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**
- 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

• 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/m²)
## Advantages over Status Quo

<table>
<thead>
<tr>
<th></th>
<th>Current</th>
<th>Crowdstop.AI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source</strong></td>
<td>● Concerned bystanders&lt;br&gt;● Security personnel</td>
<td>Security camera network</td>
</tr>
<tr>
<td><strong>Information</strong></td>
<td>Eye estimates</td>
<td>● Exact number of people&lt;br&gt;● Direction and magnitude of movement</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td>Limited by number of personnel</td>
<td>Potentially infinite given enough security cameras</td>
</tr>
<tr>
<td><strong>Monitored area</strong></td>
<td>Only at observed areas</td>
<td>Able to infer densities at unobserved areas</td>
</tr>
</tbody>
</table>
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)

Object tracking (e.g. centroid tracking)

- Bounding box + classification + object ID
Tracker Comparison

Centroid Tracker

Frame t-1

Object-1

Object-2

Centroids

Frame t

Object-1

Object-2

Object-3

Euclidean distance

IOU (Intersection over Union) Tracker

P1: IOU: 0.9

P2: IOU: 0.7

P3: IOU: 0.95

P4: IOU: 0.8

P5: IOU: 0.85

P6: IOU: 0.45

P7:
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

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

<table>
<thead>
<tr>
<th>Image</th>
<th>Detector</th>
<th>Tracker</th>
<th>ID Switches</th>
<th>MOTA</th>
<th>MOTP</th>
<th>IDF1</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>YOLO</td>
<td>IOUTracker</td>
<td>26</td>
<td>0.200</td>
<td>0.274</td>
<td>0.323</td>
<td>0.270</td>
<td>0.818</td>
</tr>
<tr>
<td>Quadrant Splitting</td>
<td>YOLO</td>
<td>IOUTracker</td>
<td>21</td>
<td>0.251</td>
<td>0.270</td>
<td>0.483</td>
<td>0.413</td>
<td>0.728</td>
</tr>
</tbody>
</table>
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

Model Output
{"Zone 1": -1, "Zone 2": -1, "Zone 3": +2}
Zone-Linking Relevant Scenes

Scene 5

Scene 7

Scene 8
Zone-Linking Relevant Scenes

Scene 5

Scene 7

Scene 8
Multiple Object Tracking Pipeline Summary

- **Video Feed** Resizing & Quadrant Splitting
- **Object Tracking** IOU Tracker
- **Upload to GraphDB**
- **Camera Node**
- **Object Detection** QuadYOLO
- **Pedestrian Calculations** Density and Movement
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

- Update graph DB
- Infer neighboring nodes
- Publish alerts to SNS

Web UI via NeoDash
API spec

Camera ID: 12345

PUT /camera/12345

Positive velocity indicates movement towards the camera
### Optimizing performance: Downsampling

#### Model Metrics

<table>
<thead>
<tr>
<th>Frame Count Cadence</th>
<th>Recall</th>
<th>IDsw</th>
<th>Ground Truth</th>
<th>IDsw/GT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.432</td>
<td>88</td>
<td>10839</td>
<td>0.81%</td>
</tr>
<tr>
<td>3</td>
<td>0.427</td>
<td>70</td>
<td>3627</td>
<td>1.93%</td>
</tr>
<tr>
<td>5</td>
<td>0.411</td>
<td>87</td>
<td>2167</td>
<td>4.01%</td>
</tr>
<tr>
<td>10</td>
<td>0.319</td>
<td>40</td>
<td>1085</td>
<td>3.7%</td>
</tr>
</tbody>
</table>
Front-end visualization & UX
NeoDash Metrics

**Density**
\[ \text{Density} = \frac{\text{Number of People}}{\text{Area of Interest}} \]

Area of Interest: 10 m²
Number of people: 12
Density: 1.2 people / m²

**Velocity/Movement**
\[ \text{Velocity/Movement} = \text{Dictionary of movement across zones} \]

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

<table>
<thead>
<tr>
<th>Camera</th>
<th>Pedestrian Count</th>
<th>totalPeopleCountWithVelocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera5</td>
<td>20</td>
<td>34</td>
</tr>
<tr>
<td>Camera7</td>
<td>11</td>
<td>69</td>
</tr>
<tr>
<td>Camera8</td>
<td>19</td>
<td>38</td>
</tr>
<tr>
<td>Camera15</td>
<td>29</td>
<td>56</td>
</tr>
<tr>
<td>Camera16</td>
<td>24</td>
<td>51</td>
</tr>
<tr>
<td>Camera25</td>
<td>29</td>
<td>45</td>
</tr>
<tr>
<td>Camera26</td>
<td>24</td>
<td>53</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CameraD</th>
<th>Density</th>
<th>ProjectedDensity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera5</td>
<td>0.2</td>
<td>0.34</td>
</tr>
<tr>
<td>Camera7</td>
<td>0.02</td>
<td>0.124</td>
</tr>
<tr>
<td>Camera8</td>
<td>0.046</td>
<td>0.093</td>
</tr>
<tr>
<td>Camera15</td>
<td>0.104</td>
<td>0.201</td>
</tr>
<tr>
<td>Camera16</td>
<td>0.04</td>
<td>0.086</td>
</tr>
<tr>
<td>Camera25</td>
<td>0.171</td>
<td>0.265</td>
</tr>
<tr>
<td>Camera26</td>
<td>0.053</td>
<td>0.118</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>CameraID</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera5</td>
<td>0.2</td>
</tr>
<tr>
<td>Camera25</td>
<td>0.171</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CameraID</th>
<th>projectedDensity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera5</td>
<td>0.34</td>
</tr>
<tr>
<td>Camera15</td>
<td>0.201</td>
</tr>
<tr>
<td>Camera25</td>
<td>0.265</td>
</tr>
</tbody>
</table>
Example alert message via AWS SNS

Crowdstop AI Alert Message

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

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:a0ecc887-ca6a-4eb5-add7-4d592e679078&Endpoint=teekim@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”? 

[Image of a busy street scene with people and a circular highlighted area]
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