EP-DT Summer Student Presentations 2024
20.08.2024

ADVANCEMENTS ON AERIAL ROBOTIC SYSTEMS AND SENSOR TECHNOLOGIES FOR FUTURE HEP DETECTORS

Presenter:

Kelvin Chan

Supervisors:

Francesco Mazzei,
Paolo Francesco Scaramuzzino,
Luca Bernardi,
Corrado Gargiulo

On behalf of EP-R&D WP4.2 Robotics for Detectors

About Me:

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Education: Hong Kong University of Science and Technology
Major: Physics

Other Experience:

- HKUST Robotics team leader
- One year internship at Hong Kong Observatory

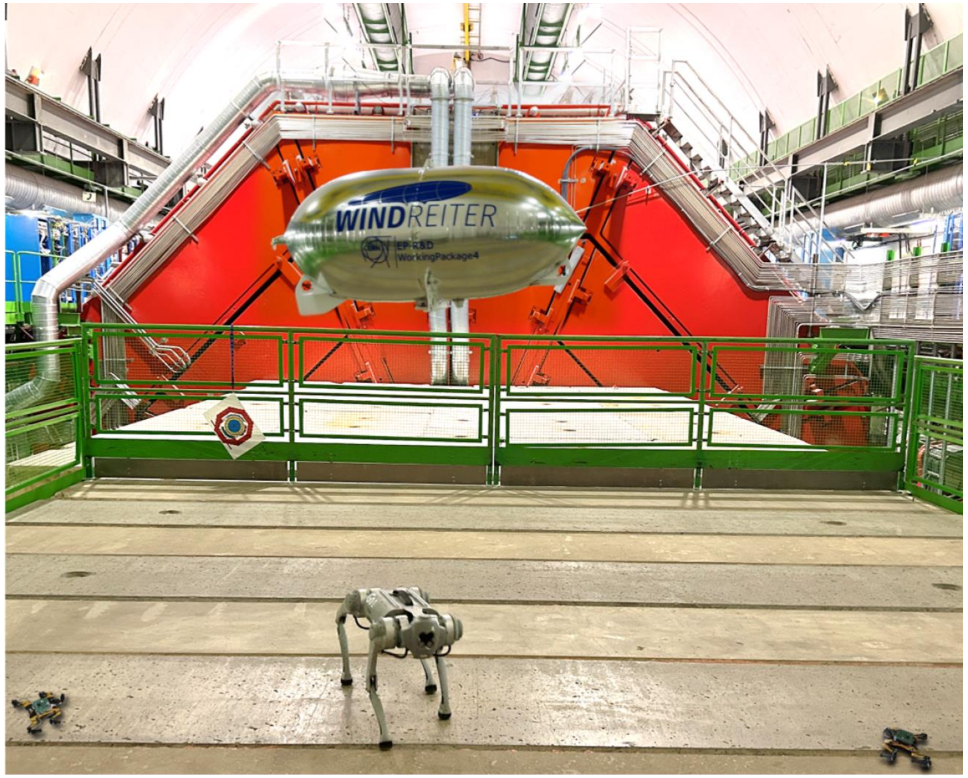
About My Group:

Project: EP-R&D WP4.2 Robotics for Detectors

Main Goal:
Automate tasks like **installation, inspection, disconnection and maintenance** of detectors using robots

Challenges:
High **radiation** and **magnetic** field environments

Solutions:
Aerial Robots Blimps (my project)
Legged Robots



About My Project:

Project: Aerial Robot (Blimp) development

What the group is working on:

1. Virtual environment for visualization of the Blimp dynamics and control (Francesco)
2. Camera placement optimization inside the cavern (Paolo)
3. Payload integration within the Blimp sensor bay

What I did during the summer project:

1. Development of a camera-based detection and localization system of the Blimp
2. Onboard integration of a radiation detector

Rundown:

Blimp localization system:

- Background Introduction
- Object detection and YOLO
- Working flow
- Results and discussion
- Localization method
- Future developments

Onboard radiation monitoring with MiniPIX TPX3 detector

- Background Introduction
- Onboard communication
- Radiation dose
- Future developments

BLIMP LOCALIZATION SYSTEM

BACKGROUND INTRODUCTION

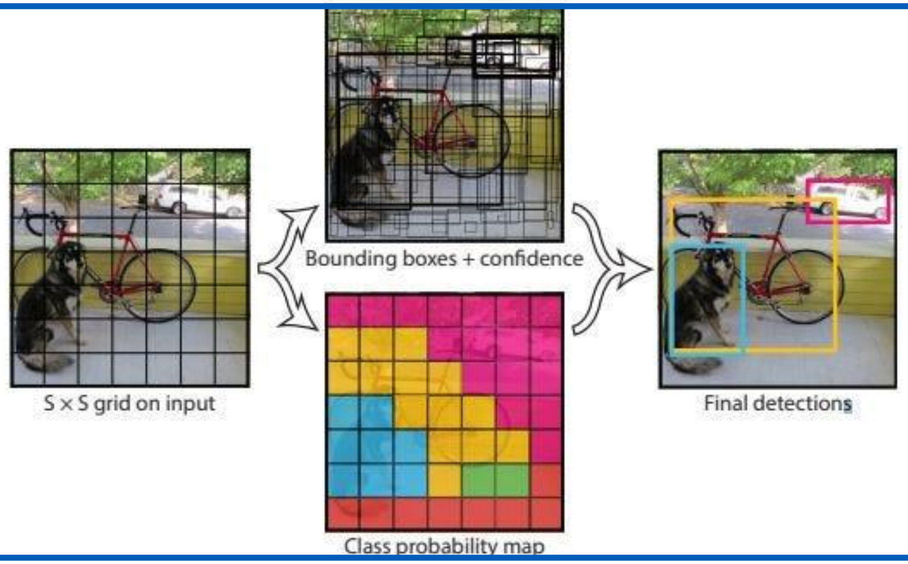
Main Goal:
- Locate the **real-time position** of the Blimp in motion

Method:
- Install **webcams** inside the **detector cavern**
- Use a camera-based detection and localization system to:
  - **Minimize** radiation & magnetic **interferences**
  - **Minimize** Blimp **weight** (**No additional hardware** is required on the Blimp)

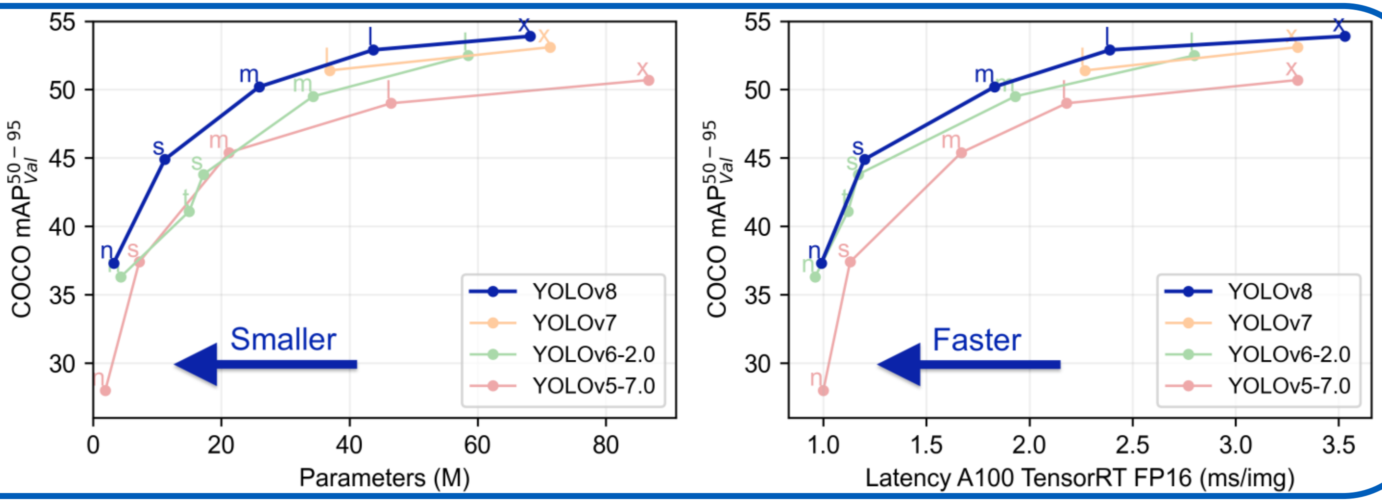
Working Principle of a camera-based Localization System:
In each video frame captured by the webcams, a **Machine Learning Model**:
1. identifies in real time the Blimp
2. creates bounding boxes around each detected Blimp
3. computes the x-y position of the center of the bounding box
4. calculates the 3D position using information coming from multiple views

You Only Look Once (YOLO):

- Deep learning algorithm for object detection
- Regression problem [1]:
  - ➔ bounding boxes and probabilities from a single image
- Exceptional speed and accuracy balance:
  - ➔ Very useful for real-time detections

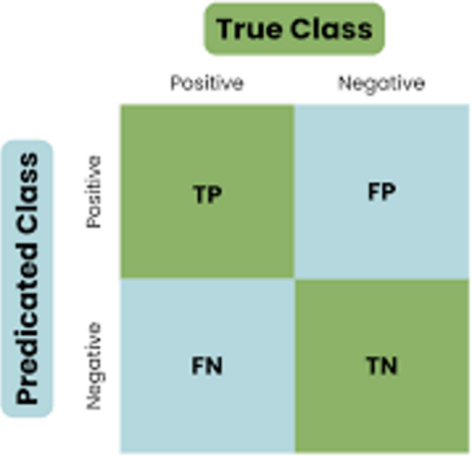


- YOLO v8 with respect to previous versions [2]:
  - ➔ Reduced latency
  - ➔ Same accuracy with less parameters



[1] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.(2016). You Only Look Once: Unified, Real-TimeObject Detection.
[2] Terven, J., Diana-Margarita Cordova-Esparza, Julio-Alejandro Romero-Gonzalez. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 toYOLOv8 and YOLO-NAS. Machine Learning and Knowledge Extraction

Object Detection Metrics:

Intersection over Union	Confusion Matrix	Precision	Recall	F1 score
$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$ 		$Precision = \frac{TP}{TP + FP}$	$Recall = \frac{TP}{TP + FN}$	$F1 = \frac{precison \ * recall}{precision + recall}$
Mismatch between the bounding box labelled by the developer and the bounding box identified by the detection model	Represents the results of model predictions	The percentage of correct predictions (how accurate the model is)	How well the model finds all positive (objects)	A measure that takes both precision and recall in account
TP: True Positive, FP: False Positive, TN: True Negative, FN: False Negative				

WORKING FLOW

Working flow:

- **Dataset Preparation**: Label Studio (Best) / Roboflow / LabelImg
- **Model Training**: YOLOv8 & Pycharm (Python Script)
- **Model deploy**: OpenCV & Pycharm (Python Script)
- **Blimp localization system**-> evaluate the performance of the detection model

GitHub

https://github.com/Kelvinchan324/CERN_blimp_dectection

Refer to GitHub page for further details and code

DATASET PREPARATION

DATASETS

Label & Source

roboflow

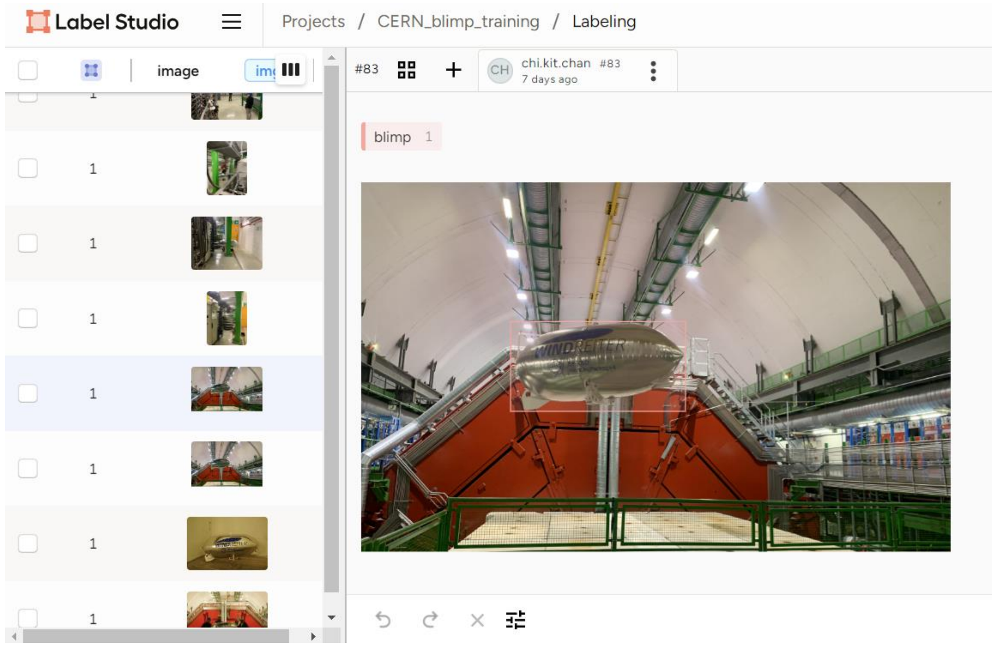
Label Studio

LabelImg

Dataset Preparation:

1. Separate dataset of images into three groups
   ➔ Training, validation and testing

2. Identify Blimps on images with labelling tools

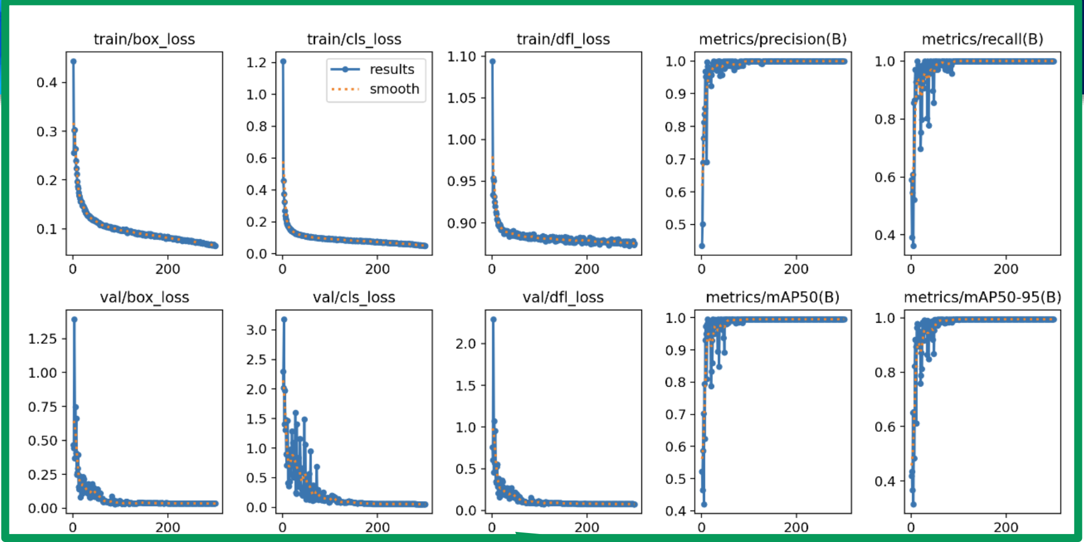


Number of photos used for different models

Training Object	Model Version	Training	Validation	Testing	Total
Generic Blimps	20240731-r	5595	69	230	5894
	20240731-py	5595	69	230	5894
Specific Blimp	20240802-py	12	1	1	14
	20240806-py	184	30	10	224
	20240807-py	287	70	24	381
	20240808-py	600	100	54	754

WORKING FLOW

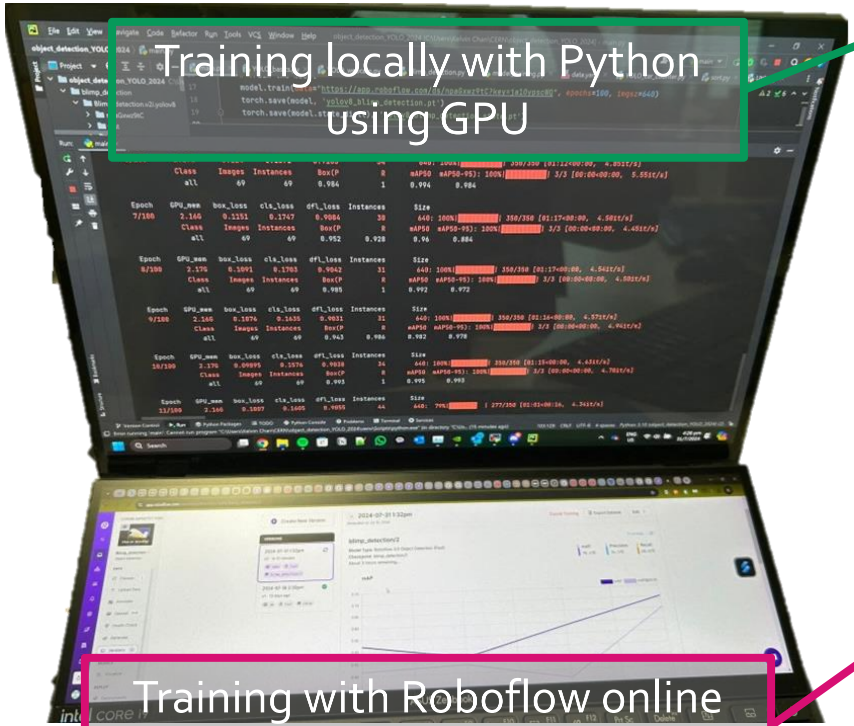
MODEL TRAINING



Model Training:

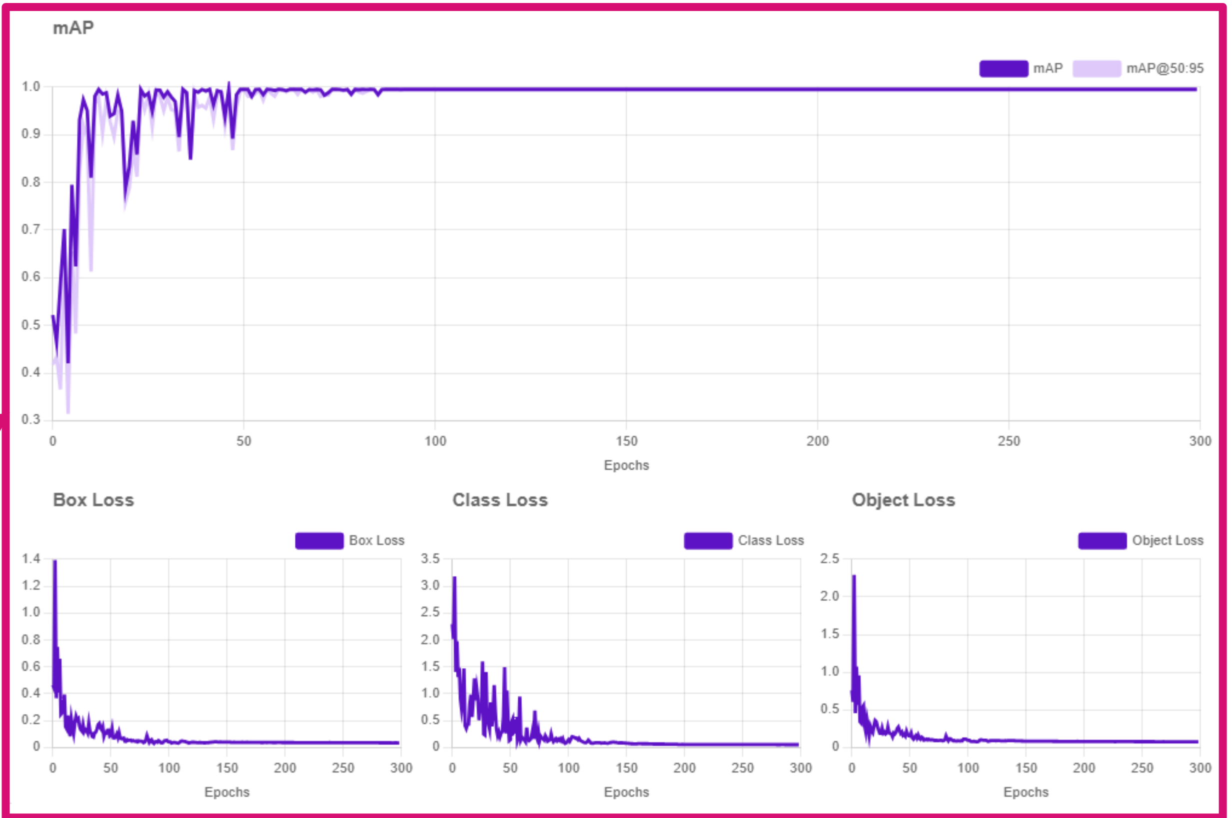
1. First model trained with Roboflow
2. Latest versions trained locally with YOLO with self-written Python scripts

Able to check training status in both

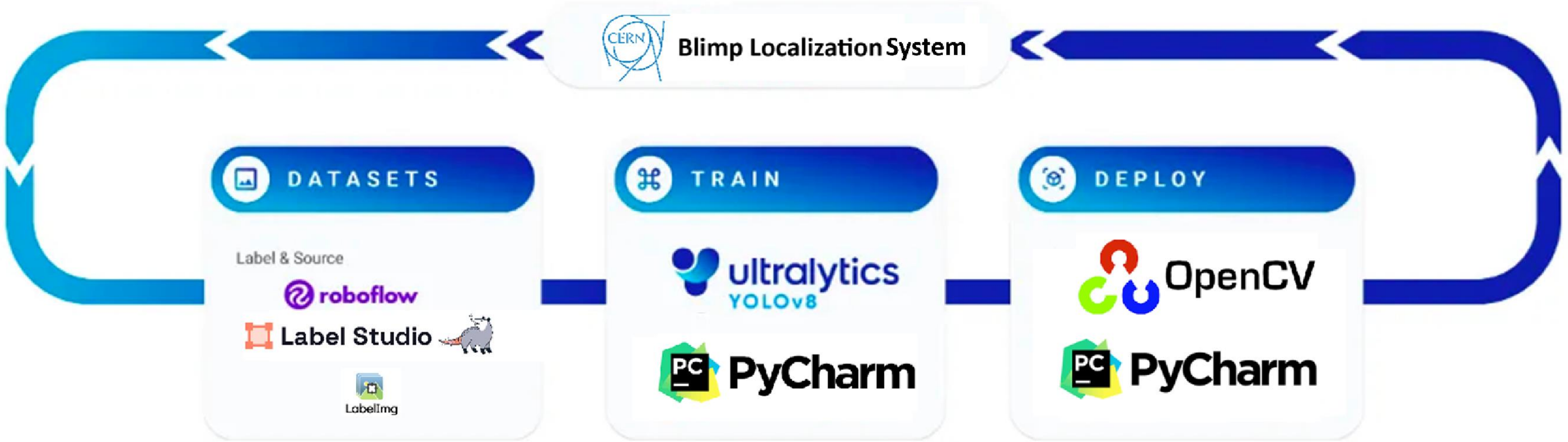


Training locally with Python using GPU

Training with Roboflow online platform



WORKING FLOW



RESULTS AND DISCUSSION

DEPLOY

OpenCV

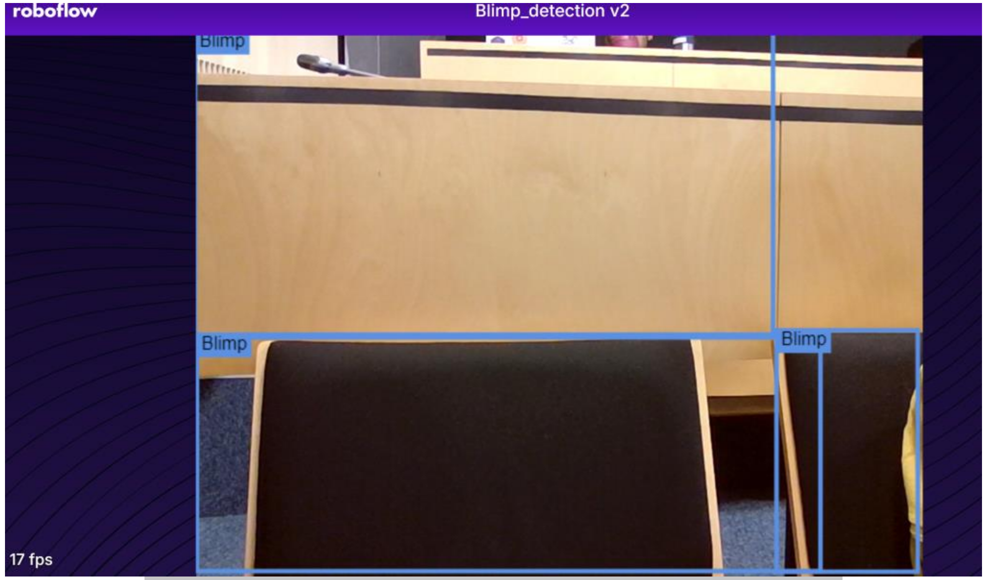
PyCharm

Results and discussions:

Models “20240731-r” (Roboflow) & “20240731-py” (Local)

- Same dataset: around **6000** generic Blimp images
- Performance: **Bad** although metrics look good
- Images are **too generic** so that the model mistakenly recognizes **everything as a Blimp**

mAP	Precision	Recall
99.5%	99.9%	100.0%



Model “20240731-r” (Roboflow)

Model “20240731-py” (Local)

RESULTS AND DISCUSSION

Results and discussions:

Model "20240802-py"

- Dataset: 14 specific Blimp images in ALICE cavern
- Performance: **Acceptable**
- **HIGH** Precision, **LOW** Recall, **LOW** F1

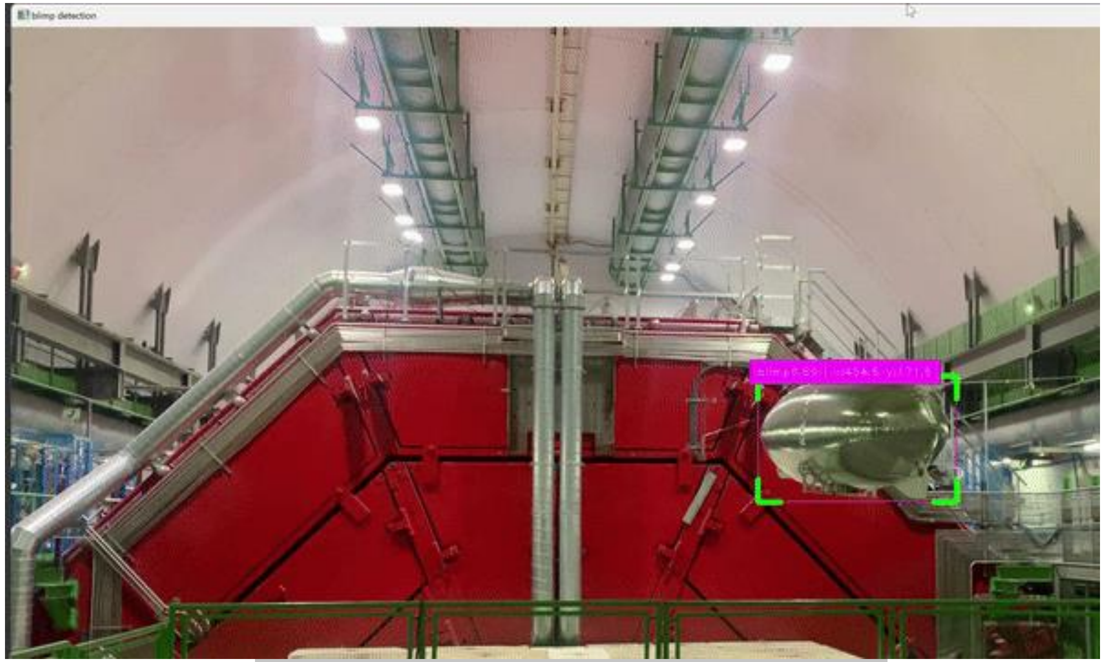
Model can recognize the Blimp but may lose track

Model "20240806-py"

- Dataset: around 200 specific Blimp images in ALICE cavern
- Performance: **Good**
- **HIGH** Precision, **Acceptable** Recall, **Acceptable** F1

Model can recognize the Blimp in 2022 videos with no issues

Model "20240802-py"



Model "20240806-py"

RESULTS AND DISCUSSION

DEPLOY
OpenCV
PyCharm

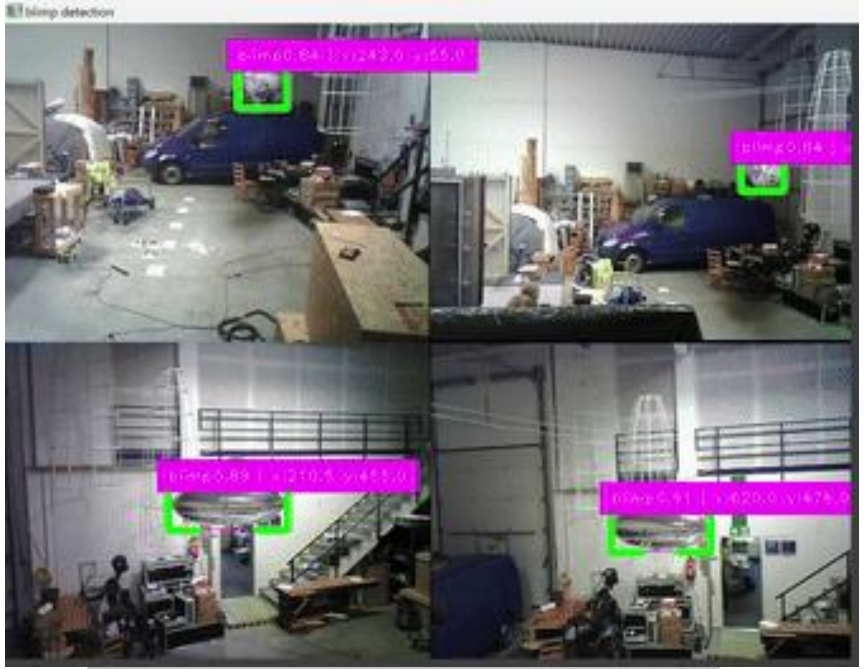
Results and discussions:

Model "20240807-py"
- Dataset: 400 specific Blimp images
- Performance: **Good**
- **HIGH** Precision, **HIGH** Recall, **HIGH** F1

Model can recognize the Blimp with different background

Model "20240808-py"
- Dataset: around 750 specific Blimp images
- Performance: **Very Good**
- **HIGH** Precision, **HIGH** Recall, **HIGH** F1

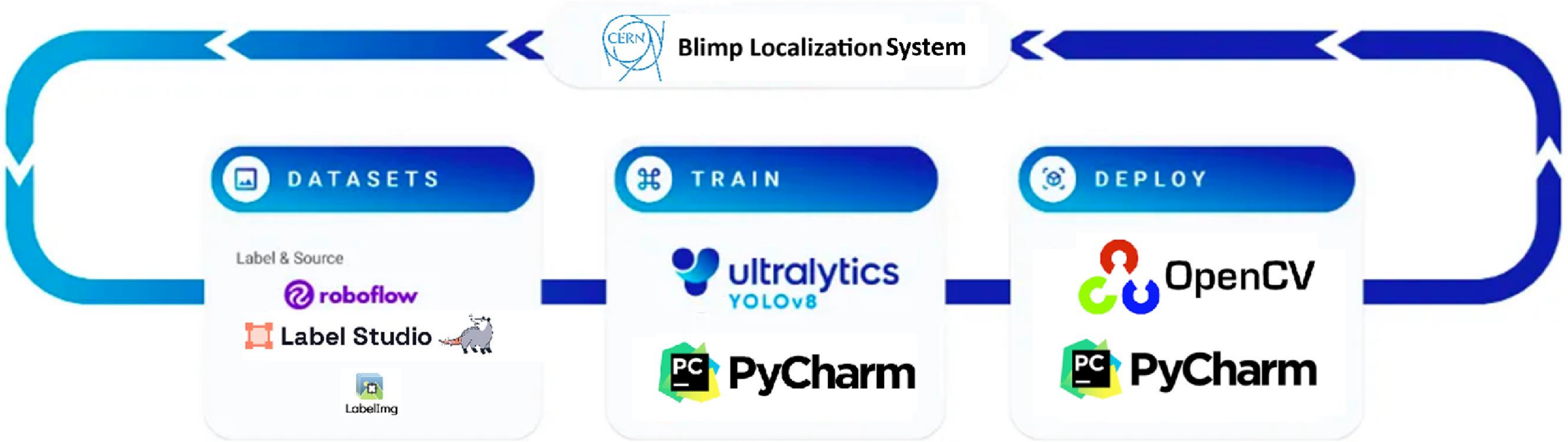
Model can recognize the Blimp even if partially visible & different lighting



Model "20240807-py"

Model "20240808-py"

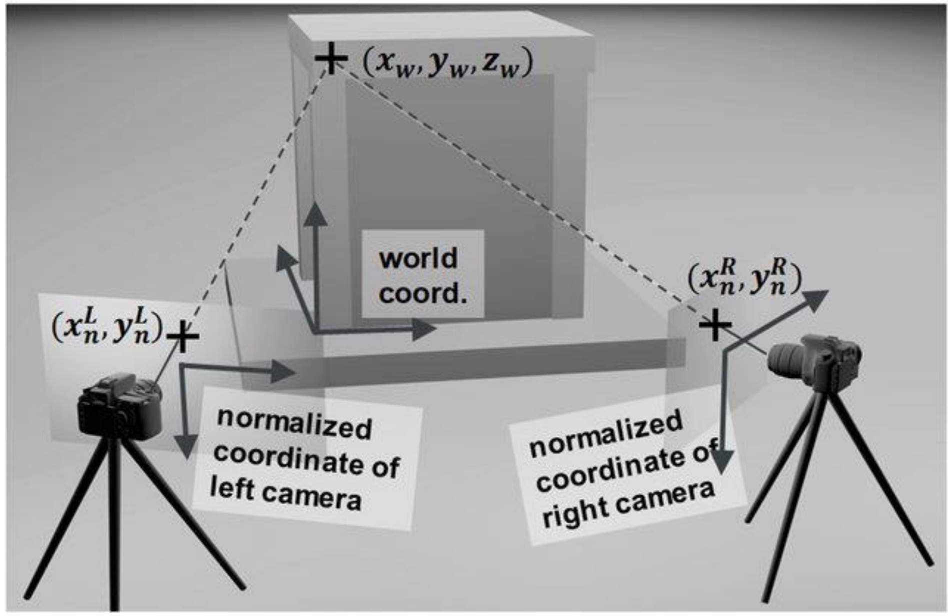
WORKING FLOW



LOCALIZATION METHOD

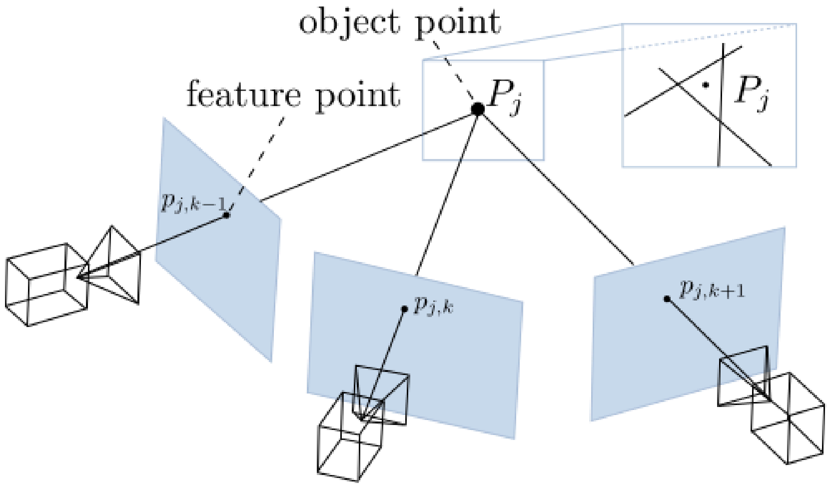
Blimp Localization System

- Transform multiple 2D image coordinates into real world coordinates [1]
- Use a Triangulation Algorithm to localize the Blimp



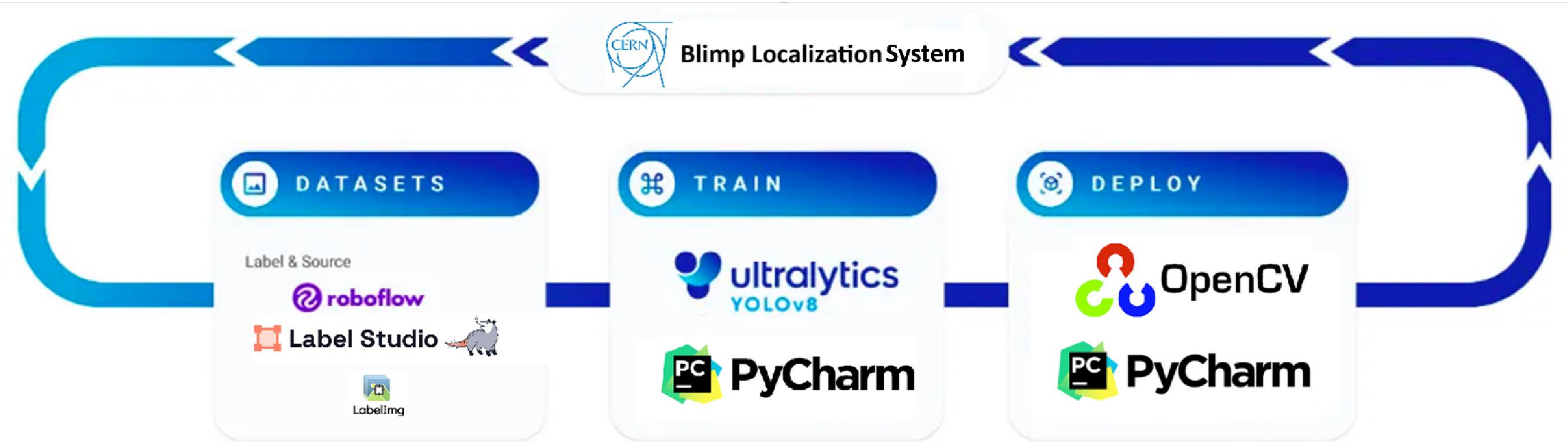
Triangulation Algorithm:

1. Input **camera parameters** (e.g., camera field of view, position, orientation, etc.)
2. Input **2D coordinates** (x-y position of the center of the bounding box around the detected Blimp) coming from the detection model from **2 cameras at least**
3. Find 3D triangulation point using **OpenCV [2]:**
   - If projections **intercept**, the intercept is identified as the **3D Blimp position**
   - Projections **may not intercept** due to **noise**, then the 3D Blimp position is identified as the **centroid** of the figure generated by the projection lines



[1] Photo: Measurement of Dynamic Responses from Large Structural Tests by Analyzing Non-Synchronized Videos. (2019). *ResearchGate*. https://doi.org/10.3390//s19163520
[2] OpenCV triangulation: OpenCV: Triangulation. (2024). Opencv.org. https://docs.opencv.org/4.x/do/dbd/group__triangulation.html

FUTURE DEVELOPMENTS



Future Developments:

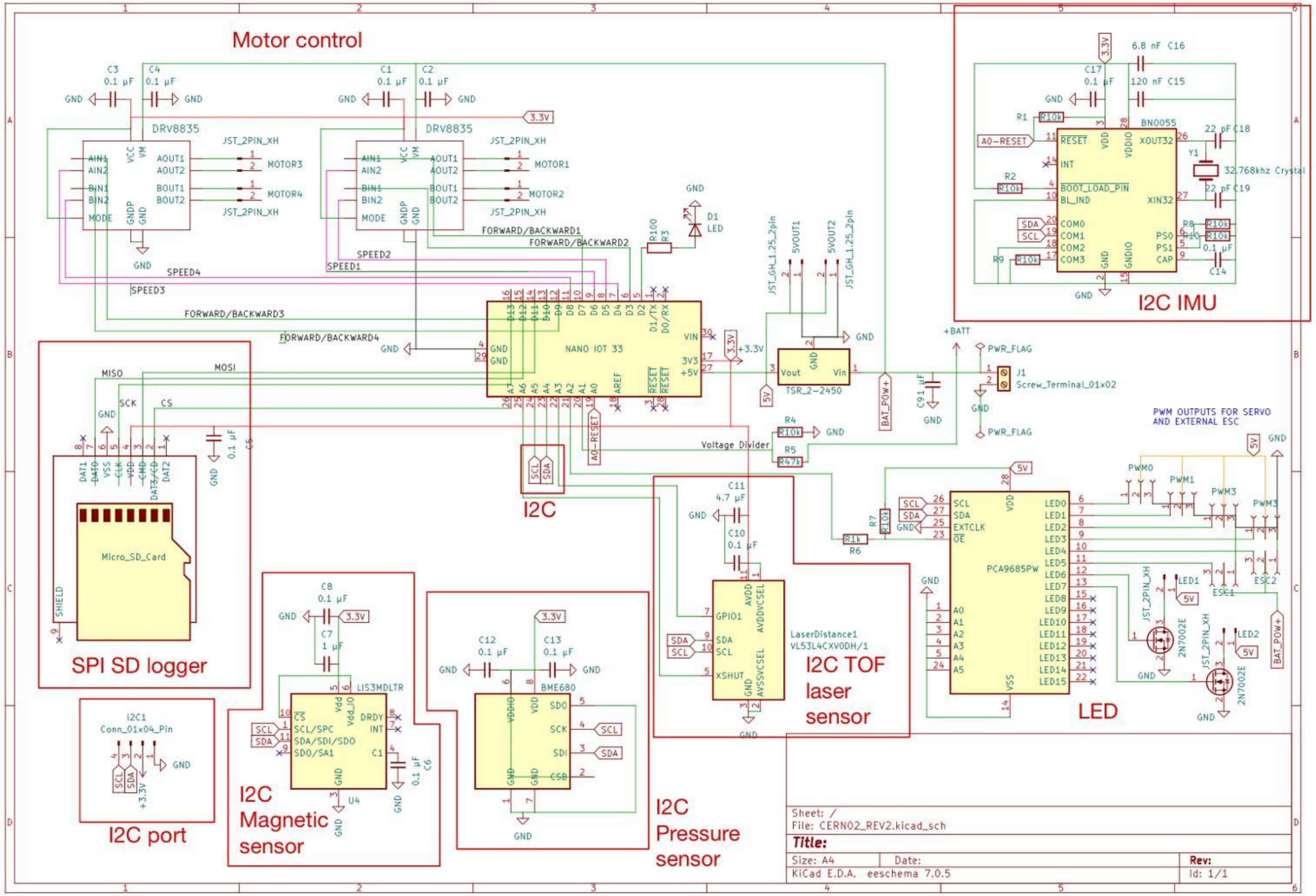
- Improve the detection model by putting the Blimp in the real environment with different lighting conditions
- Improve the detection model with real-time localization
- Test the accuracy of the detection model and localization system in simulation
- Real environment testing and camera calibrations to refine and increase the accuracy of the localization system

BLIMP PAYLOAD RADIATION SENSOR

ONBOARD COMMUNICATION

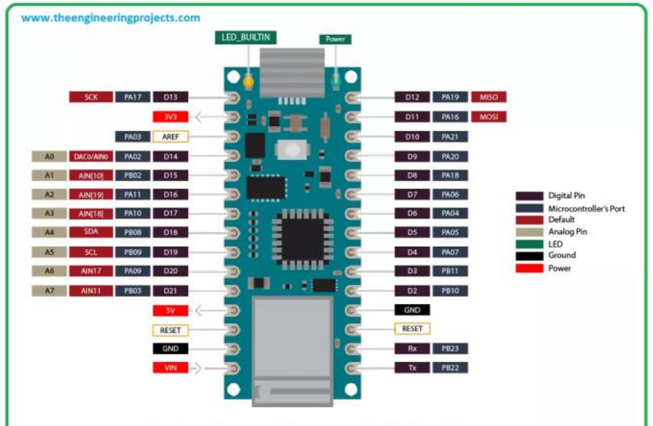
Blimp objective:

Monitor radiation and magnetic fields



Original Blimp communications

- Most sensors on the Blimp uses **I2C** protocol
- SD logger uses **SPI** protocol
- Need to add **radiation sensor**
- **No free pins available**



Arduino Nano 33 IoT

BACKGROUND INTRODUCTION

Objective:

Radiation Monitoring:

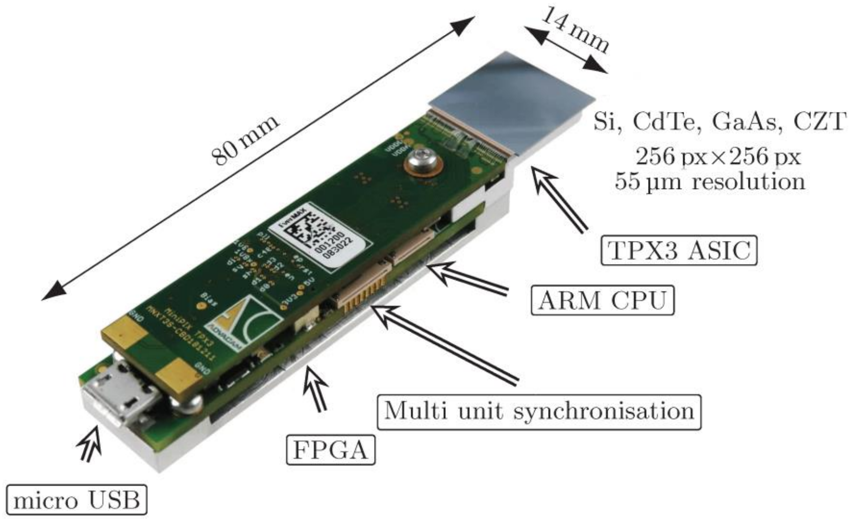
- Indicate which regions have **high radiation concentrations** that could **endanger individuals**

Blimp protection:

- Reduce **radiation exposure and sensors damage** by **avoiding high radiation areas**

MiniPix TPX3 Detector

- Weight: 41g
- Manufacturer: ADVACAM
- Chip: Timepix 3
- Sensor material: Silicon (Si)
- Sensor thickness: 500 µm
- Connectivity: micro-USB 2.0
- Operating mode: Time-of-Arrival(ToA), Time-over-Threshold (ToT)



ONBOARD COMMUNICATION

RaspberryPi

File Edit Format View Help

Index	Matrix	Index	ToA	ToT	FToA	Overflow
0	8704	8515971	23	8	0	
1	9222	8515971	39	16	0	
2	9742	8515972	33	15	0	
3	8962	8515971	37	17	0	
4	9226	8515972	34	18	0	
5	9744	8515972	35	11	0	
6	9485	8515972	42	24	0	
7	8965	8515971	50	21	0	
8	8966	8515972	31	26	0	
9	9225	8515971	60	15	0	
10	8449	8515970	45	10	0	
11	8963	8515971	37	23	0	
12	9743	8515972	44	16	0	
13	9482	8515972	38	24	0	
14	9484	8515972	50	25	0	
15	8964	8515971	61	21	0	
16	9223	8515971	50	16	0	
17	9224	8515971	71	21	0	
18	8705	8515970	52	13	0	
19	8706	8515970	58	15	0	
20	9483	8515971	43	11	0	
21	9486	8515972	35	13	0	
22	8192	8515970	48	1	0	
23	8448	8515970	136	13	0	
24	5272	47454362		3	9	0
25	20174	56169263		2	14	0
26	20672	65701144		2	9	0
27	1250	88720733		69	18	0
28	994	88720733		54	17	0
29	38404	162659485		5	25	0

Data-driven measurements

Onboard communication with a Raspberry Pi:

- Weight: 46g
- Able to install the PixetPro ARM64 API package
- Able to secure shell (ssh) to communicate with ground control station
- Able to acquire frame measurement with a Python script
- Able to read out data-driven measurements with a Python script

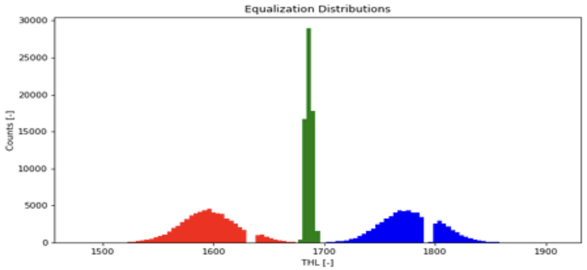
Raspberry Pi model 4B
Architecture: ARM64 (64bit)
OS: Ubuntu Desktop 23.10

RADIATION DOSE

Procedures of getting the Radiation dose :

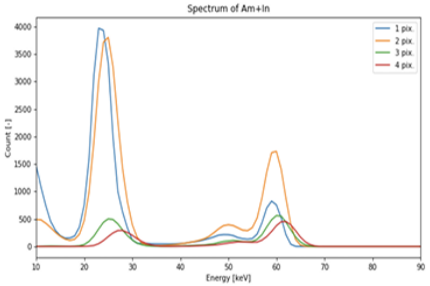
1.Calibrate Sensor
ensuring each pixel of the detector will have the **same ToT with the same type of particles**

Threshold Equalization



Parameter	Value
Mean THL	1686.206
Std. Dev	7.422
Masked Pixels	26
Final THL	1628

Energy Calibration in ToT Mode

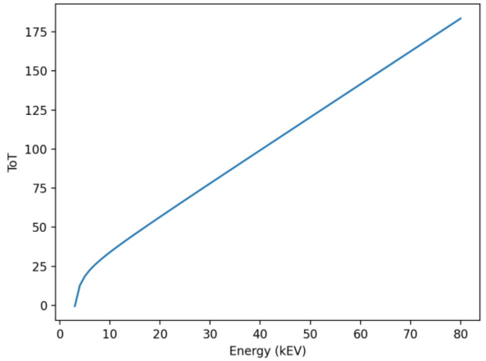


Source	Energy	1	2	3	4
Fe	5.725	5.124	8.106		
In	24.047	23.628	24.696	25.356	27.419
Am	59.499	59.378	59.748	60.546	61.734

Coefficient	Mean value
A	2.100
B	15.903
C	23.483
T	1.967

2. Energy Calibration
use **known sources** (Fe, In and Am) to construct the **"Energy to ToT graph"**

3. Analyze "Energy to ToT graph"
match **energy of each incoming particle** with ToT operating mode



4. Add up the **total energy** over a certain period

5. Calculate **effective dose** from total energy

With reference to ADVACAM Calibration documents

FUTURE DEVELOPMENTS

Future Developments:

- Use the Raspberry Pi Zero instead of the Raspberry Pi 4B to reduce the load on the Blimp
  (maximum payload of around 200g)
  46g -> 16g
- Blimp onboard ssh communication testing
- Radiation Dose Calibration of the sensor in real environment

FUTURE DEVELOPMENTS

Future Developments:

- Use the Raspberry Pi Zero instead of the Raspberry Pi 4B to reduce the load on the Blimp
  (maximum payload of around 200g)
  46g -> 16g
- Blimp onboard ssh communication testing
- Radiation Dose Calibration of the sensor in real environment

FUTURE DEVELOPMENTS

Future Developments:

- Use the Raspberry Pi Zero instead of the Raspberry Pi 4B to reduce the load on the Blimp
  (maximum payload of around 200g)
  46g -> 16g
- Blimp onboard ssh communication testing
- Radiation Dose Calibration of the sensor in real environment

OVERALL CONCLUSION

Overall Conclusion:

Blimp Localization System:

- Latest detection model **efficiently identifies the Blimps** in video frames
- The first developed detection models gave **False Positives**
- **Significant improvements** were obtained after refining the detection model
- The localization system must use **triangulation with multiple camera views**

Onboard radiation monitoring with MiniPIX TPX3 detector:

- A **Raspberry Pi** was used for data readout, due to **communication and weight limits**
- Implement the procedure to get the **dose rate from energy** using ToT measurements
- Refinements and real-environment testing are required

THANK YOU FOR YOUR ATTENTION

BACKUP SLIDES

You Only Look Once (YOLO):

- YOLO frames object detection as a **regression problem**, predicting bounding boxes and associated **class probabilities** directly from **full images in a single evaluation**
- To predict each bounding box, or object on the image, the YOLO algorithm **adopts characteristics from the entire image**.
- Additionally, it simultaneously predicts **all bounding boxes for a picture** in all of the classes.
- The image is divided into a **S X S grid**, and predictions are made for each grid cell regarding B bounding boxes, their corresponding confidence, and the probability of the C class. The tensor encoded with these predictions is **S*S* (B*5+C)**

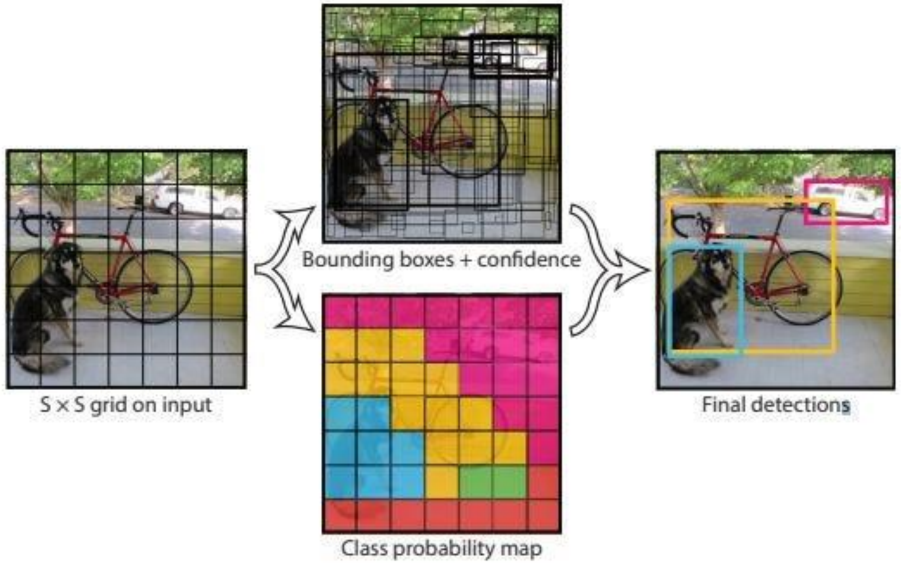


Figure 1: Working principle of YOLO

BACKUP SLIDES

Hybrid Silicon pixel detectors:

- Aluminum layer on top
- n-type implant just below the top surface,
- p-type implant pixels toward the bottom
- **High voltage** (positive) connection to the **aluminum surface**
- **Negative connection** to the **p-type** silicon layer

- Via a layer of **solder bumps**, a hybrid readout **ASIC (Application Specific Integrated Circuit) is attached** to the sensor component.

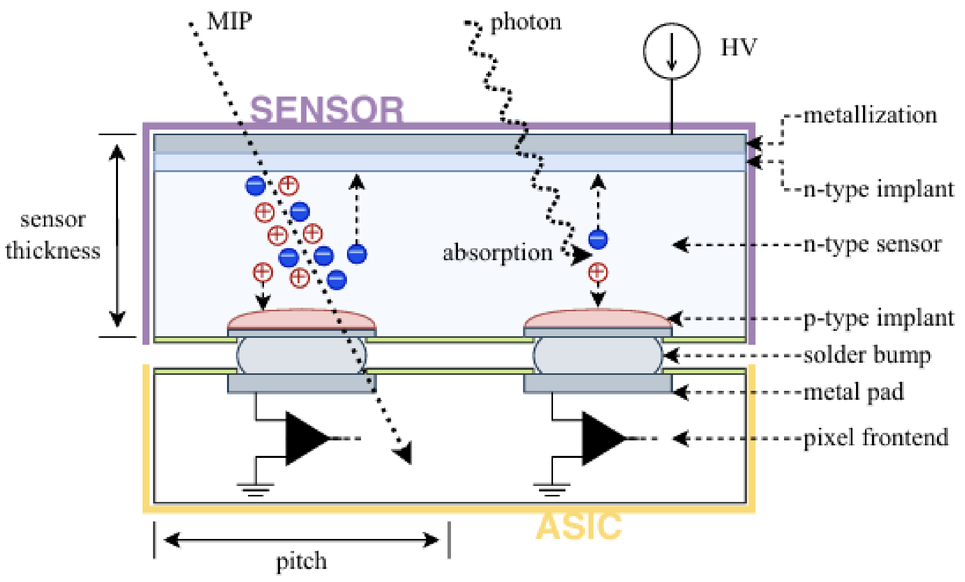
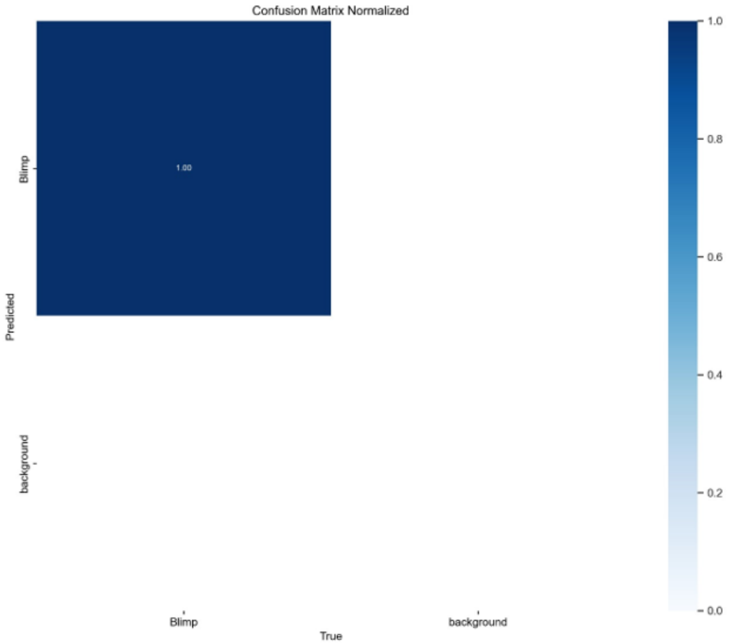
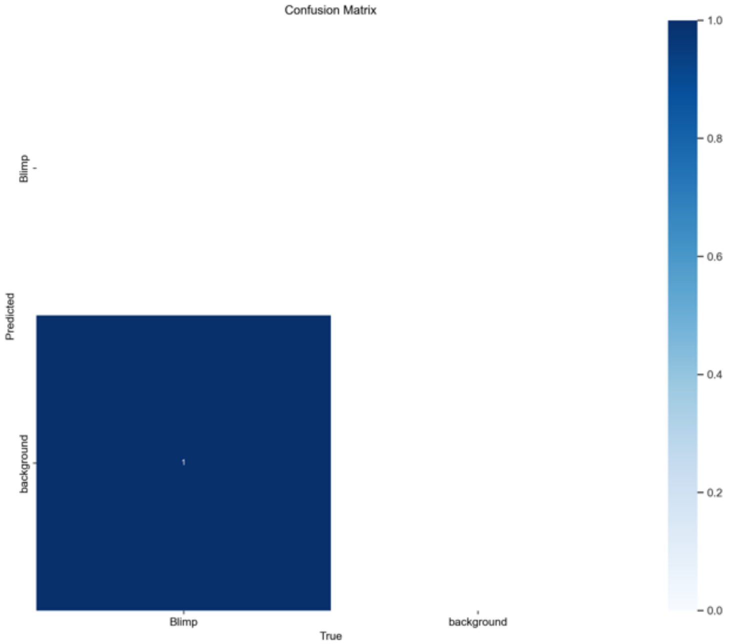
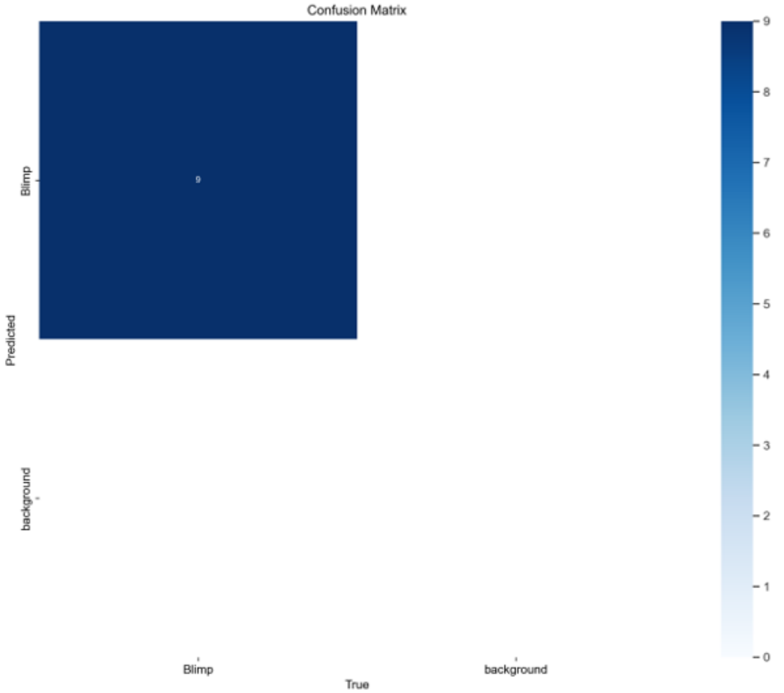
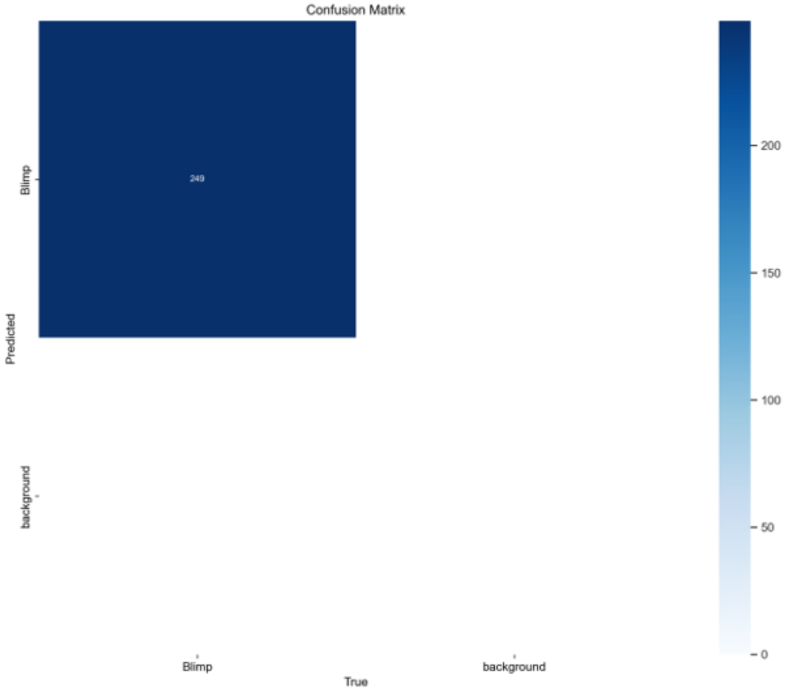
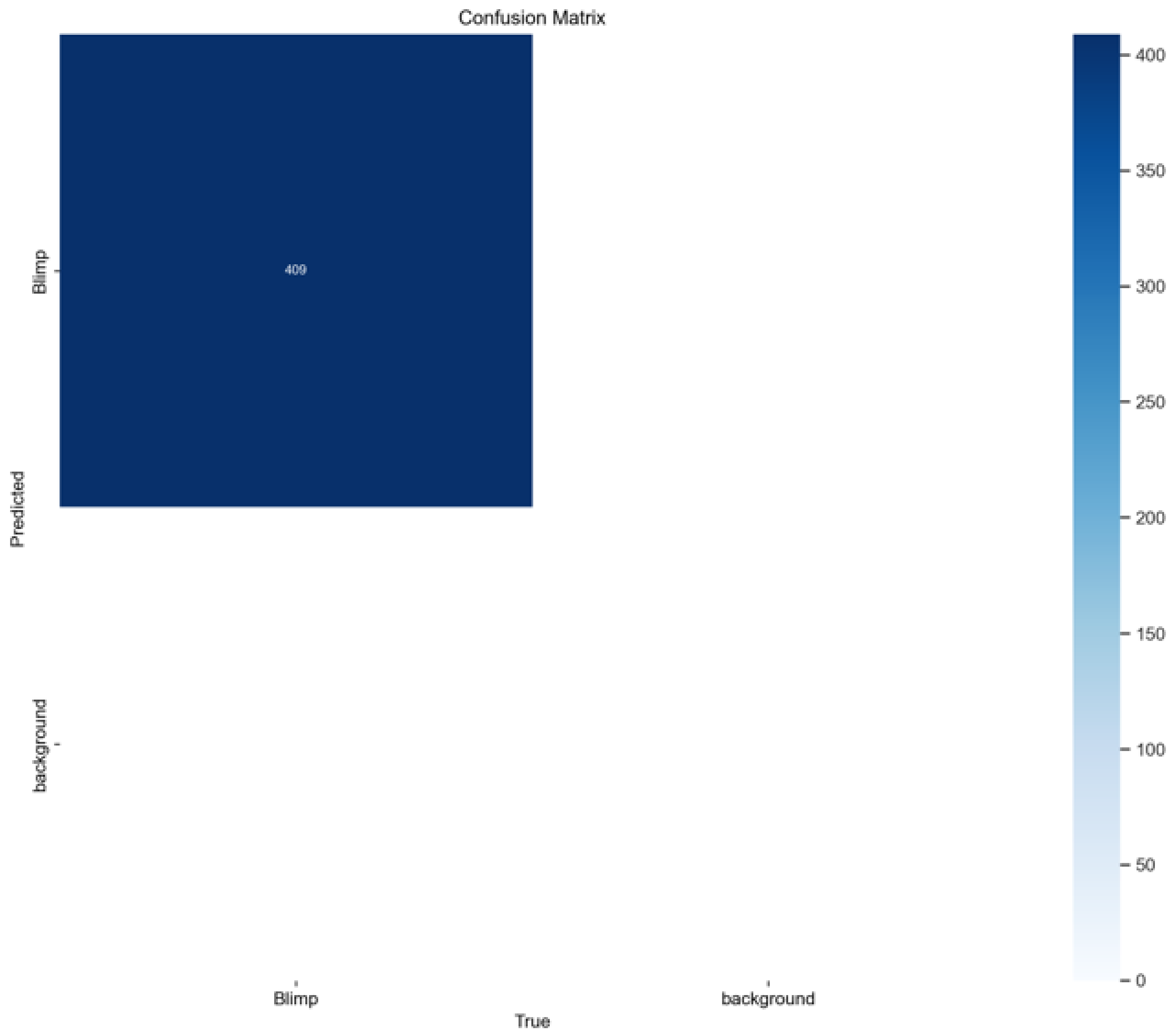


Figure 27:Schematic cross-section of a hybrid pixel detector

Models	Confusion Matrix
20240731-py	
20240802-py	
20240806-py	
20240807-py	

RESULTS AND DISCUSSION

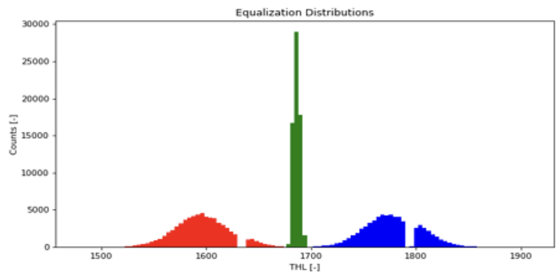
20240808-py	

Procedures of getting the Radiation dose :

With reference to ADVACAM Calibration documents

1.Calibrate Sensor
ensuring each pixel of the detector will have the **same ToT with the same type of particles**

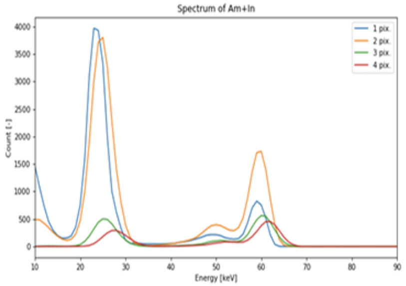
Threshold Equalization



Parameter	Value
Mean THL	1686.206
Std. Dev	7.422
Masked Pixels	26
Final THL	1628

2. Energy Calibration
use **known sources** (Fe, In and Am) to construct the **"Energy to ToT graph"**

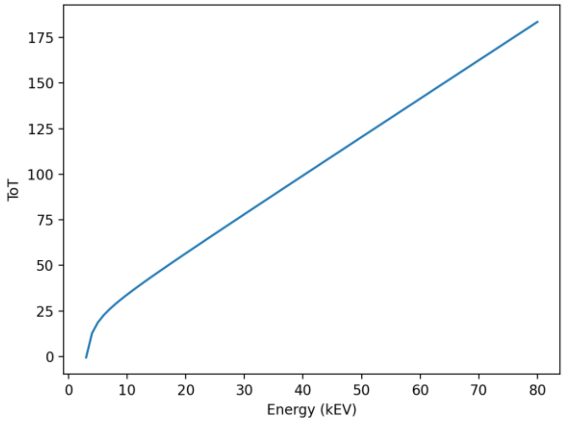
Energy Calibration in ToT Mode



Source	Energy	1	2	3	4
Fe	5.725	5.124	8.106		
In	24.047	23.628	24.696	25.356	27.419
Am	59.499	59.378	59.748	60.546	61.734

Coefficient	Mean value
A	2.100
B	15.903
C	23.483
T	1.967

3. Analyze "Energy to ToT graph"
match **energy of each incoming particle** with ToT operating mode



4. Add up the **total energy** over a certain period

5. Calculate **effective dose** from total energy