Floodbusters

Project HydroCams

The Team

Jennie Butch

Team Lead

Nathan Hill

Architect

Noah Gooby

Client Communications

Jade Meskill

Archivist

Dylan Anderson

Release Manager

Capstone Instructor: Isaac Newton Shaffer

Capstone Mentor: Vahid Nikoonejad Fard

Our Client

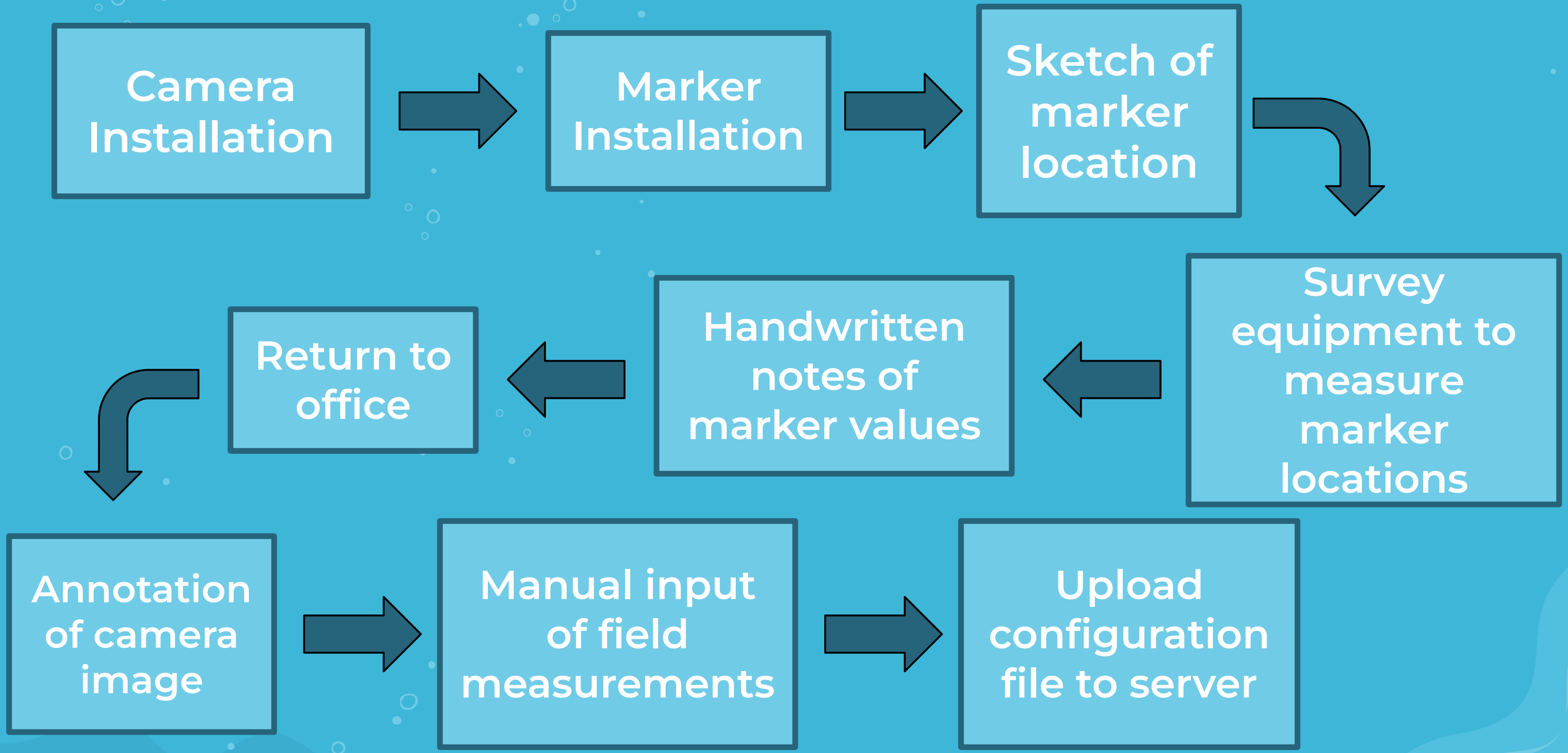
- Professor of Computer Science and researcher - NAU SICCS

- Cofounder of the FloodAware Project, overseeing the development of HydroCams

- Dr. Doerry's Goals for HydroCams:
    - Easy to Install
    - Affordable
    - Solar Powered
    - Cell-Connected
    - Automatically Calibrated

Dr. Eck Doerry

Current Process



Camera Installation

Marker Installation

Sketch of marker location

Survey equipment to measure marker locations

Handwritten notes of marker values

Return to office

Annotation of camera image

Manual input of field measurements

Upload configuration file to server

Problem Statement

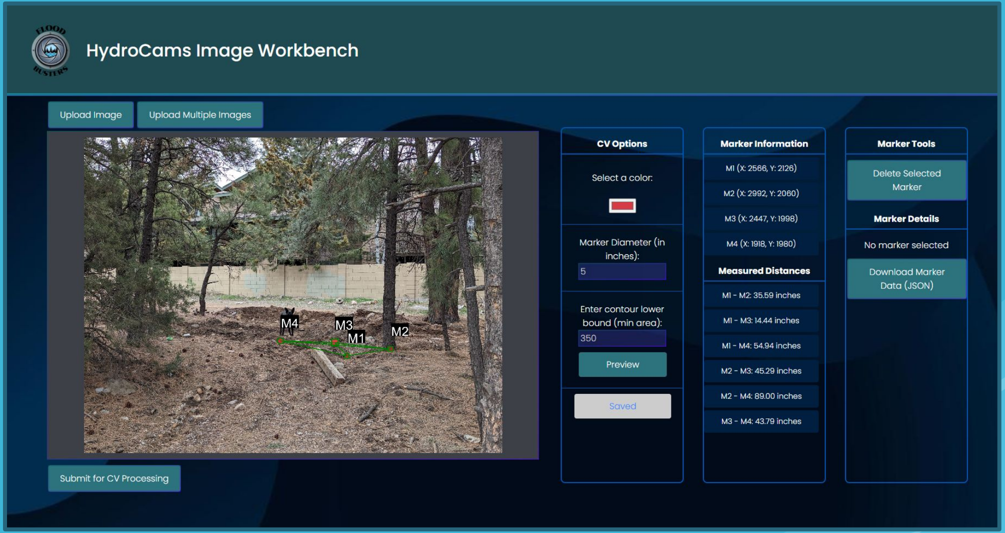
- Current flood monitoring systems require expensive and labor-intensive processes to generate calibration data
    - Specialized, expensive surveying equipment
    - Highly trained installation technicians
    - Often rely on hand-drawn images and notes
    - Prone to user error, potentially requiring multiple trips to the camera installation

- These factors culminate in an impractical and inefficient system, ripe for improvement

Solution Overview

Online Image Workbench for Calibration File Generation

- Configurable CV parameters
- Automated marker / zero point detection
- Detailed information about markers and relationships
- Downloadable calibration file
- Easily iterable workflow, with large time and resource improvements



HydroCams Image Workbench

CV Options

Select a color: 🟥

Marker Diameter (in inches): 5

Enter contour lower bound (min area): 350

Marker Information

M1 (X: 2566, Y: 2126)

M2 (X: 2992, Y: 2060)

M3 (X: 2447, Y: 1998)

M4 (X: 1918, Y: 1980)

Measured Distances

M1 - M2: 35.59 inches

M1 - M3: 14.44 inches

M1 - M4: 54.94 inches

M2 - M3: 45.29 inches

M2 - M4: 89.00 inches

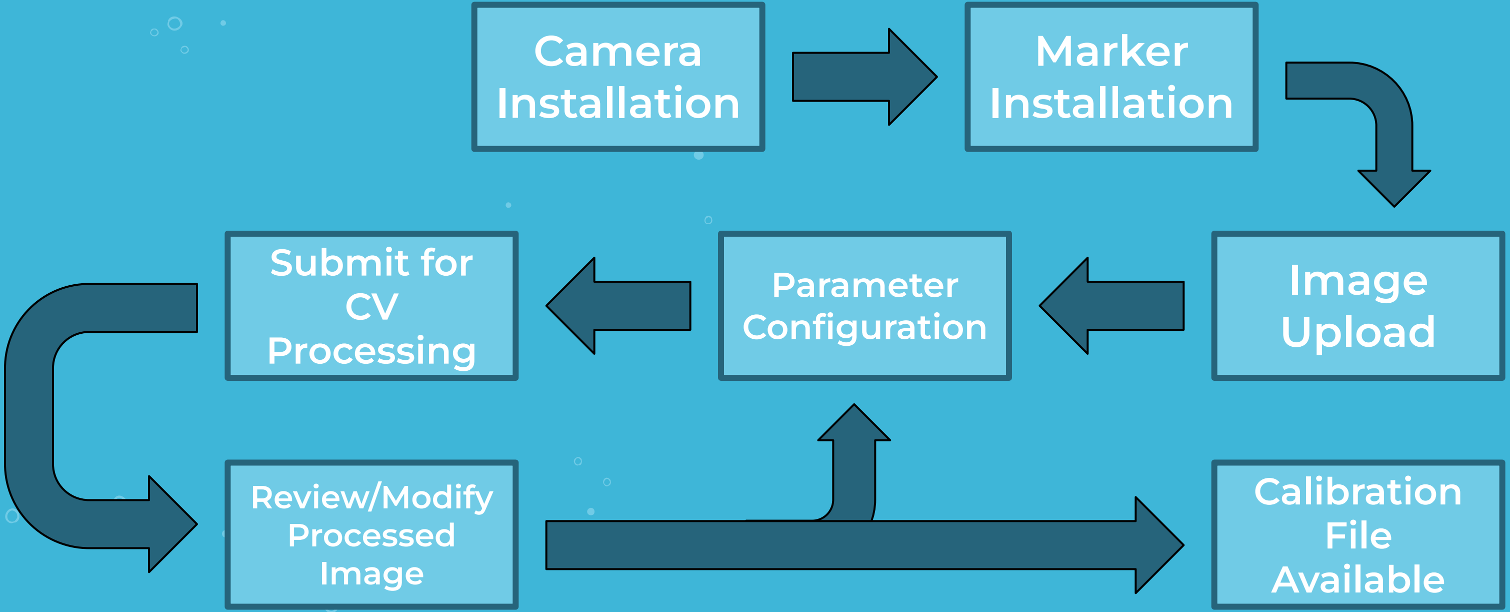
M3 - M4: 43.79 inches

Marker Tools

Marker Details

No marker selected

New, Improved Process



Camera Installation

Marker Installation

Image Upload

Parameter Configuration

Submit for CV Processing

Review/Modify Processed Image

Calibration File Available

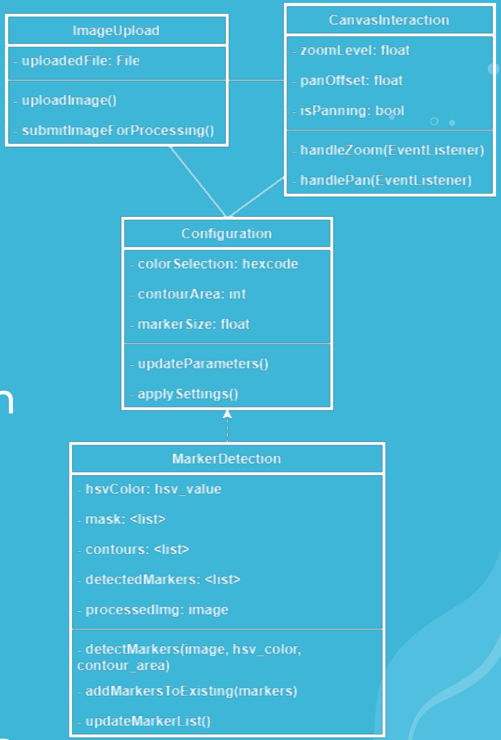
Implementation Overview

- Requirements Overview:
  - Ability to detect markers of varying sizes, shapes, and colors
  - Depth/Size calibration using known-size markers (zero-points)
  - User-friendly UI for uploading, adjusting, and running the CV

- Architectural Overview:
  - Front End: Built using HTML/CSS/JavaScript
  - Back End: Python Flask server for image upload and processing
  - Storage: Temporary file system image storage for processing; no long-term storage needed

Implementation Details

- Image Upload and Canvas Interaction
  - Allows user to upload images and dynamically interact with them using zoom/pan navigation
- Configuration
  - Provides options for users to configure CV parameters, including color selection and minimum contour area
- Marker Detection and Adjustment
  - Automatically detects markers based on color and size parameters
  - Can be easily re-run to identify any missing markers, or markers with entirely different properties



ImageUpload
- uploadedFile: File
- uploadImage()
- submitImageForProcessing()

CanvasInteraction
- zoomLevel: float
- panOffset: float
- isPanning: bool
- handleZoom(EventListener)
- handlePan(EventListener)

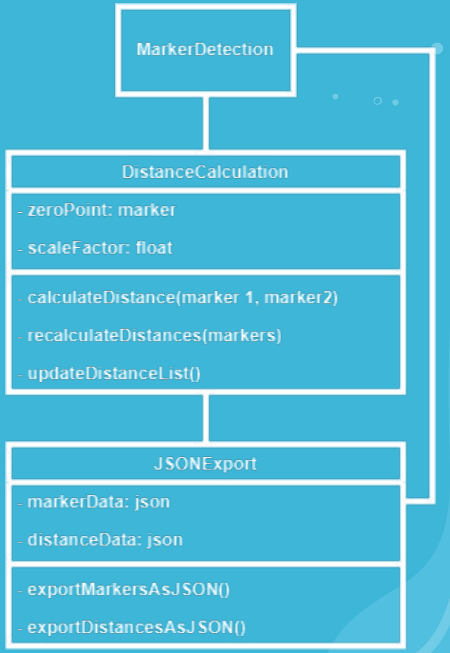
Configuration
- colorSelection: hexcode
- contourArea: int
- markerSize: float
- updateParameters()
- applySettings()

MarkerDetection
- hsvColor: hsv_value
- mask: <list>
- contours: <list>
- detectedMarkers: <list>
- processedImg: image
- detectMarkers(image, hsv_color, contour_area)
- addMarkersToExisting(markers)
- updateMarkerList()

Implementation Details, cont.

- Distance Calculation
  - Calculates distances between markers and a zero-point using pixels, converted to real world units

- JSON Export
  - Allows users to export marker data and measured distances in JSON format
  - Facilitates integration with other systems (e.g., HydroCams) for further analysis



MarkerDetection

DistanceCalculation

- zeroPoint: marker
- scaleFactor: float

- calculateDistance(marker 1, marker2)
- recalculateDistances(markers)
- updateDistanceList()

JSONExport

- markerData: json
- distanceData: json

- exportMarkersAsJSON()
- exportDistancesAsJSON()

Prototype Review

Challenges and Resolutions

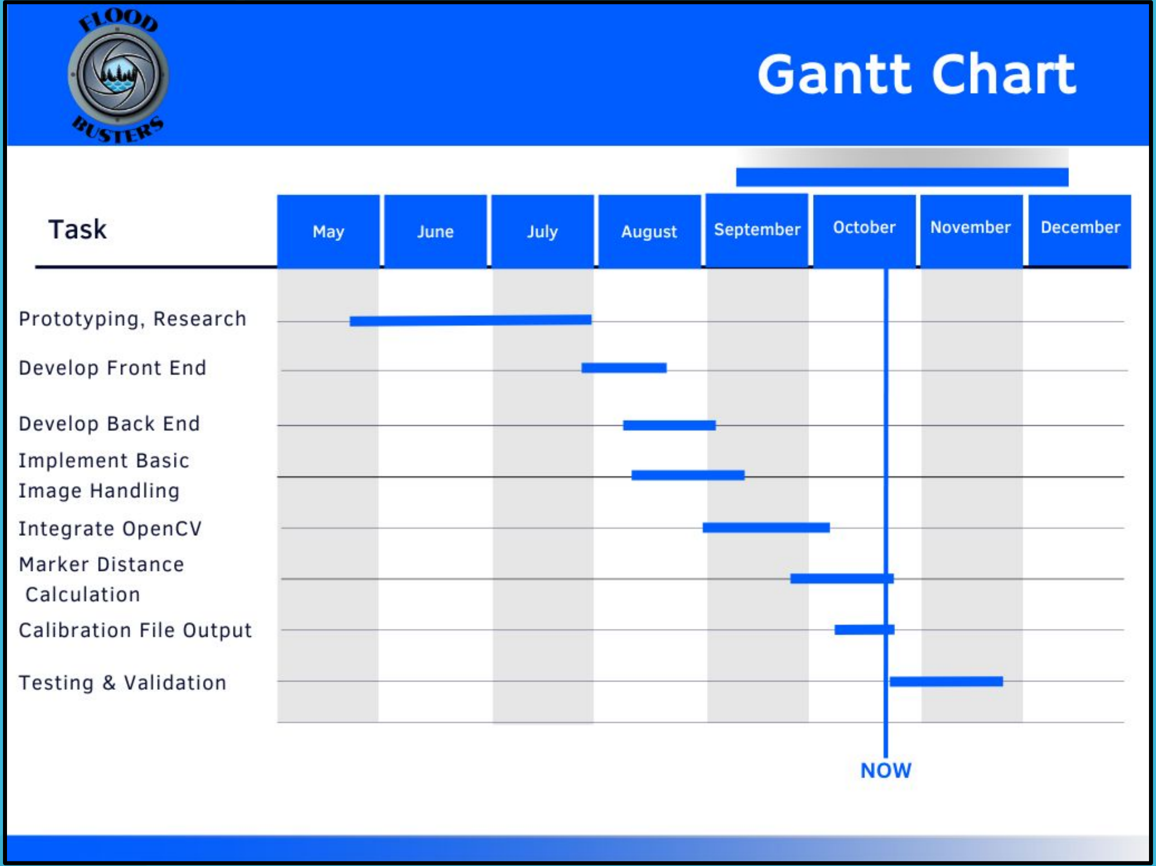
- C1: Structure-from-Motion (SfM) was too resource-intensive
  - R1: Switched to pixel-based distance calculations for simpler, efficient processing
- C2: Inconsistent marker detection under varying lighting
  - R2: Added configurable color selection and re-run functionality to improve detection
- C3: Depth causes skewed distance calculations
  - R3: Developing zero-point marker calibration for more accurate scaling
- C4: Ensuring accuracy across various camera angles
  - R4: Enhancing algorithm to adapt to marker size and pixel scaling factor

Testing Plan

-

pytest

Schedule


FLOOD BUSTERS
Gantt Chart
Task
May
June
July
August
September
October
November
December
Prototyping, Research
Develop Front End
Develop Back End
Implement Basic Image Handling
Integrate OpenCV
Marker Distance Calculation
Calibration File Output
Testing & Validation
NOW

Conclusion

- Flooding regularly wreaks havoc on lives and property, and is only expected to worsen
- Current flood monitoring systems are too cumbersome and expensive to be practical
- Our solution involves an online image workbench that utilizes computer vision to provide streamlined calibration data to help automate flood detection
- Our next steps are: To improve current distance calculation methods and then perform thorough testing and validation
- We are confident that our efforts will revolutionize the realm of flood detection, saving lives and millions of dollars in the process

Thank you!