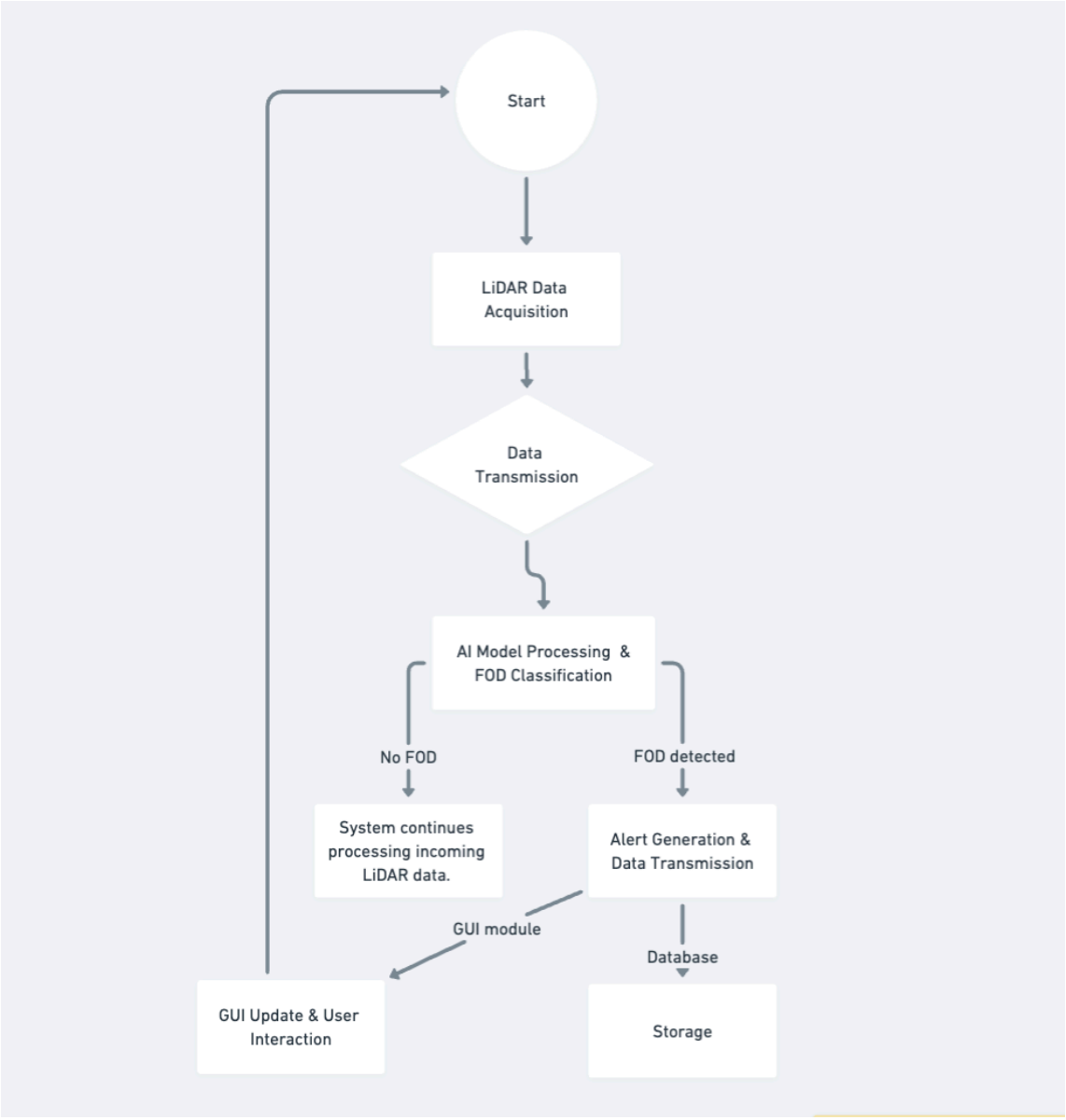
AI-powered FOD Detection System
For Naval Air Stations and Aircraft Carriers

1. Introduction

Foreign Object Debris (FOD) poses a significant threat to aircraft safety on Naval Air Stations and Aircraft Carriers. Existing methods like manual inspections and sweepers are labor-intensive, unreliable, and prone to human error. This document proposes an AI-powered FOD detection system mounted on a vehicle platform to address these limitations.



2. FOD Detection Sensors

Several sensor technologies can be employed for FOD detection. Here's a breakdown with reasoning for the chosen solution:

- **LiDAR (Light Detection and Ranging):** LiDAR excels at creating high-resolution 3D maps of the environment. It can effectively detect objects of various sizes and materials, including small FOD like screws and rivets (meeting the 3mm minimum requirement). Additionally, LiDAR functions well in various lighting conditions, making it suitable for day and night operations.
- **GPS, and Optical cameras**
- **mmWave Radar Sensors / 94.5 GHz FMCW radar**
- **Electro-Optical Cameras (EO Cameras):** While EO cameras provide high-resolution visual data, their effectiveness diminishes in low-light conditions. They might struggle to differentiate between FOD and runway markings/shadows.
- **Infrared Cameras (IR Cameras):** IR cameras can detect heat signatures, potentially identifying hot debris like burning objects. However, they wouldn't be ideal for all FOD types and might be susceptible to false positives due to engine heat on the runway.

Therefore, **LiDAR** is the optimal choice due to its superior 3D mapping capabilities, small object detection, and all-weather functionality.Additionally, the model's capability to detect very small FOD with high accuracy within a short time frame

Optimal Choice : **mmWave Radar Sensors / 94.5 GHz FMCW radar with Optical cameras**

This system has been found to be highly efficient, weather resilient, and cost-effective.

Reference :
https://www.proquest.com/openview/ef6a66b253dfeba73ac767cd9911fc9f/1?pq-origsite=gscholar&cbl=2045096

3. AI Model for FOD Classification

Processing LiDAR data for FOD classification is well-suited for an AI model. Here's why:

- **Real-time Detection:** An AI model can analyze LiDAR data in real-time, enabling immediate alerts for FOD presence.
- **Reduced Human Error:** AI eliminates human fatigue and subjectivity, leading to more

consistent and reliable FOD detection.

- **Learning and Adaptability:** The AI model can be trained on a vast dataset of FOD images, enhancing its ability to identify various debris types over time.

Deep Dive into AI Model for FOD Classification: The AI model plays a crucial role in classifying objects detected by the LiDAR sensor as FOD or not, further identifying the specific FOD type. Here's a breakdown of the model selection and training data considerations:

3.1. Model Selection:

Deep Learning Convolutional Neural Networks (CNNs):

Highly suitable for image recognition tasks like FOD classification due to their ability to extract features from LiDAR point cloud data. Popular choices include:

- ➔ **YOLO (You Only Look Once):** A real-time object detection model that can potentially be adapted for FOD classification on LiDAR data.
- ➔ **VGG16:** A well-established pre-trained CNN architecture that can be fine-tuned for FOD classification.
- ➔ **ResNet:** Another powerful pre-trained CNN known for its depth and performance in image recognition tasks.

3.2. Training Data Considerations:

The success of the AI model hinges on the quality and diversity of the training data. Here's what to consider:

- **Data Types:**
  - **LiDAR Point Clouds:** 3D representations of the runway environment captured by the LiDAR sensor. These point clouds should be labeled with bounding boxes around FOD objects, indicating their location and size.
  - **Images:** High-resolution images of various FOD types on runways can be used to augment the training data, especially for models like VGG16 and ResNet.
- **Data Diversity:** The training data should encompass a wide range of FOD types, sizes, materials, and lighting conditions to ensure the model generalizes well to real-world scenarios. Here are some factors to include:
    - Different types of objects: Train on a broad spectrum of FOD types e.g. bolts, birds and others as stated by FAA.
    - Many examples: There should be multiple images with annotated instances for each object type.
    - Realistic settings: Lighting conditions (bright, dim), weather (dry, wet) and backgrounds can vary.
    - More than one view point: Images from close-up and wide zoom levels will also be helpful in accommodating various object scales.

        - A big dataset that is diverse enough: A huge number of diverse data prevents overfitting, thereby enhancing the model's performance.

        - Complexities related to scenarios: These include occlusion, clutter, partial visibility and object motion for real-world robustness.https://arxiv.org/pdf/2110.03072.pdf

- **Data Augmentation Techniques:** Techniques like random rotations, scaling, and cropping can be applied to artificially increase the size and diversity of the training data, improving the model's ability to handle variations in real-world FOD appearances.

3.3 Training Process:

- The chosen deep learning framework (e.g., TensorFlow, PyTorch) will be used to train the model on the prepared dataset.
- The training process involves feeding the labeled data into the model and adjusting its internal parameters to minimize classification errors.
- Techniques like dropout and regularization can be employed to prevent overfitting, where the model performs well on the training data but poorly on unseen data.

3.4 Model Evaluation:

- Once trained, the model's performance is evaluated on a separate test dataset that wasn't used for training. This helps assess its generalization ability and identify potential weaknesses.
- Metrics like accuracy, precision, and recall are used to measure the model's effectiveness in classifying FOD objects.

3.5 Continuous Learning:

- As new FOD types or variations are encountered, the model can be continually improved through a process called transfer learning. Here, the pre-trained model is fine-tuned on new data specific to these FOD types, enhancing its overall classification capabilities.
- By carefully selecting an appropriate deep learning model, gathering a diverse and well-labeled training dataset, and implementing a robust training and evaluation process, you can develop a highly accurate AI model for FOD classification in the AI-powered FOD detection system.

4. Graphical User Interface (GUI)

Features and Functionalities:

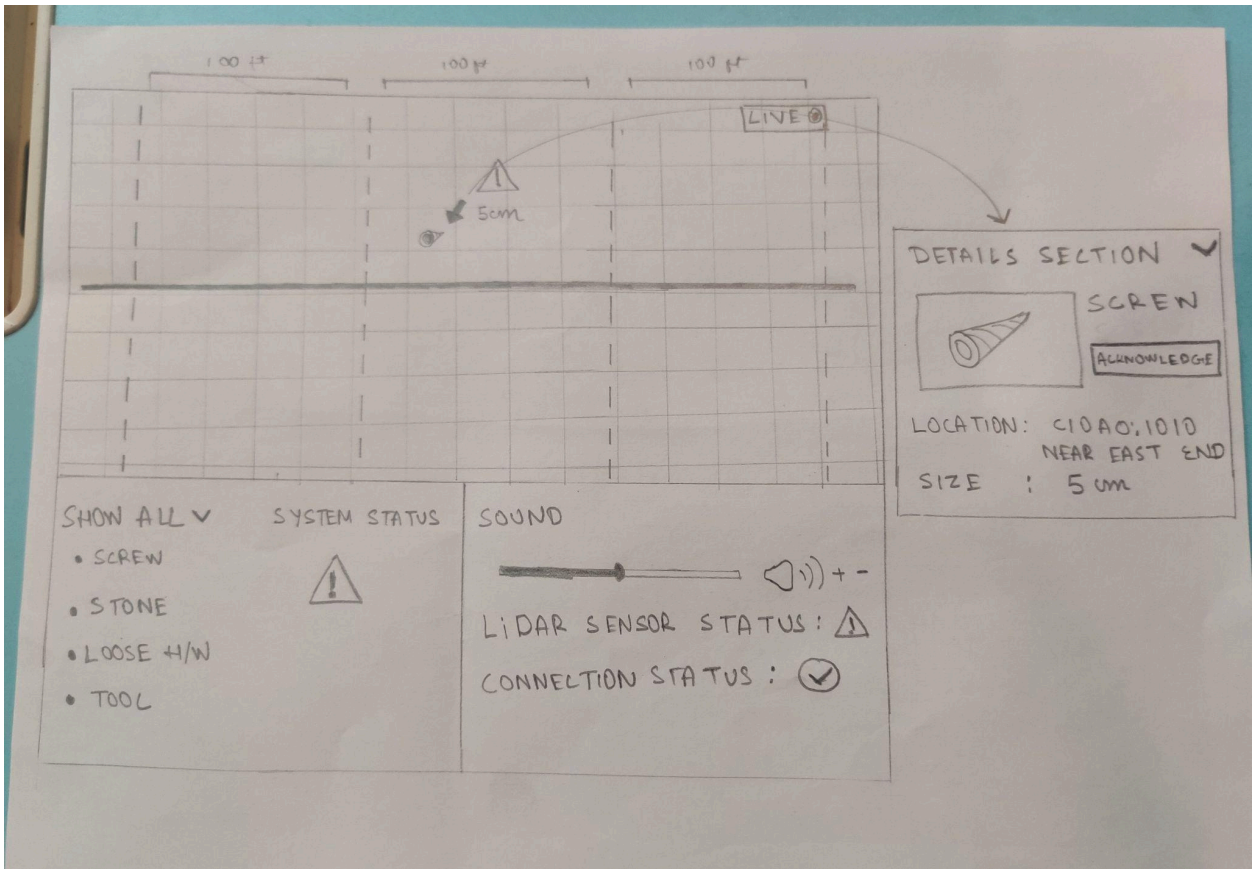
- **Real-time Runway Map:** A map displaying the scanned runway area with identified

FOD highlighted.

- **FOD Classification:** Displaying the type and estimated size of detected FOD.
- **Alert System:** Audible and visual alerts for FOD presence along with location data.
- **FOD History Log:** Recording timestamps, location, and type of detected FOD for future reference.
- **System Settings:** Options for configuring system parameters like alert thresholds.

4.1 Wireframe Design:

AI-powered FOD Detection System GUI, incorporating an industrial design aesthetic:



Runway Map (Top Section - 2/3 of the Screen):

- **Background:**
  - Utilize a high-resolution, black and white satellite image or texture of the runway for a realistic representation.
  - Ensure clear visibility of runway markings:
    - Centerline - Thick white solid line.
    - Hash marks - Short white lines perpendicular to the centerline, typically every 50 or 100 feet.
    - Runway numbers/designations - Painted markings on each runway end, clearly visible from the pilot's perspective.

- **Grid System:**
  - Implement a subtle light gray grid overlay on the map for distance reference.
  - Number markers along the edges (every 100 meters/feet) for easy location identification.
- **FOD Icons and Overlays:**
  - Represent detected FOD with colored, shaded icons that are easily distinguishable from the background:
    - **High Priority (Immediate Threat):** Bright red triangle with a blinking white border to draw immediate attention.
    - **Medium Priority (Potential Threat):** Yellow diamond.
    - **Low Priority (Caution):** Orange circle.
  - Include a small white numeric label next to the icon, indicating the estimated size of the FOD (e.g., "5cm").
  - Upon selecting an FOD icon on the map (optional):
    - Highlight the chosen FOD icon with a brighter color or pulsating effect for better visibility.
    - Trigger the Alert Details Panel in the middle section to display information about the selected FOD.
- **Map Controls (Optional - Top Right Corner):**
  - Zoom in (+) and zoom out (-) buttons for focusing on specific runway sections.
  - Toggle switch to display a live camera feed overlaying the map (if a camera is integrated with the system). Consider a transparency adjustment slider to optimize readability of FOD icons underneath the camera feed.

Alert Details Panel (Middle Section - Optional - Collapsible):

- This accordion-style panel expands to reveal details when a specific FOD on the map is clicked.
- Content:
  - **FOD Classification:**
    - Icon matching the one displayed on the map for the selected FOD.
    - Clear text label for the FOD type (e.g., "Screw", "Tool", "Bird").
  - **Size:** Estimated size of the FOD, displayed below the classification icon.
  - **Time Detected:** Timestamp of when the FOD was first detected by the system.
  - **Location:**
    - Specific GPS coordinates of the FOD.
    - Runway section reference (e.g., "East end, near taxiway intersection").
  - **Image (Optional):**
    - If the system captures an image of the FOD, display it here for confirmation.
  - **Acknowledge Button:** A red button labeled "Acknowledge" that allows the user to confirm they have seen the alert and take appropriate action. Upon clicking, the button grays out or changes text to "Acknowledged" to avoid accidental reactivation.

System Controls and Information (Bottom Section - 1/3 of the Screen):

- **Left Side:**
  - **FOD Type Filter:**
    - Dropdown menu listing all possible FOD types.
    - Selecting a type filters the displayed icons on the map to only show that specific FOD

type.
    - An additional option "Show All" allows returning to the unfiltered view.
  - **System Status:**
    - Green checkmark icon with a label indicating "Operational" for normal system function.
    - Alternative status icons with accompanying text messages can be used for warnings (yellow triangle) or critical errors (red exclamation mark).
- **Right Side:**
  - **Alert Settings:**
    - Slider control for adjusting the audio alert volume.
    - Mute/unmute toggle switch for audio alerts, visually representing the current state (mute symbol or speaker symbol).
  - **LiDAR Sensor Status (Optional):**
    - Green checkmark for normal operation.
    - Yellow triangle for warnings (e.g., signal strength issues).
  - **Connection Status (Optional):**
    - Green checkmark for a stable connection.
    - Red exclamation mark for connection loss.

Previously Created UI Designs:
https://www.tavatrip.com/

2. User Interaction Flow:

- **System Startup:**
  - The user turns on the system, and the GUI automatically launches.
  - The runway map on the top half remains blank initially.
- **Real-time Operation:**
  - As the FOD detection vehicle moves, the map populates dynamically. The scanned area might be highlighted with a different color or shading.
  - If the AI model detects FOD, an immediate alert is triggered:
    - The visual alert in the center section (red flashing circle) activates.
    - An audible alert might sound (depending on user settings).
    - The FOD Details section displays the type and estimated size of the detected object.
- **User Actions:**
  - The user can acknowledge the alert and focus on clearing the FOD.
  - They might access the optional settings menu to adjust preferences.
- **FOD History (Optional):**
  - The system can maintain a record of detected FOD. This could be a separate screen accessible through a menu button.
  - The FOD history would list timestamps, location data, and type of each detected debris.

3. Design Considerations:

- **Clarity and Simplicity:** Prioritize a clean, uncluttered interface for quick information processing.
- **Color Coding:** Use distinct colors to represent FOD types (e.g., red for high priority, yellow for caution).
- **Minimal Text:** Rely on icons and visual cues whenever possible for faster comprehension.
- **Customization:** Allow for some level of user customization (e.g., alert volume) to cater to individual preferences.

User Interaction:

- The user starts the system, and the real-time map populates as the vehicle scans the runway.
- FOD triggers an alert with details displayed on the screen.
- The user can review the FOD history log and adjust system settings as needed.

5. Backend Architecture

Brief:

- **Sensor Module:** Captures LiDAR data from the runway environment.
- **AI Processing Unit:** Processes LiDAR data using the trained AI model for FOD classification.
- **GUI Module:** Provides a user interface for system interaction and visualization.
- **Database:** Stores FOD history data (timestamps, location, type).

Here's a recommendation for languages based on the constraints:

- Given the requirement for offline deployment, web technologies (HTML, CSS, JavaScript) with a focus on offline packaging using Service Workers can be a viable option. It leverages familiar languages and offers flexibility, but achieving a feature-rich and visually appealing GUI might require additional effort.
- Electron is another strong contender, allowing you to build a standalone application using web technologies while providing access to native functionalities through Node.js integration. However, the application size might be larger, and packaging for different operating systems needs to be considered.

Here's a rough database schema for the FOD data using MongoDB:

```
const fodSchema = new mongoose.Schema({
  timestamp: {
    type: Date,
    default: Date.now,
    required: true,
  },
  location: {
    type: {
      type: String,
      enum: ['Point'], // GeoJSON Point type
      required: true,
    },
    coordinates: {
      type: [Number], // Array of [longitude, latitude]
      required: true,
      index: '2dsphere', // Enable geospatial indexing
    },
  },
  type: {
    type: String,
    required: true,
  },
  size: {
    type: {
      type: String, // ("5cm x 3cm") or ( 0.05)
    },
    required: true,
  },
  image: {
    type: String,
  },
  alertAcknowledged: {
    type: Boolean,
    default: false,
  },
});
```

Explanation:

- **timestamp:** This field stores the date and time when the FOD was detected (required).
- **location:** This field uses the GeoJSON Point type to represent the FOD's location on the runway. It includes:
  - **type:** Specifies the geometry type as "Point".
  - **coordinates:** An array containing the longitude and latitude coordinates (required and indexed for efficient geospatial queries).
- **type:** This field stores the identified type of FOD (e.g., "Screw", "Tool") and is

required.
- **size:** This field stores the estimated size of the FOD. It can be a string representing dimensions (e.g., "5cm x 3cm") or a number for simpler cases (e.g., 0.05 meters). This field is also required.
- **Optional fields:** These fields can be added later based on future needs:
  - **image:** Stores a path or reference to an image of the FOD captured by the system (if available).
  - **alertAcknowledged:** A boolean flag indicating whether the FOD alert has been acknowledged by the user.

GeoJSON Indexing:

The coordinates field within the location object is indexed using the 2dsphere index. This allows for efficient spatial queries, enabling functionalities like:

- Retrieving all FOD data within a specific radius of a given point on the runway.
- Visualizing historical FOD occurrences on a map based on their location data.

The backend architecture is the backbone of the system, handling data flow and communication between various components. Here's a detailed breakdown of the architecture for the AI-powered FOD detection system:

Components:

- **Sensor Module:**
  - Captures real-time LiDAR data from the runway environment.
  - Pre-processes the raw LiDAR data (e.g., filtering noise) for efficient transmission.
- **AI Processing Unit:**
  - Receives the pre-processed LiDAR data from the sensor module.
  - Runs the trained AI model on the data to identify potential FOD objects.
  - Outputs the classification results (FOD type, location) when FOD is detected.
- **GUI Module:**
  - Receives data from the AI processing unit, including real-time runway map updates and FOD alerts.
  - Processes and displays the information on the user interface.
  - Allows user interaction for acknowledging alerts and potentially accessing settings.
- **Database:**
  - Stores historical FOD data for future reference and analysis.
  - This data might include timestamps, location coordinates, and type of detected FOD.

Data Flow (utilizing Flowcharts):

Here's a step-by-step explanation with a flowchart representation:

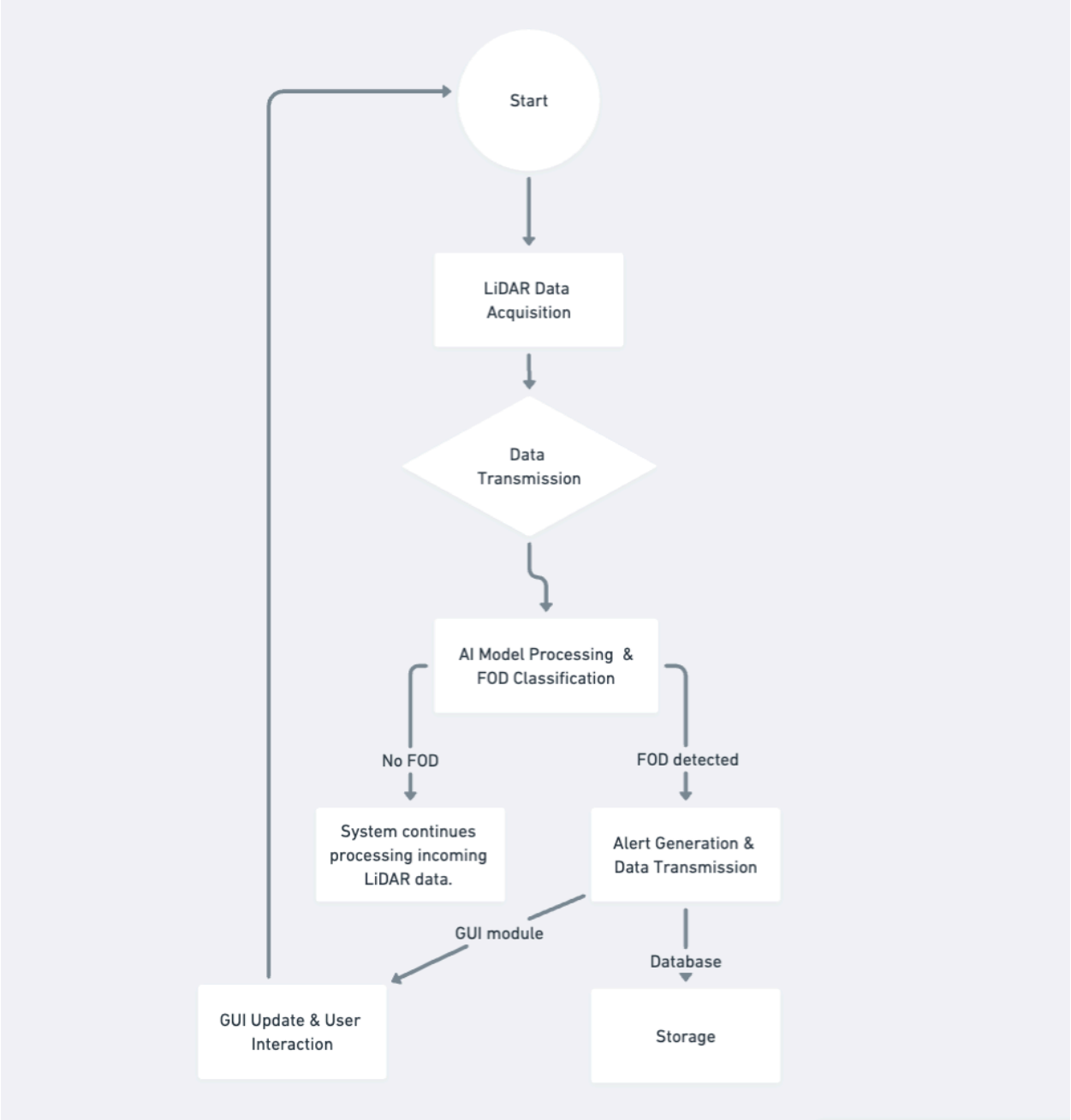
1. **Start:** The system is powered on, and all components initialize.
2. **LiDAR Data Acquisition :**
    - The sensor module continuously scans the runway environment using LiDAR.
    - The raw LiDAR data is pre-processed to remove noise or irrelevant information.
3. **Data Transmission:**
    - The pre-processed LiDAR data is transmitted from the sensor module to the AI processing unit.
    - This communication can leverage a high-bandwidth, low-latency protocol like CAN bus for real-time data exchange.
4. **AI Model Processing :**
    - The AI processing unit receives the LiDAR data stream.
    - It feeds the data into the trained AI model for real-time analysis.
5. **FOD Classification :**
    - The AI model analyzes the data to identify potential FOD objects.
    - If **FOD is detected (represented by arrow #3):**
        - The AI model extracts relevant information like FOD type and estimated location from the LiDAR data.
    - If **no FOD is detected (represented by arrow #4):**
        - No further action is taken at this stage. The system continues to process incoming LiDAR data.
6. **Alert Generation & Data Transmission:**
    - In case of FOD detection, the AI processing unit generates an alert message containing details like FOD type, location, and timestamp.
    - This alert message is transmitted to the GUI module for visual and audio notifications (represented by arrow #5).
    - Additionally, the relevant FOD data is sent to the database for storage (represented by arrow #6).
7. **GUI Update & User Interaction:**
    - The GUI module receives the alert message and FOD data.
    - It processes this information to update the real-time runway map, highlighting the FOD location.
    - The GUI triggers visual and audio alerts to notify the user.
    - The user can acknowledge the alert and interact with the system as needed (e.g., adjust settings).
8. **Database Storage:**
    - The database receives the FOD data (type, location, timestamp) and stores it for historical reference.

Communication Protocol:

- **Internal communication:** Utilize a high-bandwidth, low-latency protocol like **CAN bus (Controller Area Network)** for real-time data exchange between modules.

Data Flow (Flowchart):

1. LiDAR data is continuously acquired by the sensor module.
2. The data stream is fed into the AI processing unit.
3. The AI model analyzes the data and classifies potential FOD.
4. If FOD is detected, an alert is sent to the GUI module for visual and audio notifications.
5. Relevant FOD data (timestamp, location, type) is stored in the database.
6. The GUI displays the real-time runway map with identified FOD and allows user interaction.



6. Hardware Requirements

6.1 Processing Unit:

- **Central Processing Unit (CPU):** Select a CPU with sufficient cores and processing power to handle the following tasks in real-time:
  - LiDAR data processing (point cloud generation, filtering, and feature extraction).
  - Running the AI model for FOD classification.
  - Communication with other system components (LiDAR sensor, display unit).
- **Graphics Processing Unit (GPU):** Deep learning models leverage GPUs for faster processing. Here's what to consider:
  - **Processing power:** A powerful GPU is recommended for real-time FOD classification, especially if using complex models. Consider GPUs designed for embedded systems for efficiency and compact size.
  - **Memory (VRAM):** Adequate VRAM is crucial for storing intermediate processing results within the GPU for faster access.
- **Memory (RAM):** Choose sufficient RAM to handle LiDAR data streams, model execution, and application multitasking.
- **Storage (SSD):** A solid-state drive (SSD) is essential for fast data storage and retrieval. Consider capacity needs for historical FOD data and potentially captured FOD images.
- **Ruggedization:** The processing unit needs to withstand shocks, vibrations, and temperature extremes encountered on naval vessels. Choose a unit with a fanless design or proper cooling mechanisms for harsh environments.

6.2 LiDAR Sensor:

- **Range:** The LiDAR sensor's range should cover the entire width and desired length of the runway for complete inspection in a single pass.
- **Field of View (FOV):** A wide FOV is necessary to capture data across multiple traffic lanes on the runway simultaneously.
- **Point Density:** High point density ensures detailed 3D representations for effective FOD object detection, especially small objects like screws or bolts (meeting the minimum 3mm requirement). Consider the trade-off between point density and data acquisition speed (higher density might require slower scans).
- **Weatherproofing:** The sensor needs to function reliably in various weather conditions (rain, snow, fog). Look for sensors with IP ratings indicating their level of protection against dust and water ingress.

6.3 Display Unit:

- **Size:** Choose a display unit with a size that provides clear visibility of the GUI elements and FOD data visualization. Consider factors like space constraints within the vehicle and user ergonomics.
- **Touchscreen (Optional):** A touchscreen display allows for intuitive user interaction with the system. However, physical buttons can be an alternative depending on user preferences and operating environment considerations (e.g., potential limitations with gloves or wet hands).
- **Brightness:** The display should have high brightness for clear visibility in various lighting conditions, including direct sunlight. Ruggedized displays often prioritize durability over brightness; consider using an anti-glare screen protector to improve usability in bright environments.
- **Ruggedization:** The display unit should be resistant to shocks, vibrations, and temperature extremes to ensure reliable operation on naval vessels.

Additional Considerations:

- **Power Supply:** Select a reliable power supply solution that can provide continuous operation for the system. Options include:

  Onboard battery system: Choose a high-capacity battery system that can last for extended deployments between charges. Consider factors like battery weight and charging time.
  Connection to vehicle's electrical system: If possible, connect the system to the vehicle's electrical system for continuous power during operation. Ensure proper voltage regulation to avoid damaging system components.
- **Communication Interface:** Implement a high-bandwidth, low-latency communication protocol like CAN bus to ensure real-time data exchange between the LiDAR sensor, processing unit, and display unit.

7. Additional Considerations:

- **Security:** Implement appropriate security measures to protect the system from unauthorized access or data breaches.
- **Redundancy:** Consider incorporating redundancy in critical components (e.g., dual LiDAR sensors) to enhance system reliability.
- **Data Management:** Establish a strategy for data management, including data storage capacity, backup procedures, and data anonymization (if applicable).

8. Conclusion :

Summary of LLD for AI-powered FOD Detection System
This system utilizes a vehicle-mounted platform with sensors and AI to detect Foreign Object Debris (FOD) on runways at Naval Air Stations and Aircraft Carriers.

1. FOD Detection with Sensors:

- **LiDAR Sensor:** Chosen for its high-resolution 3D mapping capabilities, small object detection (meeting the 3mm minimum requirement), and all-weather functionality.
- **Discarded Options:**
  - Electro-Optical Cameras (EO Cameras): Limited effectiveness in low-light conditions.
  - Infrared Cameras (IR Cameras): Not ideal for all FOD types and susceptible to false positives.

2. AI Model for FOD Classification:

- Processes LiDAR data in real-time for immediate FOD alerts.
- Reduces human error compared to manual inspections.
- Learns and adapts to identify various FOD types over time based on a vast training dataset.

3. Graphical User Interface (GUI):

- **Features:**
  - Real-time runway map with FOD highlighted.
  - FOD classification (type and size).
  - Alert system (visual and audible) for FOD presence with location data.
  - FOD history log for reference.
  - System settings for configuration.
- **Wireframe Design:**
  - Top half: Runway map with color-coded FOD location and type.
  - Bottom half: Information panel with sections for FOD details, alerts, and optional system settings.
- **User Interaction:**
  - User starts the system, and the map populates as the vehicle scans.
  - FOD triggers alerts with details displayed on the screen.
  - User can review FOD history and adjust system settings.

4. Backend Architecture:

- **Components:**
  - Sensor Module: Captures LiDAR data from the runway environment.
  - AI Processing Unit: Processes LiDAR data for FOD classification using the trained AI model.
  - GUI Module: Provides the user interface for system interaction and visualization.
- Database: Stores historical FOD data (timestamps, location, type).
- **Data Flow:** LiDAR data flows from the sensor module to the AI unit for processing. Classified FOD data and alerts are sent to the GUI module for display. Relevant FOD data is stored in the database.
- **Communication Protocol:** Utilizes a high-bandwidth, low-latency protocol like CAN bus for real-time data exchange between modules.
- **Hardware Requirements:**
  - Powerful embedded computer with a dedicated GPU for AI processing (ruggedized for harsh environments).
  - High-resolution LiDAR sensor suitable for outdoor operation and long-range scanning.
  - Rugged display unit for the GUI.

5. Backend Implementation :

- Node.js and Mongoose (or similar database library) can be used for the backend server.
- Mongoose schema defines the structure of FOD data in the database.
- API endpoints are created for receiving FOD data from the sensor module and storing it, as well as retrieving historical FOD data for the GUI.

  Reasoning :

  - **Event-driven, non-blocking I/O model:** Well-suited for real-time applications like this system, where it needs to handle continuous data streams from the LiDAR sensor and respond promptly.
  - **Rich ecosystem of libraries and frameworks:** Node.js boasts a vast library ecosystem, including options for:
    - Interfacing with hardware devices like LiDAR sensors (e.g., serialport library).
    - Building real-time applications (e.g., socket.io).
    - Data processing and manipulation (e.g., lodash).
    - Interacting with MongoDB (e.g., Mongoose ODM).
  - **Large and active community:** Extensive online resources, tutorials, and forums can aid in development and troubleshooting.

6. Additional Considerations:

- Security measures to protect the system from unauthorized access.
- Redundancy in critical components for enhanced reliability.
- Data management strategy for storage, backup, and potential anonymization.

This LLD design outlines a comprehensive AI-powered FOD detection system with a user-friendly interface, robust backend architecture, and the potential for Node.js implementation.