Matching Inertial Logs to Video Tracking for Indoor Positioning

Eric Lau, **G**eronimo Mirano, **H**arihar Subramanyam

I. Introduction

Motivation

- Indoor positioning systems require expensive hardware beacons or access points.
- Combining building security cameras with user smartphone data may provide cost-effective indoor positioning that leverages commodity equipment.

Objective

- Track moving objects using a fixed, nearly overhead, video camera.
- Collect logs from phone's inertial sensors to estimate a user's motion.
- Position user by matching inertial logs to a tracked object.

II. Experiment

Equipment

- Logitech C930e webcam
- iPhone (x 3)
- MacBook Pro

Procedure

- Collected data in 5 locations: Stata, Lobby 7, Walker, Bldg. 66, Bldg. 34.
- Fixed camera above, looking down on experimenters.
- Record timestamp at the start of data collection.
- Two experimenters (four for Bldg. 34), walk around below, each carrying an iPhone running an inertial logger.
- Camera records footage of experimenters.

Collected Data

- Inertial logs from accelerometer, gyroscope
- Overhead video of experimenters
- Timestamp at the start of the experiment

III. System

Object Tracker (Harihar Subramanyam)

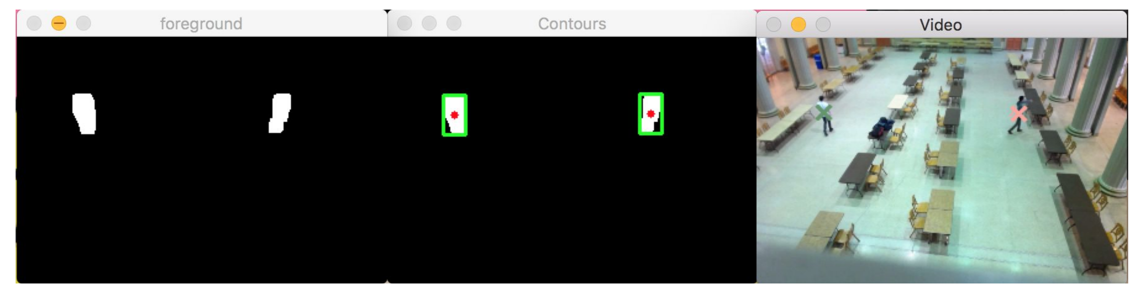

Figure 1. Split-screen of object tracker showing extracted foreground, contours, and trackers plotted on footage.

- **Input Frame** – a color image →
- **Shrink Frame** – to speed up computation →
- **Perspective Transform** – to get overhead view →
- **Foreground Extraction (MoG)** – to locate moving objects ↓
- **Median Blur + Dilate** – to denoise + enlarge objects →
- **Bounding Boxes + Mass Centers** – to denote obj. size/location →
- **Merge Nearby Contours** – to repair broken objects →
- **Filter Out Small Contours** – to denoise →
- Assign **Kalman Filters** to Mass Centers (**Hungarian Algorithm**) – to decide what measurement (i.e. mass center) to feed each filter →
- **Unpair Faraway Filters/Centers** – to remove poor assignments →
- **Give Nearby Filters to Same Center** – to handle overlapping obj. ↓
- **Remove Dead Filters** – to handle offscreen objects →
- **Suppress Spurious New Filters** – to denoise →
- **Measurement + Prediction Update** – to update Kalman Filters →
- **Output Object Locations** – `Array[(filterId, pixelX, pixelY)]`

Inertial Logs (Geronimo Mirano)

- Trajectory is reconstructed from inertial data via finite-difference integration.
- Friction and smoothing are added for stability.
- Data are projected onto a 2D plane in the estimator's best-guess direction of gravity.
- Explored frequency analysis of inertia and found it sufficient to estimate footstep rate, but didn't have a use for it in our final system.

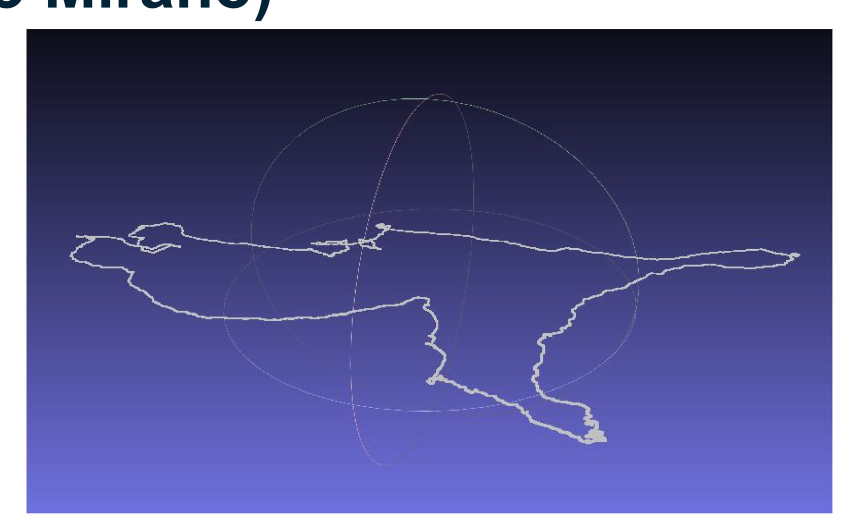

Figure 2. Reconstructed trajectory.

Matching Algorithm (Eric Lau)

Object Tracking: `<`t_s, `trackerID`, x_i, x_j, v_i, v_j`>`, ... Inertial Data: `<`t_s, v_i, v_j `>`, ...

Matching

1. **Simulate online system request each second.** Aggregate all object tracking and inertial samples with timestamp t_s in each 1 second window.
2. **Compute Spearman rank correlation score** between windowed **velocity** data for object tracking and inertial sensing, for each `trackerID`.
3. **Sequence** x_i, x_j **locations** to coarse 20x20 grid.
4. **Perform belief update of internal Hidden Markov Model (HMM)** to compute updated probability distribution over states.
5. **Assign user** to `trackerID` with x_i, x_j located in maximum-probability state.

HMM Parameters

- **Hidden states:** Grid cells, numbered *0..399*
- **Observations:** Windowed data
- **State transition probabilities:** Prioritize local transitions, discourage otherwise.
- **Emission probabilities:** Weighted by max correlation score of `trackerID` in state.

Belief Update

Given prev. probability distribution over states,

- **Multiply by transition probability matrix:** output probability of reaching each state *s'*.
- **Take element-wise product of destination** (above) and **emission** probabilities: output updated probability distribution over states.

Output estimate of user's bounding box ID: `trackerID`

IV. Evaluation

Ground Truth Labels

- Consists of `(timestamp, pixelX, pixelY)` tuples for each experimenter.
- Found by playing video and dragging mouse to follow experimenters' paths.

Error Metric

- Euclidean distance between estimated pixel location and ground truth location

Results

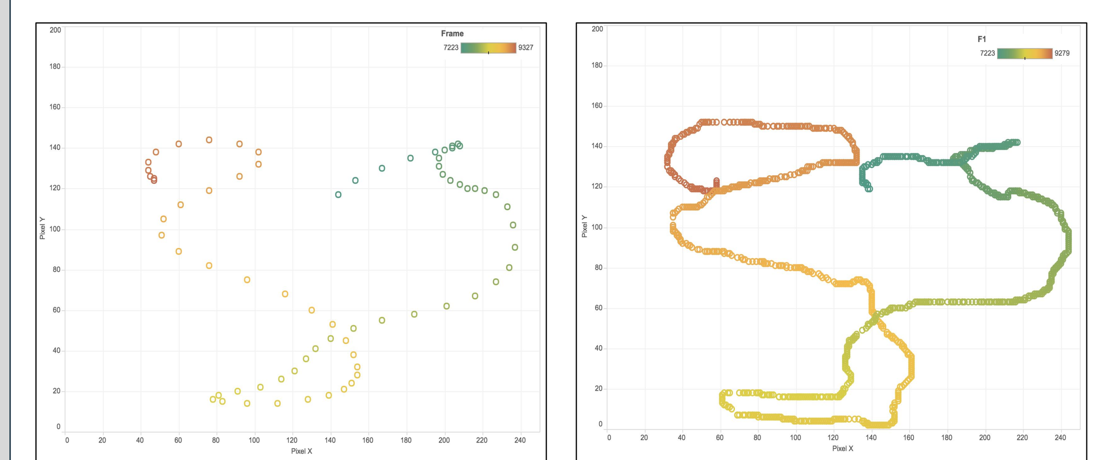

Figure 3. Trajectory estimate from matching (left) compared to ground truth (right).

	Mean Error	Median Error	Maximum Error
Experimenter 1 (of 2), Walker	11.6 pixels	10.8 pixels	31.6 pixels
Experimenter 2 (of 2), Walker	33.3 pixels	13.1 pixels	159.9 pixels
Experimenter 1 (of 4), Bldg. 34	67.8 pixels	43.1 pixels	216.6 pixels
Experimenter 2 (of 4), Bldg. 34	54.6 pixels	44.7 pixels	193.8 pixels

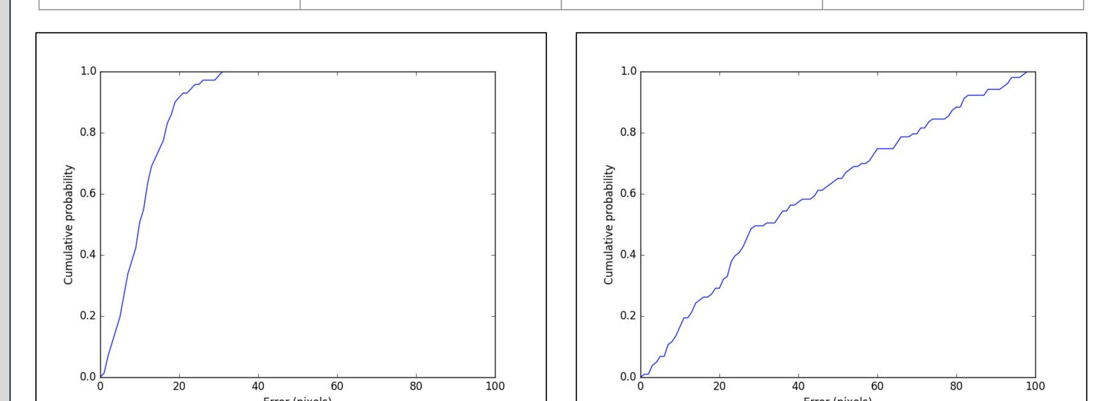

Figure 4. CDF error plots of Walker (left) and Bldg. 34 (right).

V. Conclusion and Future Work

- Used background-subtraction based object tracking and collected inertial data.
- Matched inertial logs to video tracking using correlation + HMM matching algorithm to achieve <50 pixel median accuracy in 300x300 pixel video footage.
- Next steps: Real-time positioning, w/ ongoing work in client/server architecture; map pixel `(x, y)` position to real-world coordinates.

Code

- **Tracker + Ground Truth Collector + Evaluator:** https://github.com/hariharsubramanyam/ObjectTracker
- **Inertial:** https://bitbucket.org/geronm/motionprocessing/
- **Matching Algorithm + Client / Server Architecture:** https://bitbucket.org/ericlaubitbucket/6s062finalproject