CrowdStop.AI

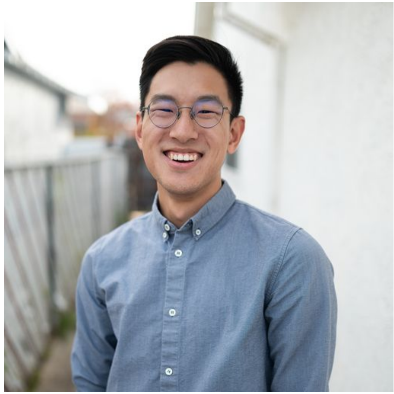
Final Capstone Presentation

Tae Kim, Jeremy Lan, Michelle Lee

Team Members

Tae Kim

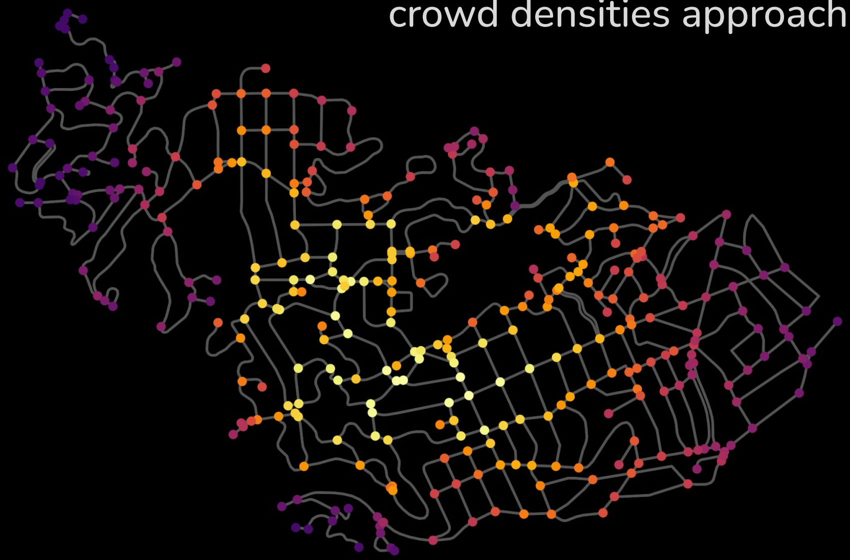
Jeremy Lan



Michelle Lee

Mission Objective

Implement an **crowd monitoring system** using a network of security cameras to automatically **detect and alert authorities in real-time** when crowd densities approach potentially critical levels in any given node



What is a crowd crush?

Magnitude of the Problem	Root of the Issue	Our Stakeholders

Magnitude of the Problem

6000+ injuries per year globally

Recent Crush Incidents (Deaths)

- 159 (South Korea, 2022)
- 135 (Indonesia, 2022)
- 2500 (Saudi Arabia, 2015)

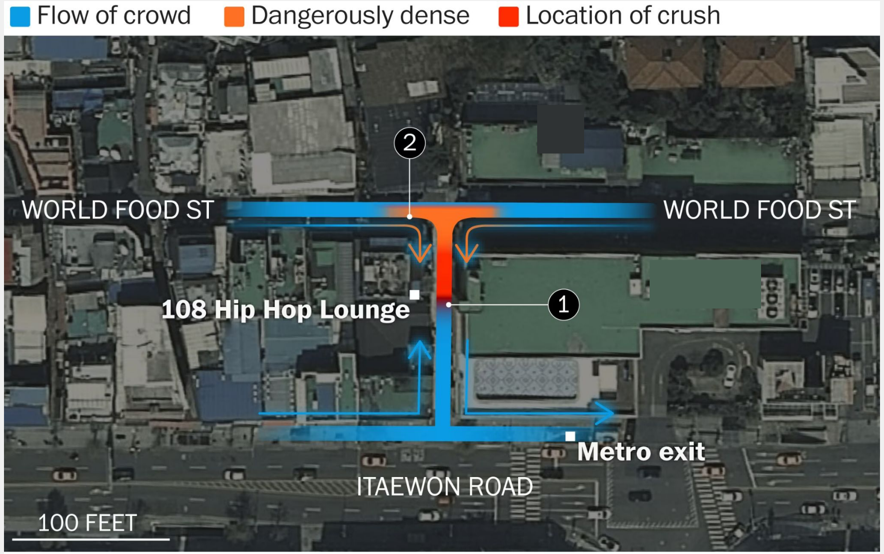
Root of the Issue

- Insufficient Event Security
- Poor management and planning
- **Inability to monitor and detect critical or near-critical situations**

Our Stakeholders

- Public Safety Officials
- Stadium Operators
- Law Enforcement

Case study: Seoul Halloween Crush 2022



Satellite © 2022 NAVER / SPOT / National Geographic Information Institute

- First concerned distress calls recorded at 6:34 PM
- Crowd crush occurred between 10:08 - 10:20 PM
- Emergency services unable to reach victims until 11:45 PM

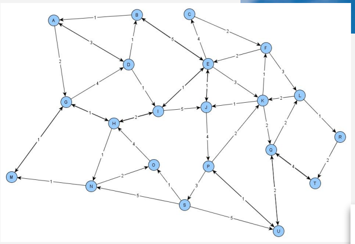
Plenty of time to alert authorities in advance to deploy security measures

Product description

Network of security cameras with edge computing units to detect and track pedestrian movement

Graph database tracking pedestrian movement across nodes

Web UI + Alert system to local authorities



Goal: Alert local authorities of potential danger before density reaches critical levels (7 people/m^2)

Advantages over Status Quo

	Current	Crowdstop.AI
Source	• Concerned bystanders • Security personnel	Security camera network
Information	Eye estimates	• Exact number of people • Direction and magnitude of movement
Scalability	Limited by number of personnel	Potentially infinite given enough security cameras
Monitored area	Only at observed areas	Able to infer densities at unobserved areas

Data - SOMPT22

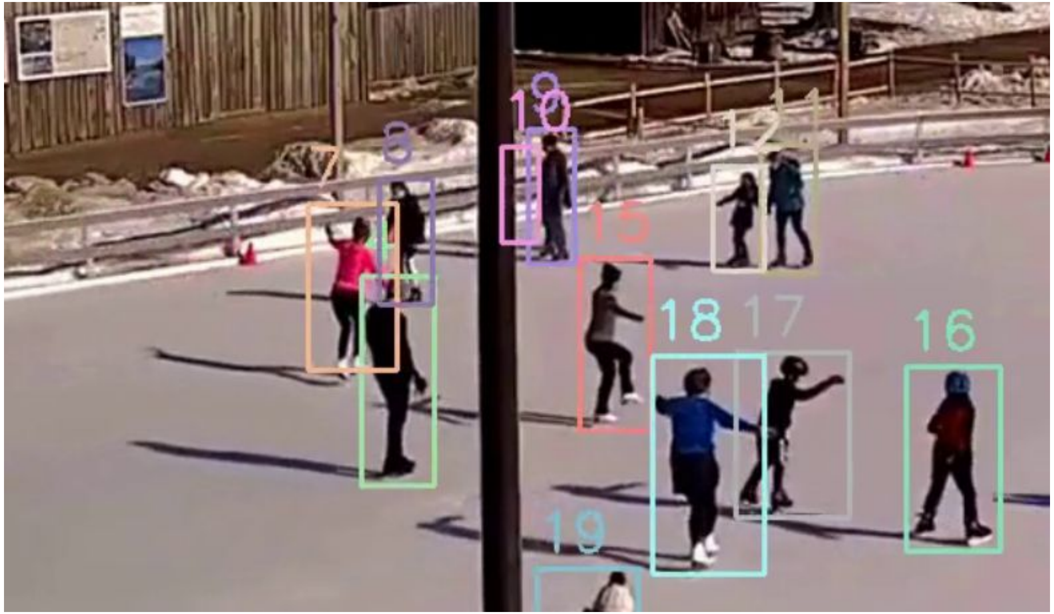
Model Training: SOMPT-22 Dataset

Dataset contains 14 “Scenes” consisting of video frames and a list of annotations

- Frame #
- Person ID #
- Bounding box
  (x, y, width, height)

Total Dataset:

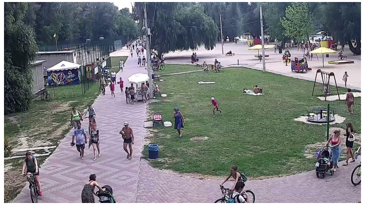
- 21k frames
- 800k annotations
- Average density: 37 people per image

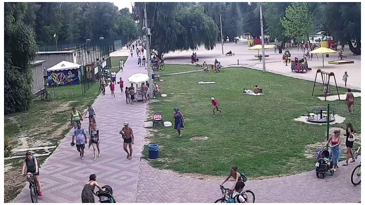


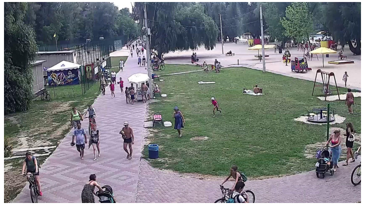
Object Detection & Tracking Model

Multiple Object Tracking

Video frames



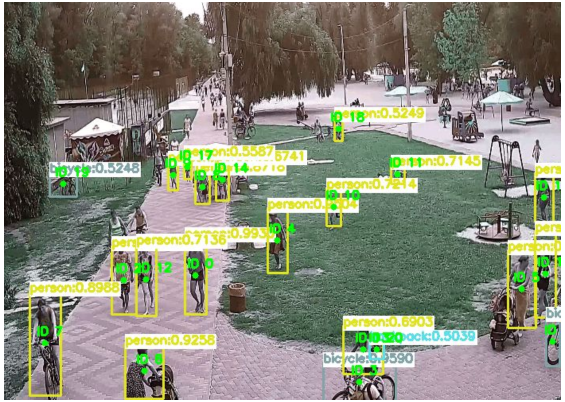




Object detection
(e.g. YOLOv3)

Bounding box +
classification

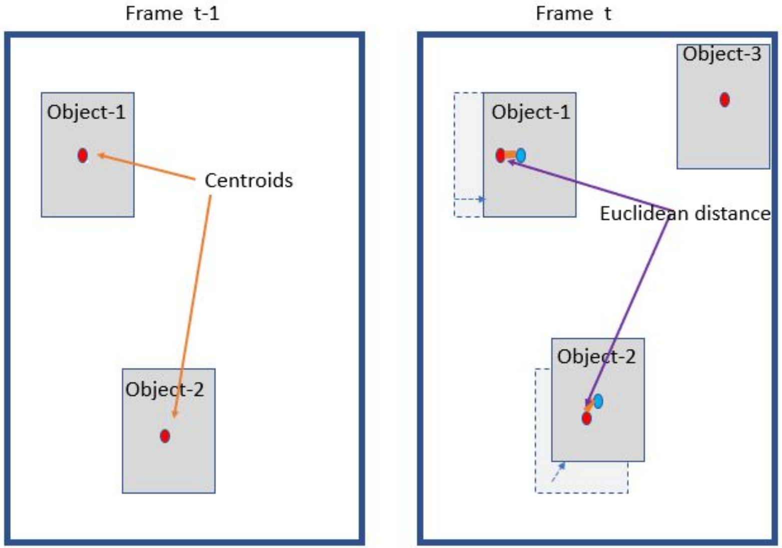
Object tracking
(e.g. centroid tracking)



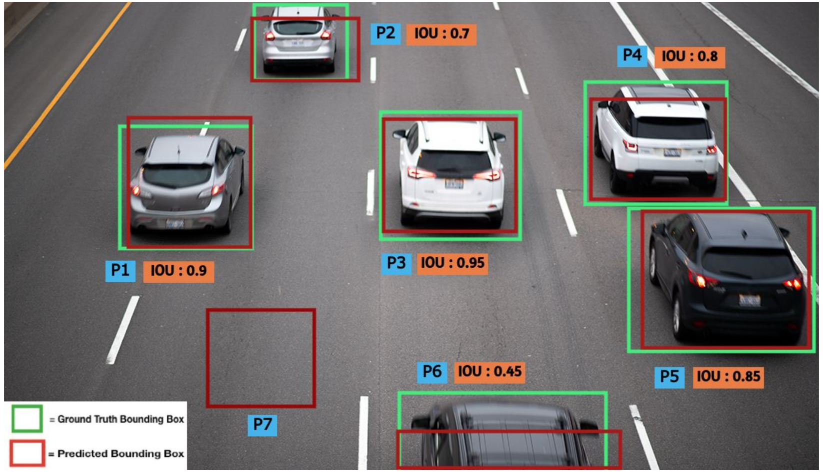
Bounding box + classification + object ID

Tracker Comparison

Centroid Tracker



IOU (Intersection over Union) Tracker



Model Performance Evaluator

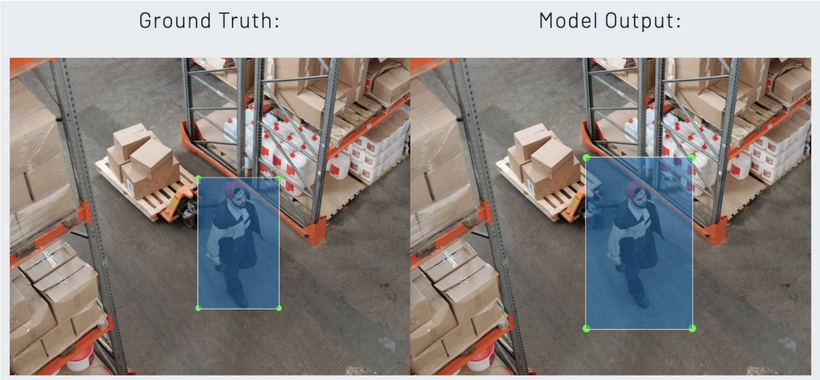
MOTA (Multiple Object Tracking Accuracy)

- Overall tracking accuracy metric

$$MOTA = 1 - \frac{\sum_t FN_t + FP_t + IDS_t}{\sum_t GT_t}$$

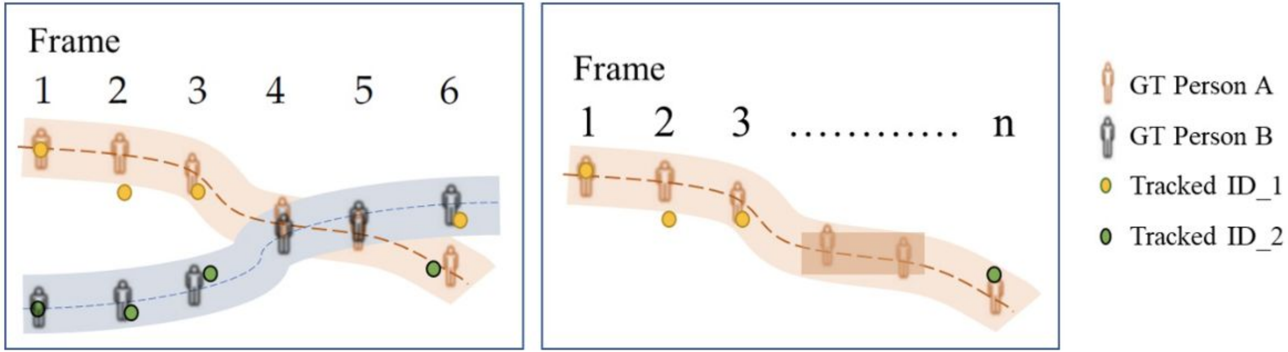
MOTP (Multiple Object Tracking Precision)

- Spatial precision of object tracking, measuring how closely the tracked object's positions match the ground truth positions
    - Avg distance between the centers of the two
    - Lower value indicates higher tracking precision



Model Performance - ID Switches

- ID Switch: incorrectly changing the ID of a trajectory
  - Left box: frames 4-5 where person A and B are not detected and result in ID switches in frame 6
  - Right box: lose track of person after frame 3, later identifying the person with a new ID



Evaluation Metrics: Object Detection

Using the first 50 out of 1800 frames for a sample video

Detector	Tracker	MOTA	MOTP	IDF1	ID Switches	Recall	Precision
YOLO	IOUTracker	0.200	0.274	0.323	26	0.270	0.818
YOLO	CentroidTracker	0.192	0.267	0.296	49	0.270	0.818
YOLO	CentroidKF_Tracker	0.185	0.267	0.263	68	0.270	0.818
YOLO	SORT	0.199	0.267	0.316	29	0.270	0.818
TF_SSDMobileNetV2	IOUTracker	0.006	0.313	0.096	13	0.077	0.537
TF_SSDMobileNetV2	CentroidTracker	0.003	0.313	0.085	21	0.077	0.537
TF_SSDMobileNetV2	CentroidKF_Tracker	0.0003	0.313	0.081	28	0.077	0.537
TF_SSDMobileNetV2	SORT	0.007	0.313	0.100	10	0.077	0.537

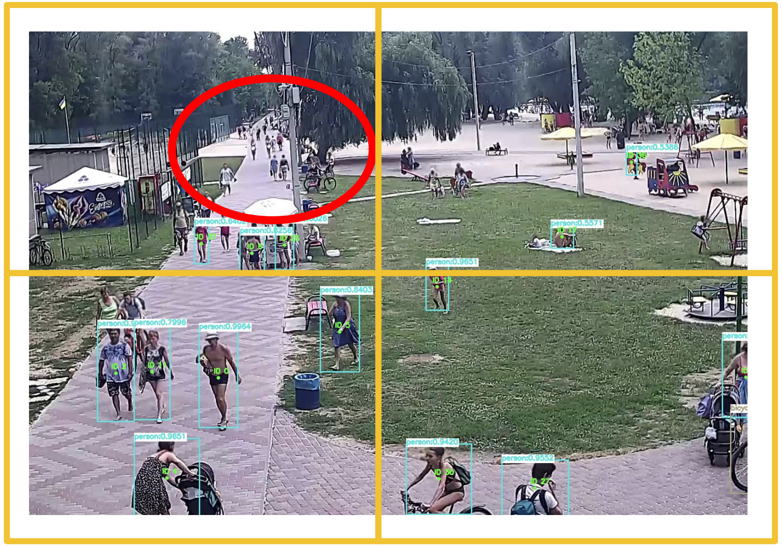
QuadYOLO

Previously struggled with low YOLO sensitivity to identify lower-resolution / smaller objects

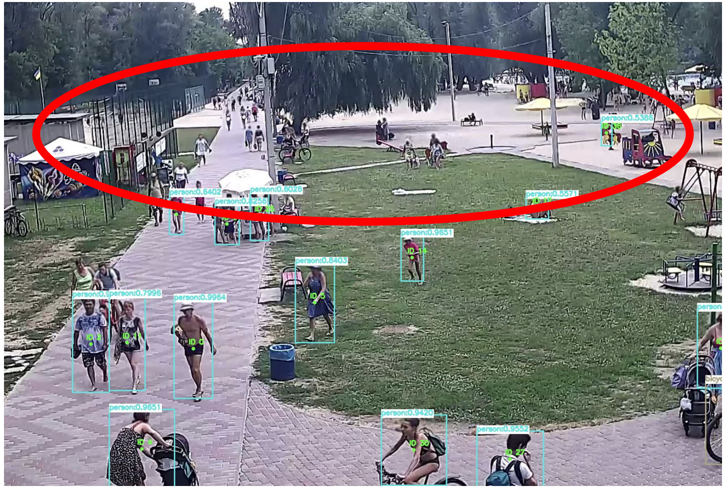
- Backgrounds of image vulnerable

Enhance YOLO detection component:

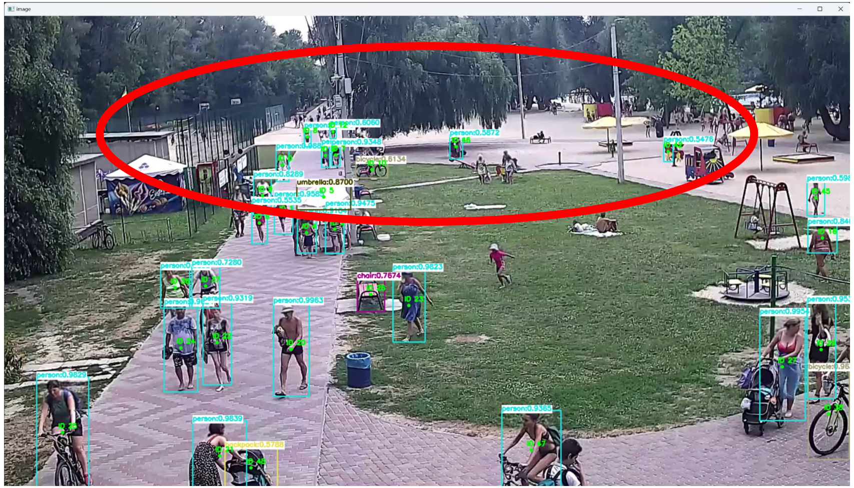
1. Divide image into quadrants
2. **Run YOLO detection to obtain bboxes**
3. Concatenate bbox IDs across entire image
4. Object Tracking proceeds as normal



Improving detection: YOLO vs QuadYOLO



YOLO, IOUTracking



QuadYOLO, IOUTracking

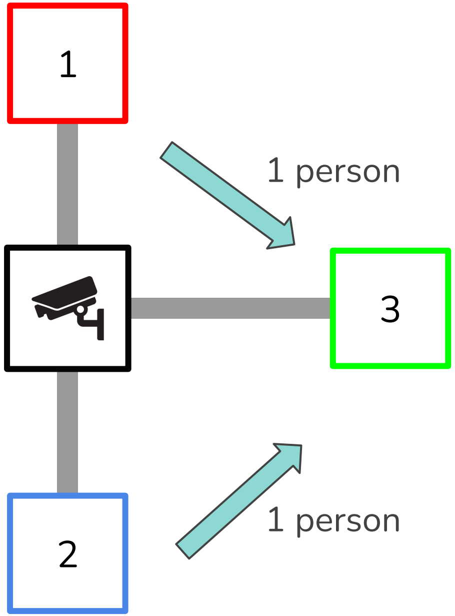
QuadYOLO Evaluation Metrics

Using the first 50 out of 1800 frames for a sample video

Image	Detector	Tracker	ID Switches	MOTA	MOTP	IDF1	Recall	Precision
Original	YOLO	IOUTracker	26	0.200	0.274	0.323	0.270	0.818
Quadrant Splitting	YOLO	IOUTracker	21	0.251	0.270	0.483	0.413	0.728

Tracking Movement across Scenes

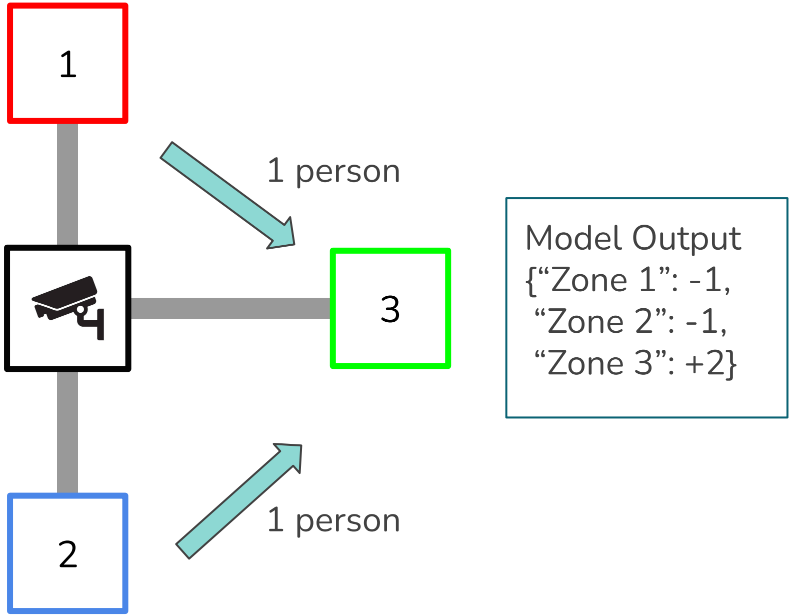
What counts as “movement”?



Model Output
{“Zone 1”: -1,
“Zone 2”: -1,
“Zone 3”: +2}

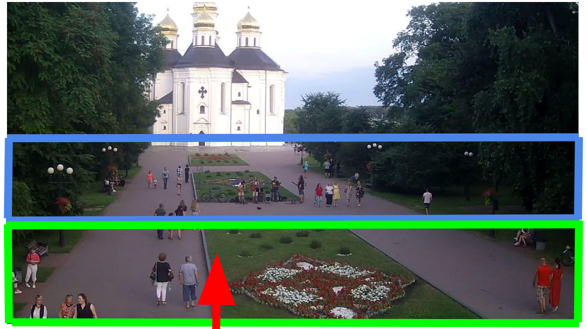
Implementation

- Zone boundaries manually configured
  - Areas of interest / pathways
- JSON upload provides zone boundaries for each scene
- Each zone records change in pedestrian IDs over a time period



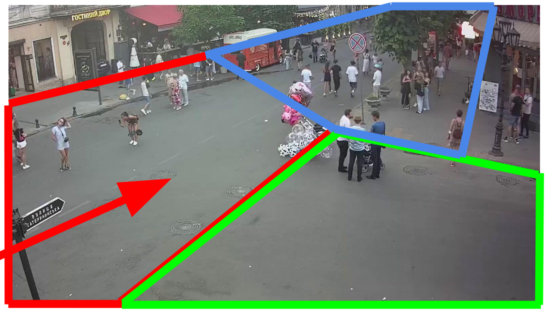
Zone-Linking Relevant Scenes

Scene 5

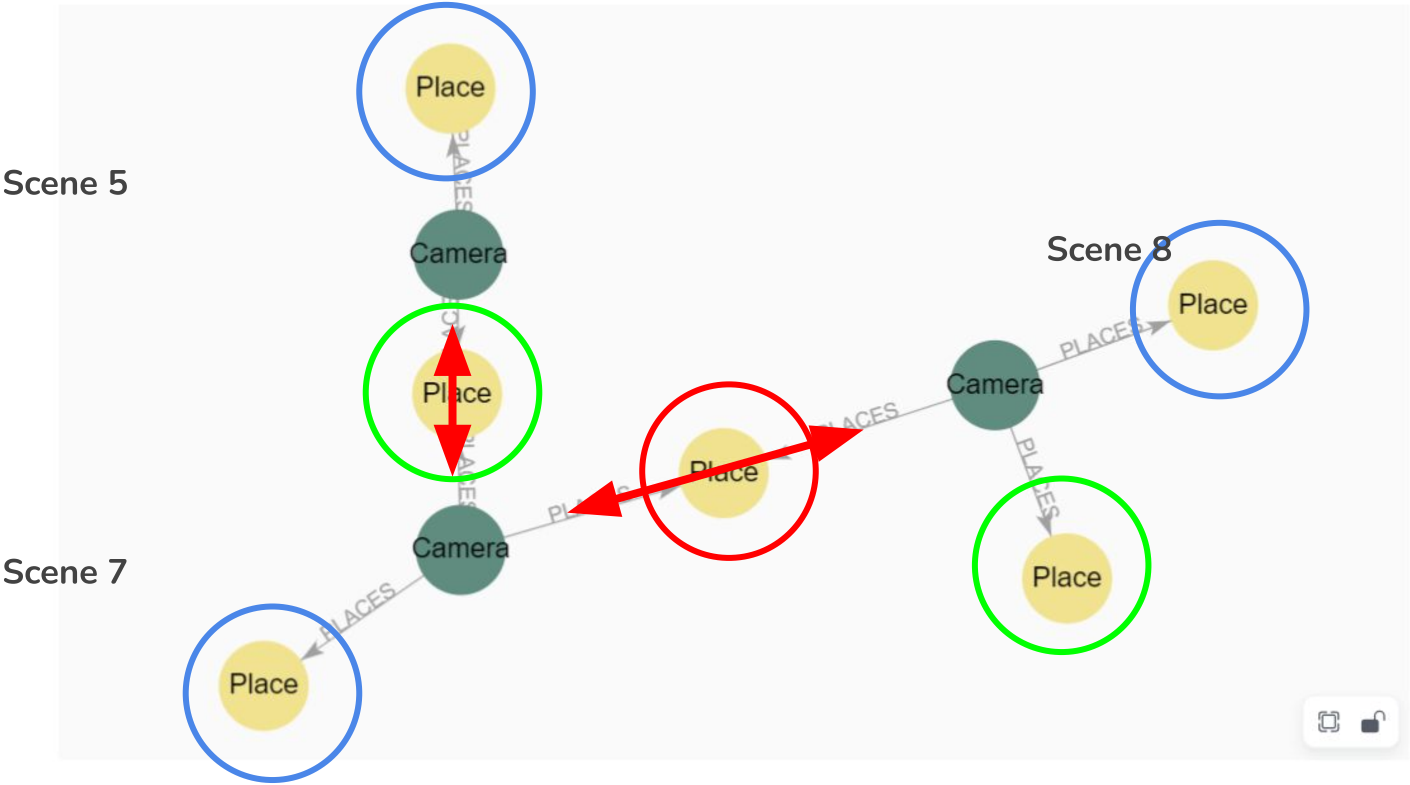


Scene 7

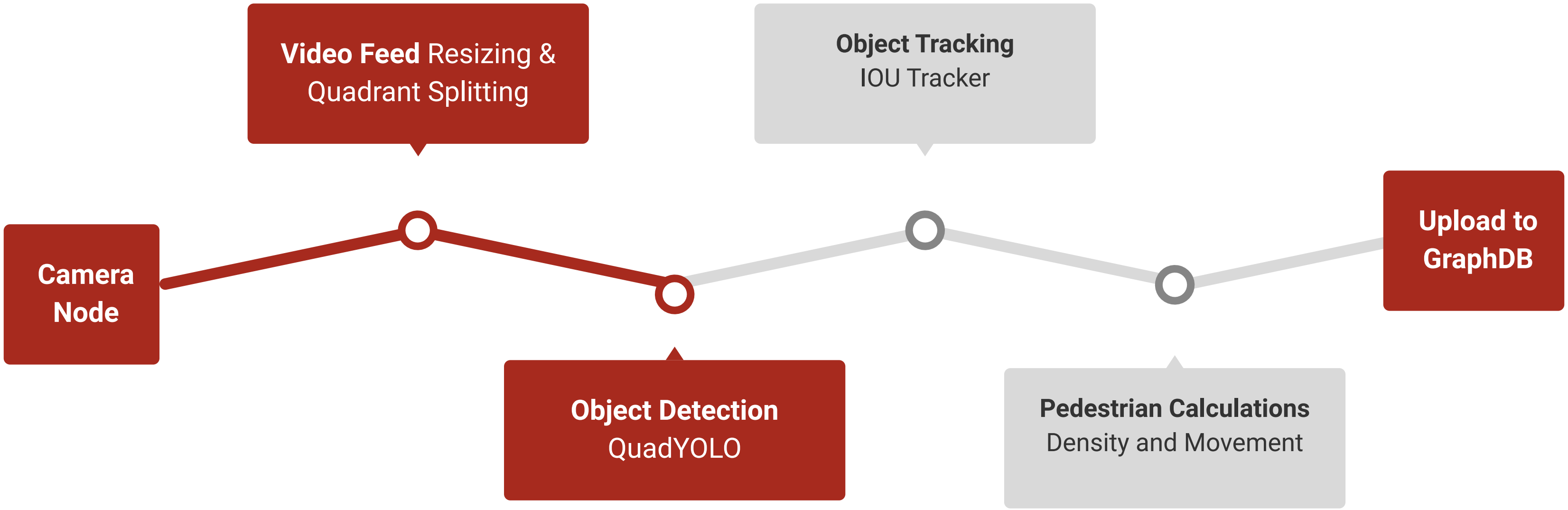
Scene 8



Zone-Linking Relevant Scenes



Multiple Object Tracking Pipeline Summary

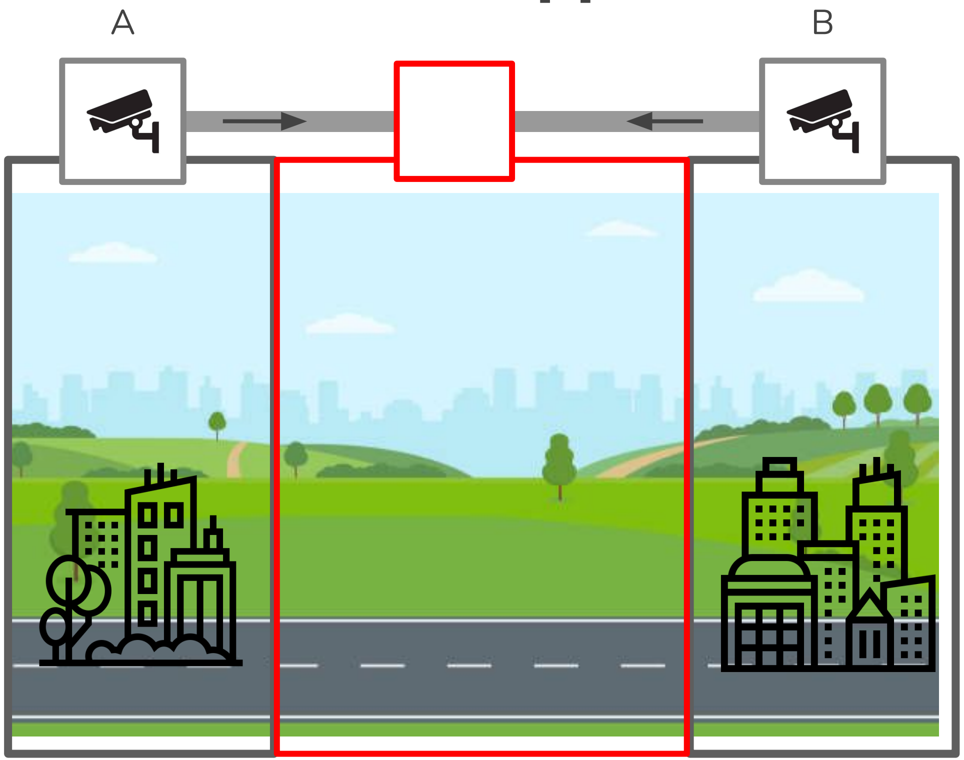


Designing the Graph Database

How to represent info in Graph DB?

We start out with:
Each camera = node

Accumulation could happen in unobserved area

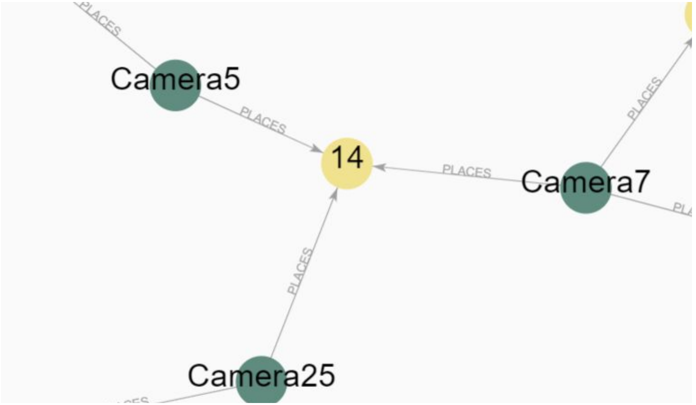


Need new node for unobserved areas

Observed and unobserved nodes

At each node, we track:

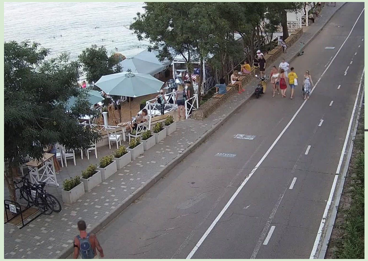
- **Metadata:** Unique ID, Name, Latitude & Longitude, Walkable Area, Distance from Adjacent Nodes
- At Observed Nodes: People Count (**direct** from camera)
- At Unobserved Nodes: **Predicted** People Count (inferred from crowd movement)



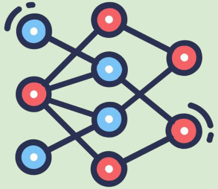
At each edge, we track movement of people from one node to another

System Design

Camera-side system design



Live video stream
(video file for POC)



MOT model

Edge computing unit

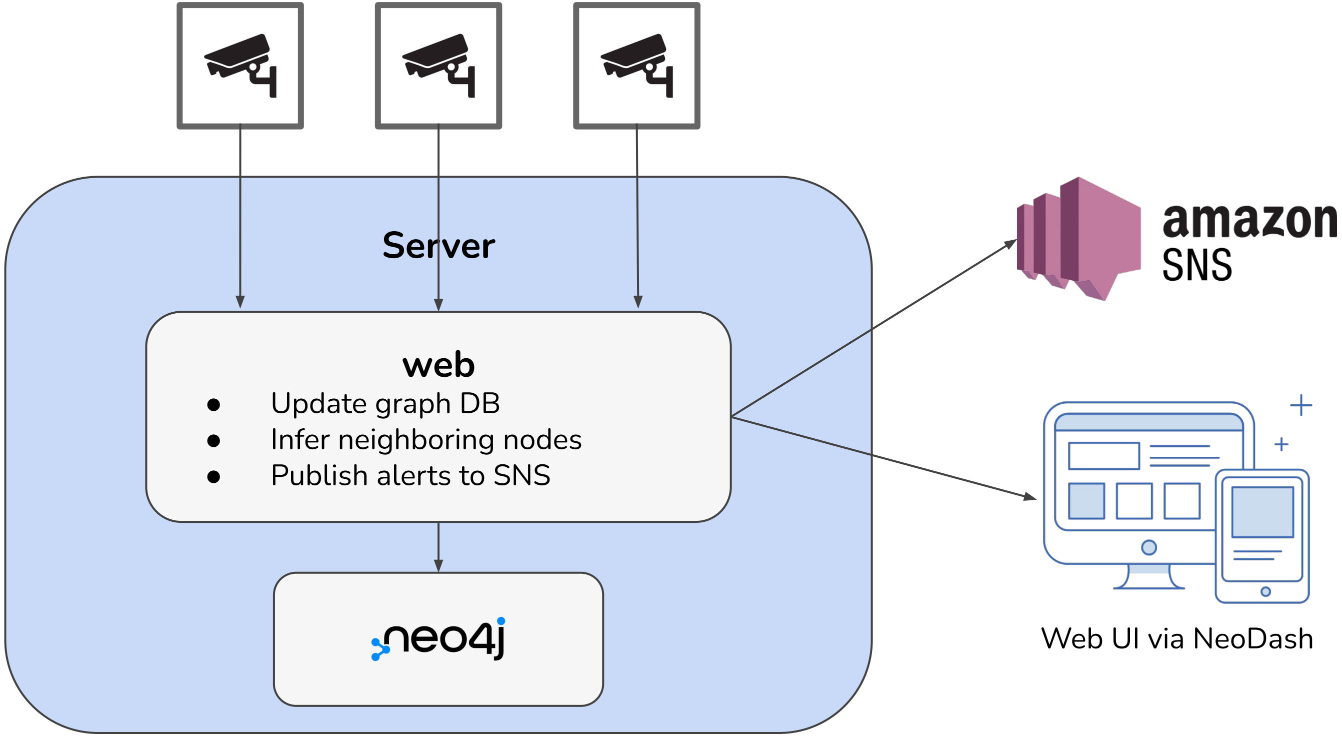
For each frame:

1. Filter for category == ”Person”
2. Calculate density within frame
3. Track movement

Every x frames, send updates to server

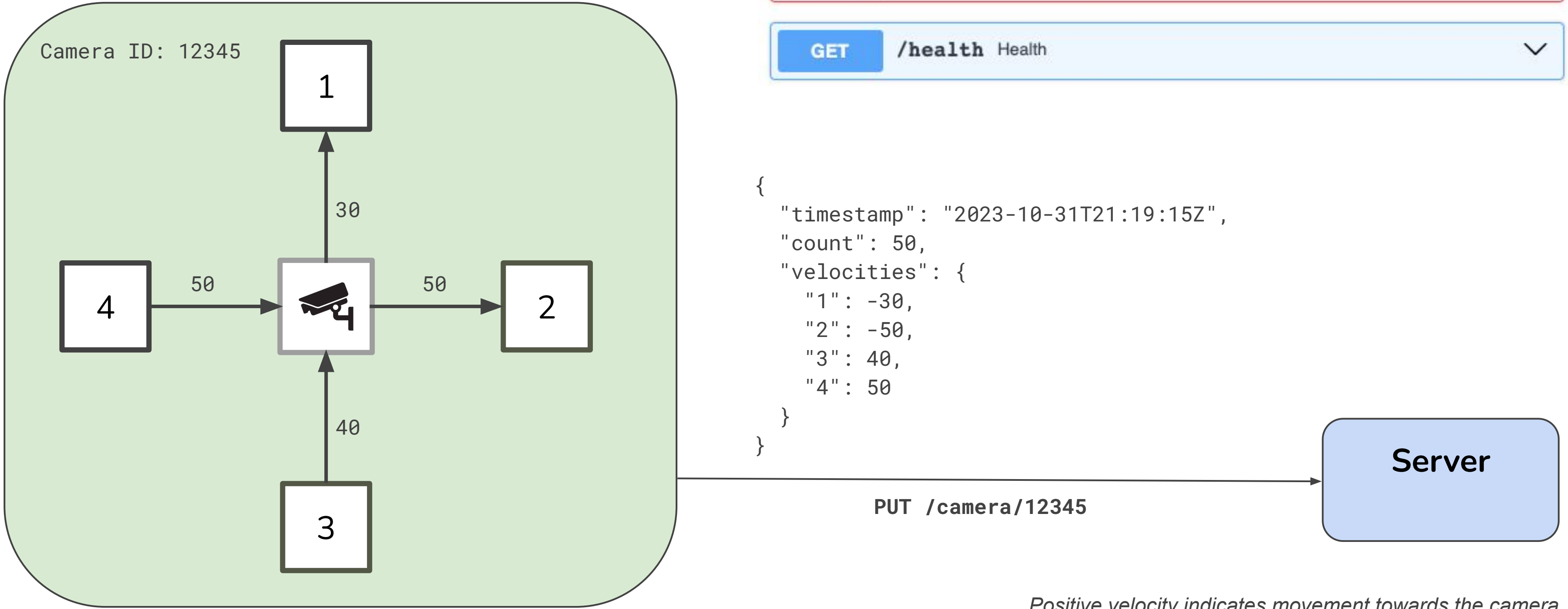
Server

Server-side design overview



API spec

POST	/camera	Create Camera
PUT	/camera/{camera_id}	Update Camera
DELETE	/camera/{camera_id}	Delete Camera
GET	/health	Health




```json
{
  "timestamp": "2023-10-31T21:19:15Z",
  "count": 50,
  "velocities": {
    "1": -30,
    "2": -50,
    "3": 40,
    "4": 50
  }
}
```


Positive velocity indicates movement towards the camera

Optimizing performance: Downsampling

Model Metrics

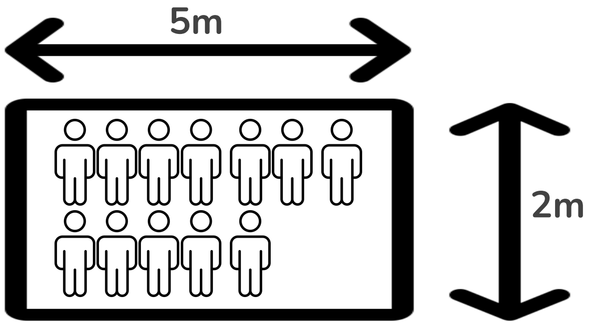
Frame Count Cadence	Recall	IDsw	Ground Truth	IDsw/GT
1	0.432	88	10839	0.81%
3	0.427	70	3627	1.93%
5	0.411	87	2167	4.01%
10	0.319	40	1085	3.7%

Front-end visualization & UX

NeoDash Metrics

Density

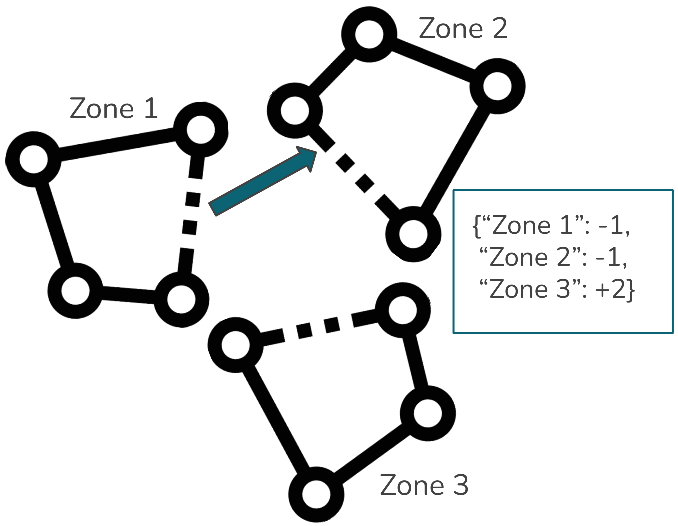
$$= \frac{\textit{Number of People}}{\textit{Area of Interest}}$$



Area of Interest: 10 m^2
Number of people: 12
Density: 1.2 people / m^2

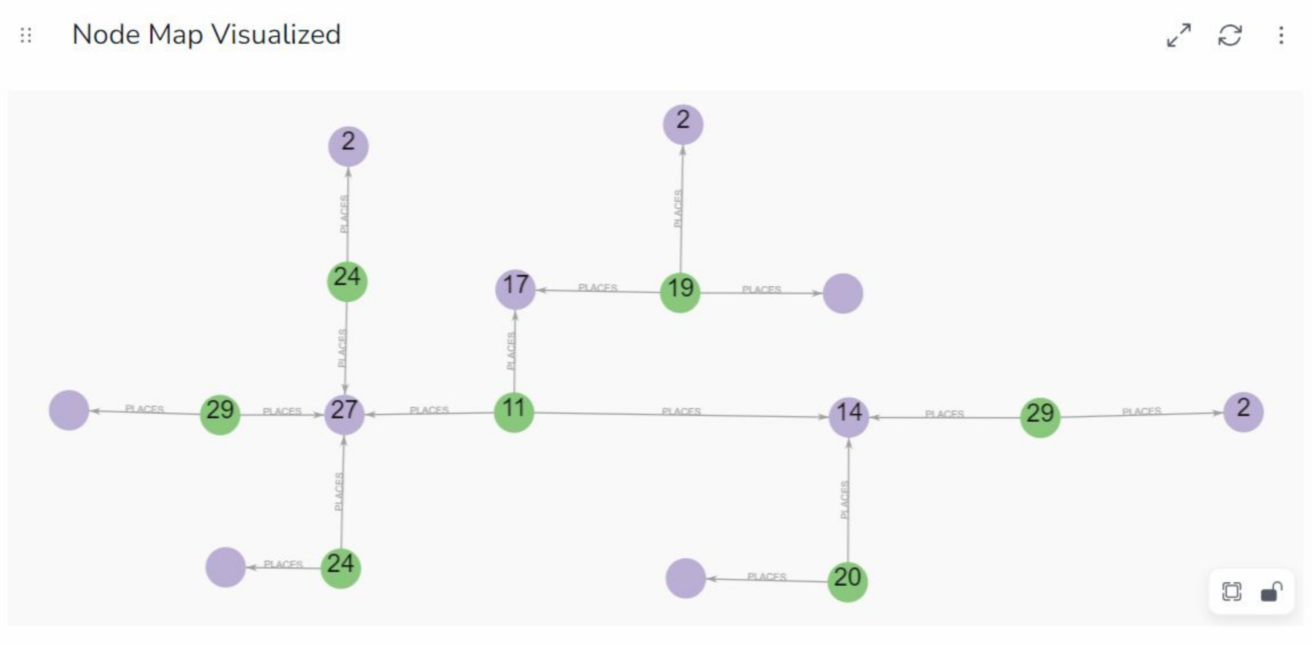
Velocity/Movement

= Dictionary of movement across zones



NeoDash Visualization Features

- Holistic View Node Map: Observed + Unobserved Regions



NeoDash Visualization Features

- Population and Density Per Node (Observed)
- Population and Density Per Node (Aggregated with Nearby Unobserved Regions)

Population per node + Adjacent Nodes

camera.name	PedestrianCount	totalPeopleCountWithVelocity
Camera5	20	34
Camera7	11	69
Camera8	19	38
Camera15	29	56
Camera16	24	51
Camera25	29	45
Camera26	24	53

1–7 of 7

Node Density

CameraID	Density	ProjectedDensity
Camera5	0.2	0.34
Camera7	0.02	0.124
Camera8	0.046	0.093
Camera15	0.104	0.201
Camera16	0.04	0.086
Camera25	0.171	0.265
Camera26	0.053	0.118

1–7 of 7

NeoDash Visualization Features

- Nodes currently exceeding critical density threshold
- Nodes projected to exceed threshold in near future (accounting for adjacent nodes)
  - Critical Thresholds can be set by user

Currently Exceeding Critical Density

CameraID	Density
Camera5	0.2
Camera25	0.171

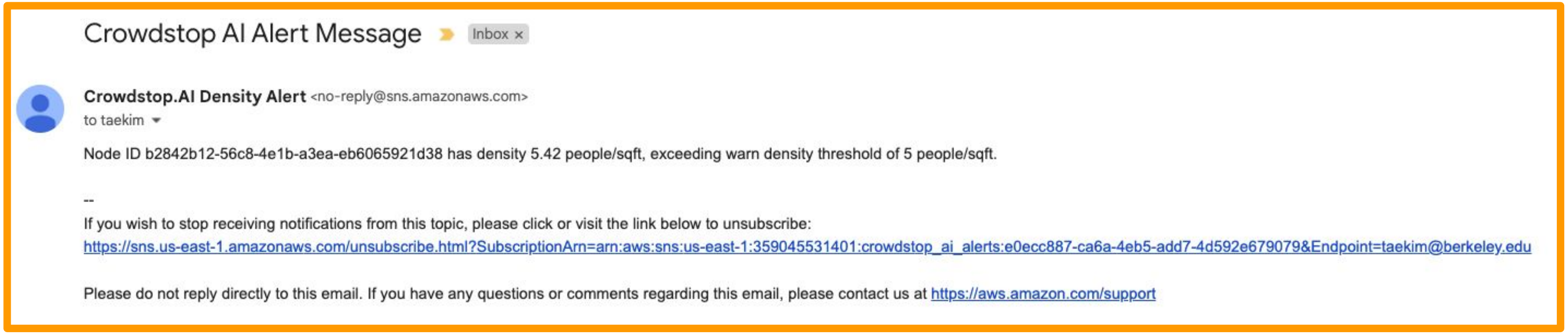
1–2 of 2

Projected To Exceed Critical Density

CameraID	projectedDensity
Camera5	0.34
Camera15	0.201
Camera25	0.265

Rows per page: 3 1–3 of 3

Example alert message via AWS SNS



Crowdstop AI Alert Message Inbox ×

Crowdstop.AI Density Alert <no-reply@sns.amazonaws.com>
to taekim

Node ID b2842b12-56c8-4e1b-a3ea-eb6065921d38 has density 5.42 people/sqft, exceeding warn density threshold of 5 people/sqft.

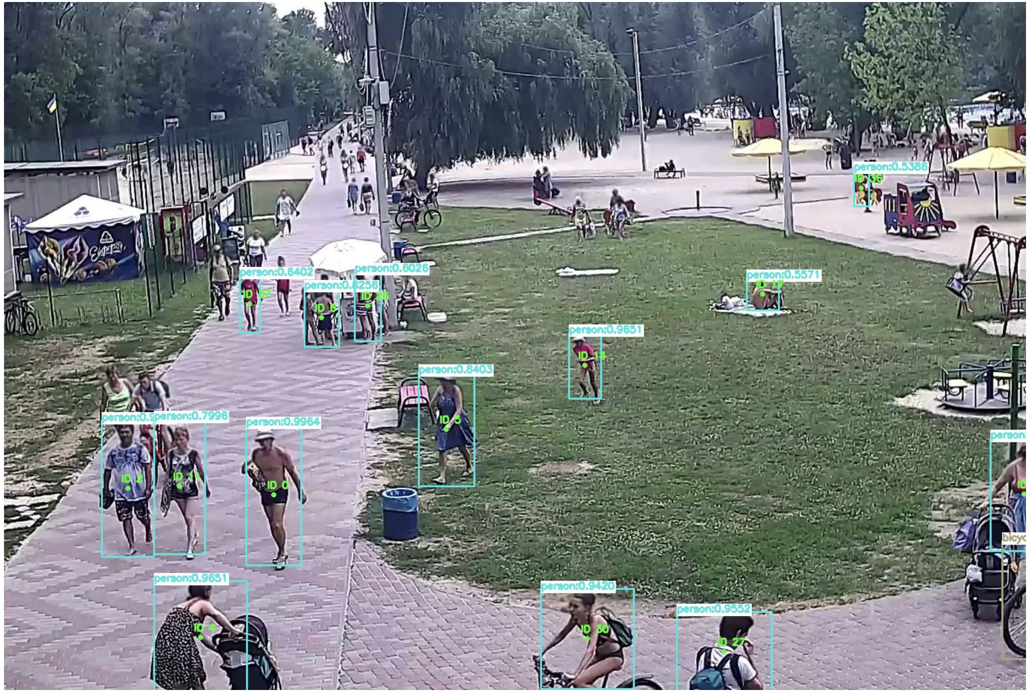
--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:359045531401:crowdstop_ai_alerts:e0ecc887-ca6a-4eb5-add7-4d592e679079&Endpoint=taekim@berkeley.edu

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

Thank you!

Appendix

Density Calculation + Anomaly Detection



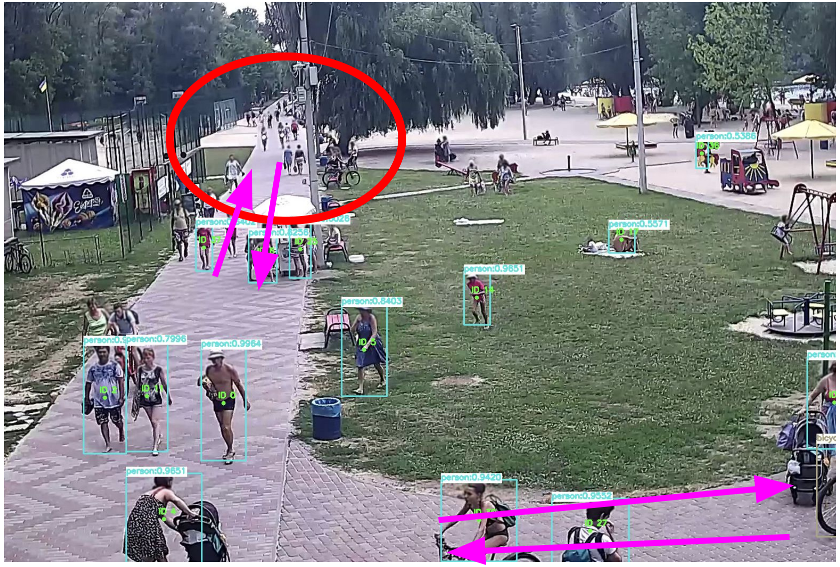
Critical crowd density:
7 people per square meter

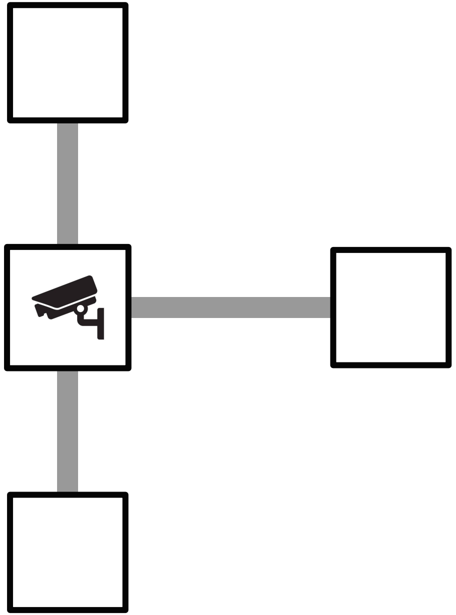
 # People Detected / Area within Frame

For each camera node:
- Area within Frame manually calculated (remove buildings, etc.)

Anomaly Detection:
- Does the Density approach critical density threshold?

What counts as “movement”?





Camera config files

Json file specific to each camera providing important metadata

- Name
- Longitude + latitude (determines uniqueness, used to generate UUID)
- Walkable surface area visible in frame in sqft
- Places the camera link to
    - Place ID
    - Zones in frame that link to place