FLOW AND ARCHITECTURE:

1. Problem Statement:

- **Goal:** Develop a solution to monitor afforestation efforts using drone imagery, identifying casualties (non-surviving saplings) and analysing growth.
- **Input:** Raw drone imagery of afforestation patches.
- **Output:** Reports/visuals highlighting sapling survival and growth, along with casualties

2. Tools and Technologies:

- **Programming and Automation libraries:**
  - Using in **Python** and its libraries like:
    - **OpenCV**: For image processing.
    - **NumPy** and **Pandas**: For handling numerical and tabular data.
    - **GeoPandas**: For geospatial data manipulation.
    - **Matplotlib** : For visualization of data ,making histogram ,graph , table for the better understanding of user .
- **Frameworks:**
  - **Django**(Using **HTML**, **CSS**, **Python**, **JavaScript**, and **3.js)** : to create interactive 3D websites for data visualization and analysis.
- **Deployment Architecture:**
  - **Cloud services (GCP):** (Using vertex AI , Cloud Vision API, Compute Engine, Cloud AI APIs, Looker Studio, Cloud Storage).
  - **Docker:** CI/CD pipelines, containerization Docker
- **Drone Imagery Tools:** Pix4D for pre-processing of drone image.

3 Understanding the Operations:

1. **OP1 (March-May, Year 1):**
- Task: Pits of 45x45x45 cm are dug and left to the sun to eliminate microbes.
- Method: Manual labor or JCBs (digging machines).
- Visibility: Pits are visible from the sky.
2. **OP2 (June 15-July 30, Year 1):**
- Task: Planting 18-month-old saplings (4-6 ft tall with minimal foliage).
- Timing: During the first monsoon rains and the following month.
3. **OP3 (Oct-Nov, Year 1, Year 2, Year 3):**
- Task: Weeding by clearing a 1m diameter of soil around saplings.
- Visibility: Cleared area is visible from the sky.

3. Solution Approach:

A. Understanding the Data

1. **Drone Elevation**: 70–80 m for consistent scale.
2. **Pixel Resolution**: 2.46–2.81 cm per pixel, ideal for sapling detection.
3. **Overlap**: 65% sidelap & 75% endlap for complete coverage.

B. DATA PREPROCESSING:

1. Organize The Raw Data:

- Metadata: Every image contains extra information (metadata) like:
  - The time the image was captured.
  - The location (latitude, longitude) for every pixel.

- Camera settings like resolution and angle.

- Steps to Organize:
  1. Look at the timestamps in the metadata to figure out when each image was taken (e.g., before OP1, after OP1, OP2, OP3).
  2. Sort the images into separate folders based on their time:
     - Folder 1: Before_OP1
     - Folder 2: After_OP1
     - Folder 3: After_OP2
     - Folder 4: After_OP3
  3. Make sure there are no duplicates or corrupted images. Rename the files for easy identification (e.g., OP1_IMG_001.jpg).

2. Generate Orthomosaics:

- Orthomosaic image : An orthomosaic is a big, stitched-together map made by combining smaller drone images. It gives a clear, bird's-eye view of the entire area.

- Advantages of Orthomosaics Image :
  - It's easier to analyze one big map instead of many small images.
  - The map uses the image's location data (geo-coordinates) to make it accurate.

- How to Create Orthomosaics:
  1. Use tools like **Pix4D**
  2. Upload all your images into the software.
  3. The software will automatically align and stitch the images using their overlap (65% sidelap and 75% endlap).

Make sure the orthomosaic is georeferenced, meaning it keeps the real-world locations of the trees.

C. IMAGE PREPROCESSING:

1. **IMAGE RESIZING:-**
   i. RESIZING: Adjust the image dimensions using cv2.resize().
   ii. IMAGE PYRAMID: Modify image resolution (scale up or down) using cv2.pyrDown() and cv2.pyrUp().
   iii. PERSPECTIVE RESIZING: Correct or adjust image perspective using OpenCV and NumPy functions:
   - cv2.getPerspectiveTransform()
   - cv2.warpPerspective()

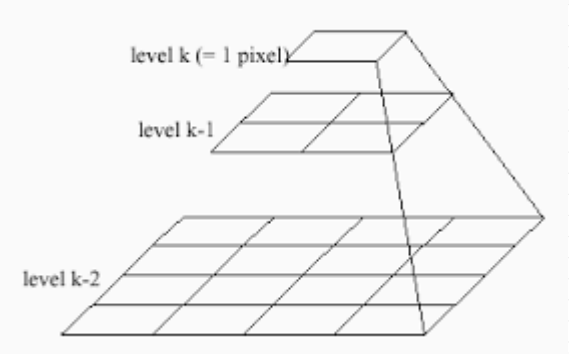
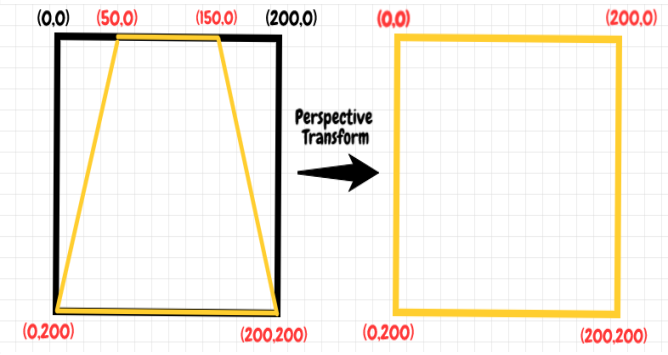


IMAGE PYRAMID



PERSPECTIVE RESIZING

1. **CONVERTING COLOR IMAGE TO GRAYSCALE :-**

Converting images to grayscale reduces complexity by using only one channel instead of three, making processing faster and easier. It focuses on important details like edges and patterns while removing unnecessary color information. Grayscale also saves storage space and works better for many algorithms. Converting images to grayscale is important because:

Less Data to Process: Grayscale images are smaller and simpler since they have one color channel instead of three (RGB), making them faster to work with.
Focus on Details: It removes color and focuses on brightness, which is useful for finding edges, shapes, or patterns.
Better for Some Tasks: Many tools and algorithms work better with grayscale because color isn't always important for tasks like face detection or text recognition.
Saves Space: Grayscale images take up less storage space than color images.

In short, grayscale makes images easier to handle while keeping the important details.

Its is done by using OpenCV:

- **cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)**
- **cv2.cvtColor(image, cv2.COLOR_HSV2GRAY)**
- **cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)**

2. IMAGE THRESHOLDING :-

1**. MULTI-OTSU THRESHOLDING**

- **Description:**
  - An extension of Otsu's method for multi-level thresholding.
  - Instead of separating the image into two classes (foreground and background), it divides the image into multiple classes based on pixel intensity.
  - The algorithm minimizes intra-class variance for all classes.

- **Steps:**
  i. Compute Histogram: Get pixel intensity distribution.
  ii. Divide Image: Split image using candidate thresholds.
  iii. Calculate Entropy: Measure disorder in each region.
  iv. Maximize Entropy: Choose threshold with highest combined entropy.

- **Use Case:** Ideal for images with more than two distinct intensity distributions, such as separating soil, saplings, and dead patches.

2. KAPUR'S METHOD THRESHOLDING

- Description:
  - Multi-level thresholding method based on maximizing the sum of entropy of classes.
  - Entropy measures the randomness or complexity within each region, and the method finds thresholds that maximize the overall entropy of segmented regions.
- Steps:
  i. **Compute Histogram**: Calculate pixel intensity distribution and probabilities.
  ii. **Divide Image**: Split image using candidate thresholds.
  iii. **Calculate Entropy**: Measure entropy for each region.
  iv. **Maximize Entropy**: Select threshold that maximizes the sum of entropies.

- Use Case: Effective when the image contains multiple objects with varying brightness levels, ensuring better segmentation of regions like healthy saplings, dry areas, and dead saplings.

3. IMAGE FILTERING :-

Remove irrelevant data such as shadows, water patches, or other artifacts using filtering techniques. OpenCV provides a wide range of functions for **filtering** and **enhancing image quality**, covering tasks like smoothing, sharpening, noise reduction, and contrast adjustment. Below is a categorized list of commonly used OpenCV functions for these purposes:

Category	Function	Use
Smoothing	cv2.blur, cv2.GaussianBlur	Noise reduction
Sharpening	cv2.filter2D, cv2.addWeighted	Enhancing details
Contrast Adjustment	cv2.equalizeHist, cv2.convertScaleAbs	Enhancing contrast
Edge Detection	cv2.Canny, cv2.Sobel, cv2.Laplacian	Highlighting edges
Noise Reduction	cv2.fastNlMeansDenoising	Removing image noise
Morphological Ops	cv2.morphologyEx, cv2.erode	Structural enhancement

D. IMAGE SEGMENTATION

Goal

- Isolate saplings from the background for individual analysis.
- Detect boundaries and segment each sapling.

Techniques

1. Thresholding: Use methods like Otsu's or adaptive thresholding for basic segmentation.
2. Deep Learning: Apply models like U-Net or Mask R-CNN for pixel-wise segmentation of saplings.

Steps

1. Contour Detection:
    - Use OpenCV's findContours to detect sapling regions.
    - [contours, _ = cv2.findContours(binary_thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)]
2. Segmentation Mask:
    - Use deep learning for precise segmentation.
    - from keras.models import load_model
      segmented_output = model.predict(preprocessed_image)
3. Bounding Boxes:
    - Draw bounding boxes around saplings for further measurements or analysis.
    - for contour in contours:
      x, y, w, h = cv2.boundingRect(contour)
      cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

- **Georeferencing:** Align images from different drone flights to ensure accurate comparisons over time.

E. FEATURE EXTRACTION:

1. SURVIVAL DETECTION: Train an ML model to classify saplings as "alive" or "casualty" based on features like color, texture, and size.

Objective: Classify saplings as "alive" or "casualty" using features like color, texture, and size.

Convolutional Neural Networks (CNNs):

- Extract spatial features (color, texture, patterns) effectively.
- Use binary classification (alive vs. casualty).
- Pretrained models (ResNet, EfficientNet, MobileNet) can be fine-tuned for this task.

2. GROWTH MEASUREMENT: Use deep learning-based regression models to estimate sapling height, canopy size, or biomass growth.

c. Machine Learning Model:

- Use a combination of:
  - **Convolutional Neural Networks (CNNs):** For visual feature extraction.
  - **Object Detection Models:** To count and locate saplings (e.g., YOLO, Faster R-CNN).
  - **Change Detection Algorithms:** To compare images over time and identify changes.

d. Output Visualization:

- Generate heatmaps or overlays on drone images to highlight casualties and healthy saplings.
- Create a dashboard to track the health and growth of patches over time.

e. Validation:

- Validate model outputs with ground truth data provided by the forest department.
- Use metrics such as accuracy, precision, recall, and F1-score for evaluation.

4 . Expected Outcomes:

- **Casualty Analysis:** Identify and quantify non-surviving saplings.
- **Growth Tracking:** Visualize and quantify growth patterns of saplings.
- **Actionable Insights:** Provide reports to assist the forest department in decision-making.

This solution will help ensure the success of afforestation programs by enabling timely interventions and improving monitoring efficiency.

1. Understanding the Data

- **Drone Elevation**:
  - Images captured at heights of 70–80 meters.
  - Ensures consistent scale and resolution for comparative analysis.
- **Pixel Resolution**:
  - Ranges from **2.46 cm to 2.81 cm** per pixel.
  - High-resolution data suitable for detecting saplings and measuring growth.
- **Overlap (Sidelap and Endlap)**:
  - 65% sidelap and 75% endlap ensure robust coverage and avoid gaps.
- **Orthomosaic**:
  - Medium-resolution stitched imagery generated by **Pix4D** software.
  - Participants can regenerate orthomosaics for better control or higher resolution using their preferred tools.
- **Metadata**:
  - Contains geo-coordinates for each pixel, enabling spatial analysis and georeferencing.

---

2. Workflow for Analysis

Step 1: Data Preprocessing

1. **Organize the Raw Data**:
   - Use metadata to filter and sort images by time (before OP1, after OP1, OP2, OP3).

2. **Generate Orthomosaics (Optional)**:
    - Use tools like Pix4D, Agisoft Metashape, or OpenDroneMap to generate custom orthomosaics.
    - Ensure georeferencing to maintain spatial accuracy.

Step 2: Image Preprocessing

- **Resizing**: Standardize image dimensions for uniform input to models.
- **Noise Filtering**: Use Gaussian blur or median filtering to reduce noise.
- **Color Adjustment**: Convert to grayscale, HSV, or other formats as needed.
- **Image Enhancement**: Use histogram equalization to improve contrast.

Step 3: Image Analysis

1. **Thresholding and Segmentation**:
    - Apply **Multi-Otsu** or **Kapur's Method** for thresholding to segment regions (saplings, soil, dead areas).
    - Extract features like color, texture, and size.
2. **Classification**:
    - Train ML models for survival detection (alive vs casualty).
    - Train regression models for growth metrics (height, canopy, etc.).

Step 4: Geo-Spatial Analysis

- Leverage metadata for spatial insights:
    - Map sapling locations using pixel geo-coordinates.
    - Measure area and density changes over time.
- Use GIS tools like **QGIS** or **ArcGIS** for visualization and analysis.

Step 5: Visualization and Reporting

- Generate comparative orthomosaic maps for all four surveys.
- Create survival and growth metrics dashboards using **Looker Studio** or **Power BI**.
- Use **Google Maps API** for interactive geospatial visualization.

---

3. Tools and Software

- **Image Processing**: OpenCV, Pillow, or Scikit-Image.
- **GIS Analysis**: QGIS, ArcGIS, or Google Earth Engine.
- **Orthomosaic Generation**: Pix4D, Agisoft Metashape, OpenDroneMap.
- **ML Modeling**: TensorFlow, PyTorch, or Scikit-Learn.
- **Visualization**: Looker Studio, Power BI, or custom dashboards with Plotly.

---

4. Final Deliverables

1. Annotated orthomosaic maps with detected saplings and casualty areas.
2. Growth metrics report (height, canopy, biomass).
3. Interactive dashboard for visualizing survival rates and spatial data.
4. Comprehensive project documentation with methodology and findings.

Simplified Workflow for Drone-Based Afforestation Monitoring

---

1. Data Collection

- **Input:** High-resolution drone images of the afforestation patch.
- Ensure consistent conditions (lighting, angle, season) for accurate analysis.

---

2. Image Preprocessing

Prepare images for analysis by cleaning and standardizing the data.

1. **Resizing:** Standardize image dimensions to make processing faster.
2. **Color Conversion:** Convert images to grayscale or HSV to highlight sapling features.
3. **Filtering:** Remove noise using techniques like blurring or sharpening.
4. **Enhancement:** Adjust contrast or brightness for better clarity.

---

3. Image Segmentation

- Separate saplings from the background.
- Use techniques like thresholding (e.g., Otsu's method) or deep learning models (e.g., U-Net or Mask R-CNN).
- Detect sapling boundaries and isolate each sapling for individual analysis.

---

4. Feature Extraction

- **Identify key characteristics of each sapling:**
    - **Color:** Average green or brown levels to assess health.
    - **Texture:** Patterns on the sapling for damage detection.
    - **Size:** Height, canopy spread, and area covered.
    - **Shape:** Irregularities in structure or growth patterns.

---

5. Survival Detection (ML Model 1)

- Train a machine learning model to classify saplings as either **"Alive"** or **"Casualty."**
- Use features like color, texture, and size for the classification.

---

6. Growth Measurement (ML Model 2)

- Use a regression model to estimate growth parameters:
    - **Height:** Estimate sapling height from image features.
    - **Canopy Size:** Measure the spread of leaves or branches.
    - **Biomass:** Predict weight based on area covered and density.

---

7. Post-Processing

- Overlay results (alive/casualty, height, canopy size) onto the original image.

- Create visual reports showing the health and growth distribution across the patch.

---

8. Deployment

- Integrate the workflow into a user-friendly application for forest officers.
- Use cloud services for storing and processing large datasets.

Outputs

1. **Survival Report:** Percentage of saplings alive vs. casualties.
2. **Growth Metrics:** Average height, canopy size, and overall growth.
3. **Visual Map:** Annotated image showing sapling health and growth.

This approach ensures an efficient and scalable solution for monitoring afforestation efforts!