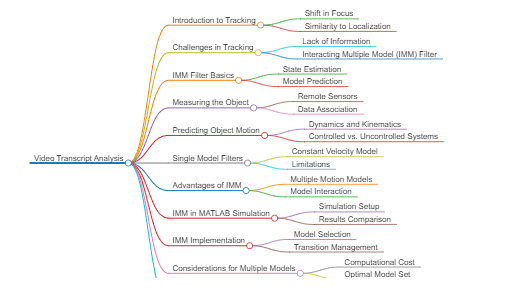
**Example:** In autonomous navigation, GPS provides position data, while accelerometers and gyroscopes provide information on movement and orientation. By fusing data from these sensors, the system can accurately estimate the vehicle's trajectory and orientation, even in areas where GPS signals are weak or unavailable.



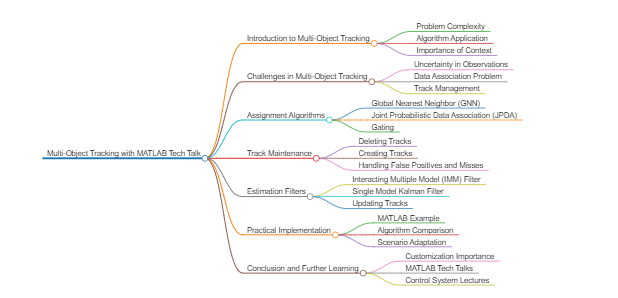
* Orientation is determined relative to a reference frame using various sensors like star trackers and angle of attack sensors.
* The focus is on using a magnetometer, accelerometer, and gyro found in modern phones and autonomous systems.
* The aim is to conceptualise the sensor fusion system without developing a full inertial measurement system.
* Different methods to represent rotation, such as roll, pitch, yaw, and quaternion, are mentioned.
* The orientation of a stationary phone can be estimated using a magnetometer and accelerometer.
* The magnetic field's direction is influenced by the gravity direction, requiring cross-products to determine north.
* An implementation example using an IMU (MP u 9250) connected to MATLAB is demonstrated.
* We visualise orientation and updates using a MATLAB viewer, performing cross products and building a direction cosine matrix.
* Problems with simple implementations, such as the effect of linear accelerations on the accelerometer's readings, are highlighted.
* Magnetometer readings can be corrupted by magnetic disturbances, which can be mitigated through calibration.
* Hard and soft iron sources affect magnetometer readings and can be calibrated out if they rotate with the system.
* Calibration involves measuring distortion and applying a transformation matrix to correct measurements.
* Dead reckoning, using gyro measurements, is introduced as a method to estimate orientation for rotating objects.
* Integration of gyro measurements can lead to drift due to sensor bias and noise.
* Sensor fusion algorithms, such as complementary and Kalman filters, are used to combine estimates from different sensors.



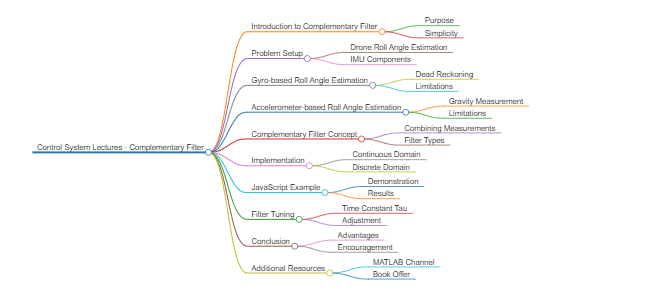
* The video discusses sensor fusion for positioning and localization, focusing on the integration of IMU and GPS sensors.
* IMU sensors are used to estimate an object's orientation, and GPS sensors measure position and velocity for enhanced estimation.
* The fusion algorithm structure is explained, emphasizing the visual contribution of each sensor to the final solution.
* GPS is sufficient for some applications with low accuracy requirements and slow motion systems.
* High-accuracy and high-update-rate applications, such as drones, may require additional sensors like IMU to complement GPS.
* MATLAB's sensor fusion and tracking toolbox is demonstrated with an example of pose estimation from asynchronous sensors.
* The example shows how the fusion algorithm uses GPS, gyro, and magnetometer data to estimate orientation and position.
* The script allows for adjusting sensor sample rates and removing sensors to observe the impact on estimation.
* Removing all sensors except GPS results in significant orientation error but reasonable position accuracy.
* Adding IMU sensors to the GPS-only setup shows minor improvements in position estimation for slow movements.
* For faster movements and rapid changes in velocity, the IMU's high update rate is crucial for accurate estimation.
* The video explains the continuous-discrete extended Kalman filter used in the fusion algorithm, which handles asynchronous sensor measurements.
* The filter's state vector includes 28 elements, estimating not only the main states but also sensor biases.
* Estimating sensor bias is crucial due to their drift over time, which can affect the accuracy of the estimation.
* The video discusses the importance of filter initialization and its impact on convergence and accuracy.
* The predict-correct process of the EKF is explained, highlighting how it uses predictions and measurements to refine state estimation.
* The video concludes by encouraging viewers to experiment with the example and explore the code for a deeper understanding.



* This introduces a shift in focus from estimating the state of one's own system to estimating the state of a remote object, transitioning from positioning and localization to single object tracking.
* Tracking an object involves determining its state such as position or velocity by integrating sensor data with models, which is more challenging with less information available.
* To address tracking difficulties, the video suggests upgrading a single model estimation filter to an Interacting Multiple Model (IMM) filter.
* The IMM filter is introduced as an effective approach for tracking uncertain objects by building intuition through simulation results.
* The superiority of the IMM filter over a single model filter in tracking a maneuvering object through comparative simulation results.
* Estimation filters operate on a predict-then-correct mechanism using system models and sensor measurements to estimate the state of a system or object.
* The challenge in tracking is the difficulty in predicting the future state of an uncontrolled object, unlike estimating the state of one's own system.
* This explains the importance of system dynamics, kinematics, and control inputs in the prediction step of the filter process.
* The concept of process noise in filters is discussed, which accounts for unknown inputs and model uncertainty, affecting the prediction reliability.
* The difference between cooperative and uncooperative tracking is highlighted, with cooperative objects sharing information that aids in prediction.
* This illustrates the limitations of a single model filter when the tracked object's motion does not match the model's assumptions.
* Increasing process noise in a filter is shown to improve tracking during motion transitions but at the cost of performance during consistent motion.
* The IMM filter is explained as running multiple simultaneous estimation filters, each with a different prediction model and process noise.
* The IMM filter's interaction between models after a measurement is described, which improves individual filter performance by blending estimates.
* The importance of selecting an optimal number of models for the IMM filter is emphasized to balance computational cost and tracking performance.



* Introduction to expanding single object tracking to multiple objects using algorithms like IMM, with considerations for additional complexities in multi-object scenarios.
* The importance of understanding the art of selecting or developing algorithms for multi-object tracking due to the variability in problem specifics.
* Challenges in multi-object tracking arising from uncertainty in object observations and predictions of their paths.
* The problem of data association in matching observations to tracked objects without unique identifiers.
* Situations where objects are close, leading to difficulties in data association due to overlapping measurements.
* The dynamic nature of multi-object tracking, including the creation and deletion of tracks based on observations.
* Strategies for track maintenance to avoid premature creation or deletion of tracks due to false positives or sensor failures.
* Different assignment algorithms for data association, including GNN and JPDA, and their respective advantages and limitations.
* The concept of gating to ignore observations outside a defined region for each track, enhancing computational efficiency.
* The use of Mahalanobis distance in GNN for matching observations to tracks based on probabilistic distance rather than Euclidean distance.
* JPDA's approach to data association by making a weighted combination of all neighboring observations.
* Criteria for creating tracks, such as maintaining tentative tracks and confirming them based on detection frequency.
* Criteria for deleting tracks, using parameters like P times in R updates to avoid premature deletion.
* The process of updating tracked objects with estimation filters after assigning observations to tracks.
* MATLAB example demonstrating the performance of GNN and JPDA algorithms in tracking two maneuvering objects.
* Comparison of GNN and JPDA algorithms, showing the benefits and drawbacks of each in different tracking scenarios.
* The necessity of assessing requirements before implementing complex tracking algorithms to balance accuracy and complexity.



Introduction to the complimentary filter, a simple yet effective method for blending measurements from two different sensors.

The practical application of the complimentary filter in estimating the roll angle of a drone using its IMU.

Explanation of how to measure the roll angle using gyro and accelerometer readings from an IMU.

Dead reckoning method for estimating roll angle by integrating angular rate measurements over time.

Limitations of dead reckoning due to cumulative errors from noise and bias in gyro measurements.

Utilizing the accelerometer to measure the roll angle based on the gravity vector pointing downwards.

Challenges in using accelerometer for roll angle estimation due to noise and external accelerations.

The stability of the accelerometer for long-term roll angle estimation as gravity vector remains constant.

The concept of combining short-term accuracy of the gyro with the long-term stability of the accelerometer.

Technical explanation of the complimentary filter implementation using low-pass and high-pass filters.

Adjusting the time constant tau to tune the filter's balance between short-term and long-term accuracy.

Discrete implementation of the complimentary filter in software for digital control algorithms.

Algorithm explanation for combining gyro and accelerometer measurements to estimate the roll angle.

Demonstration of the complimentary filter's performance in maintaining the true down direction compared to individual sensors.

Mathematical breakdown of the discrete complimentary filter implementation and its similarity to the continuous domain approach.

The simplicity and computational efficiency of using the complimentary filter for control system applications.

Invitation for questions, comments, and further exploration of drone control and simulation topics.

Acknowledgment of Patreon supporters and offer to download a digital copy of the book on control theory fundamentals.