Today's Agenda:

1. Introduction
2. Live demo
3. Outdoor pathfinding
4. Indoor pathfinding
5. Conclusion

INTRODUCTION

01

Introduction

Challenges

Students struggle with campus and room navigation.

Limitations

Current apps lack room-level directions and accuracy.

Solutions

Campus-specific website with room-to-room navigation.

Key Features

Uses live data: weather, crime, construction, traffic.

Benefits

Tailored, accurate routes and time estimates.

Impact

Solves a key student pain point effectively.

LIVE DEMO

02

Find the Best Path

Start Position

Select Start Position

End Position

Select End Position

Transportation Mode

- (•) Walking
- ( ) Biking

Safe Mode

- (•) On
- ( ) Off

Find Path

OUTDOOR PATHFINDING

03

Outdoor Pathfinding

Data collection

We mapped out the buildings of UC Davis using OpenStreetMap

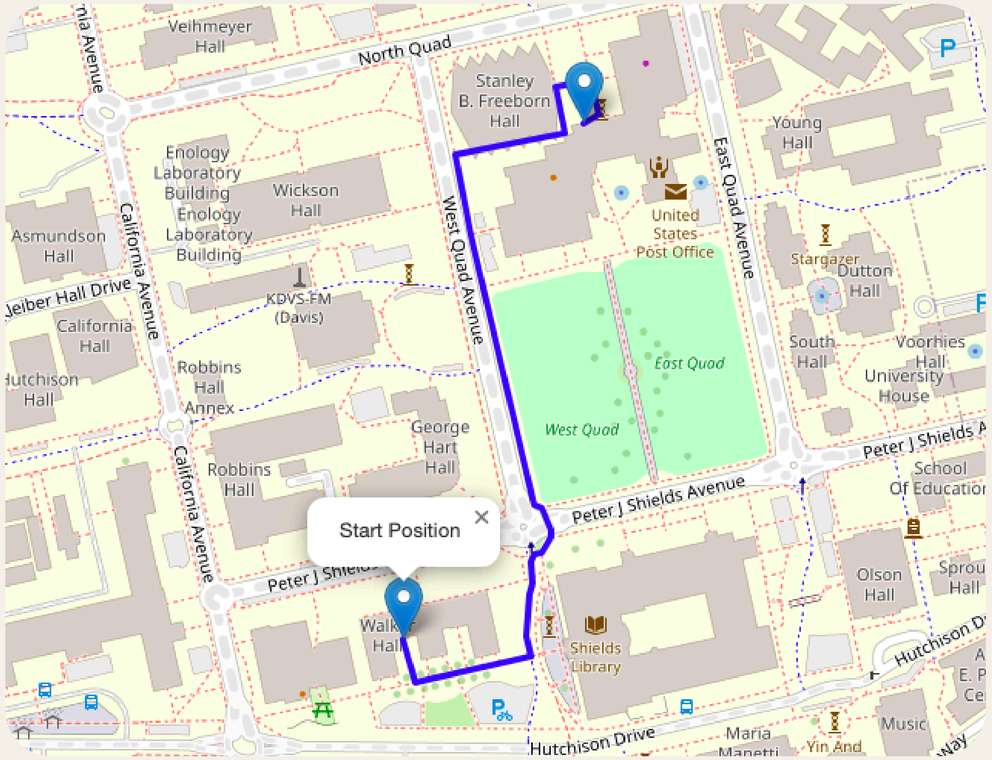
A* algorithm

We used an A* algorithm with a Euclidian heuristic where the weights are the time to travel between nodes

Weight calculation filters

We used the following filters for the weight calculation:

- Transportation mode
- Weather
- Live traffic data
- Crime proximity

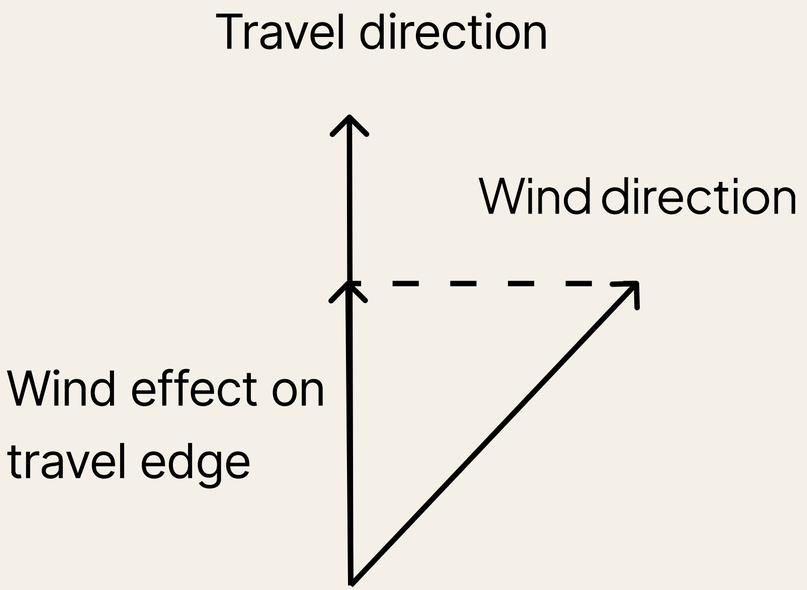


A* selects the lowest cost edge:

cost = edge time + distance to goal

Weather Weight Calculation

- **Rain:** increase time for all edges
- **Wind:** project the wind direction



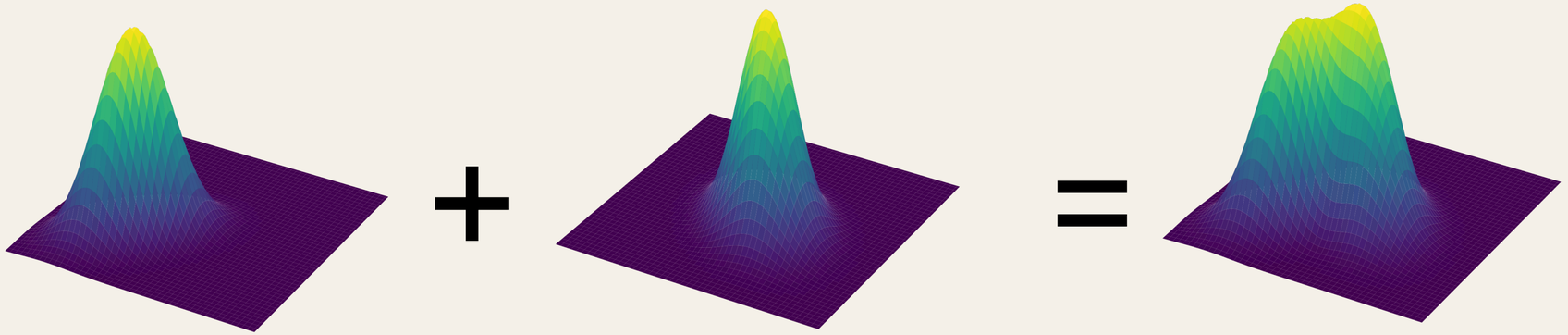
To calculate the wind effect we take the **dot product**!

wind effect = travel direction • wind direction

Transportation Weight

- **Bike:** Multiplies the edge weights by a small factor
- **Walking:** Multiplies the edge weights by a larger factor

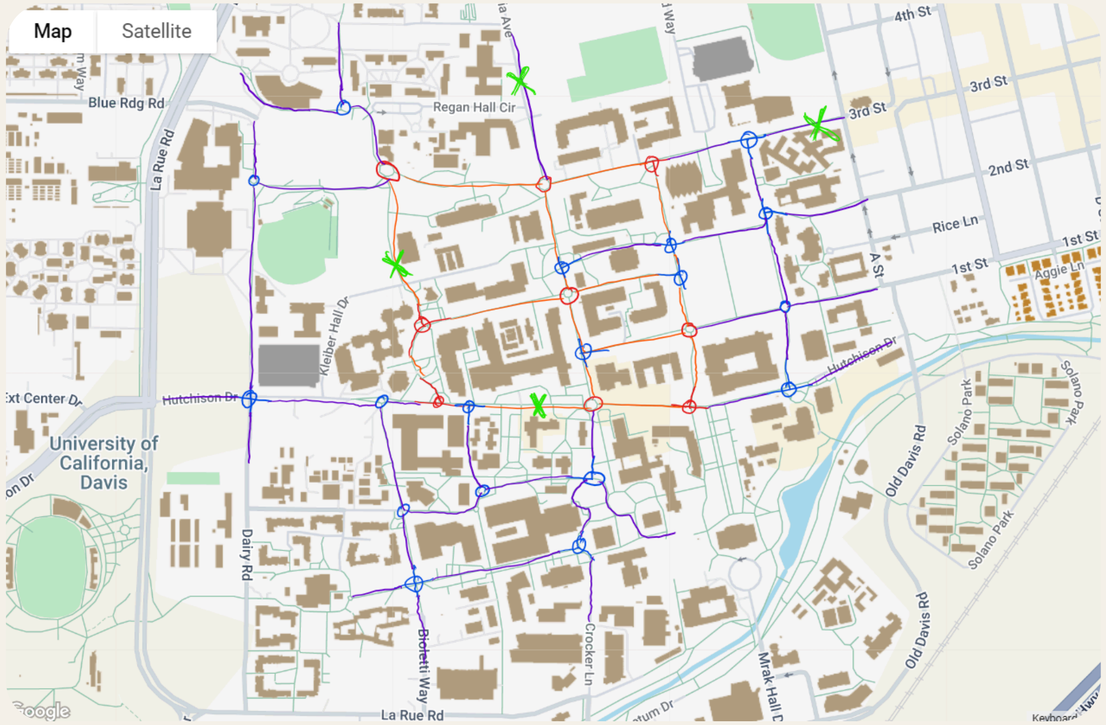
Crime Weight Calculation



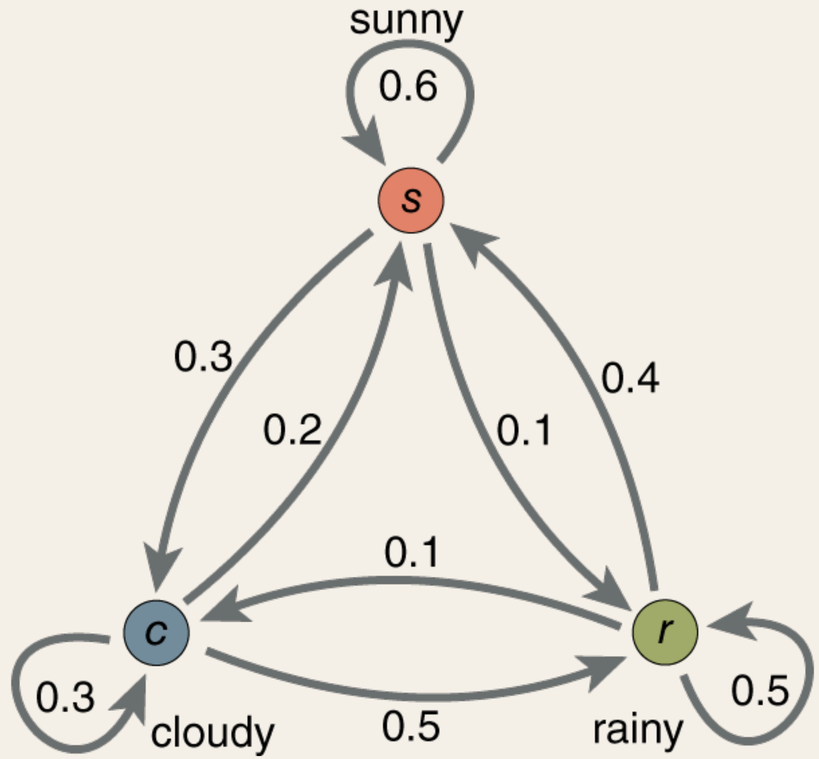
1. Create a heatmap of crime edge multipliers by summing previous crime data

2. Combine with distances to the nearest telephone post

How to calculate live traffic flows over UC Davis?



Manually select edges by drawing polygons



Convert to Markov chain

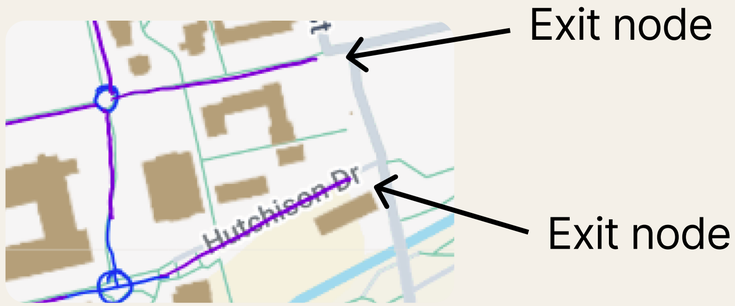
Markov Chain Creation

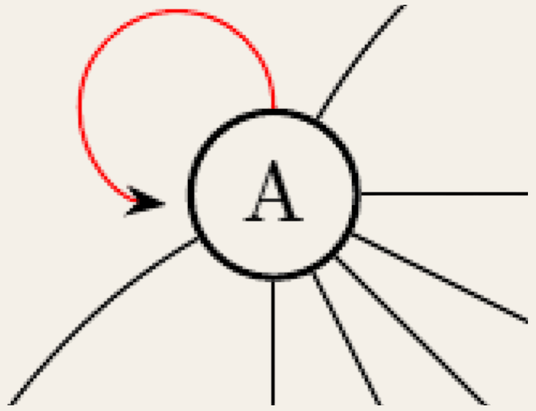
Goal: Create a Markov transition matrix

$$\begin{bmatrix} 0.1 & 0.2 & 0.4 & 0.3 \\ 0.3 & 0.1 & 0.5 & 0.1 \\ 0.1 & 0.2 & 0.1 & 0.6 \\ 0.3 & 0.3 & 0.3 & 0.1 \end{bmatrix}$$

- Exit nodes where 50% goes in and 50% goes out to sink node
- Avoid stability issues by using teleportation
- Calculate matrix based on:

1. Proximity to sink
2. Connection to one of 9 priority nodes



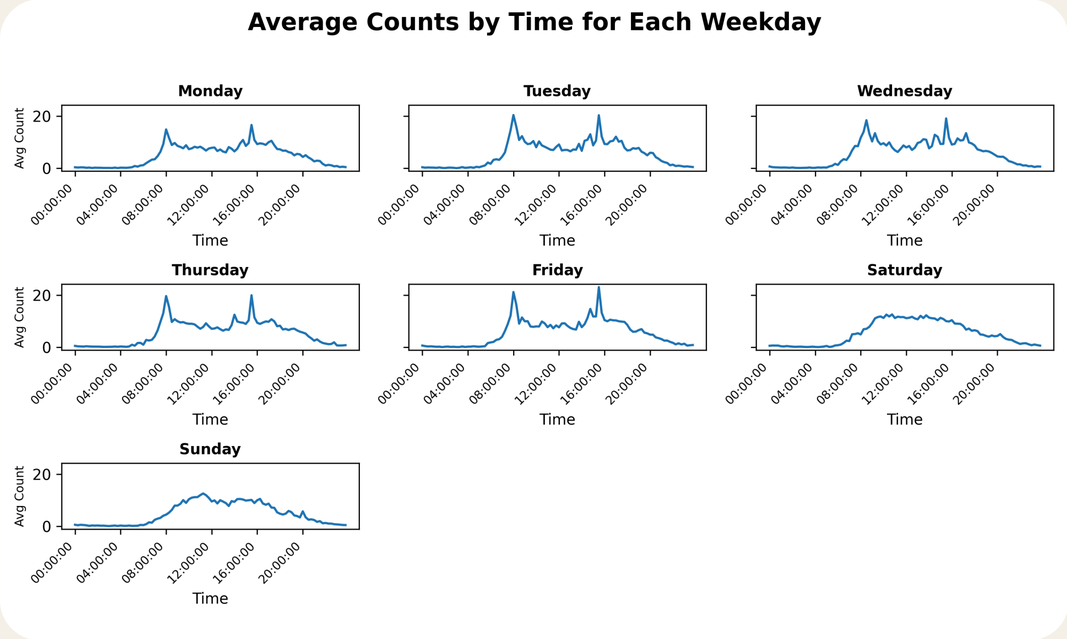


Stationary Distribution and Live Traffic Data

- From Markov chain calculate stationary distribution
- Scale the stationary distribution with the live data

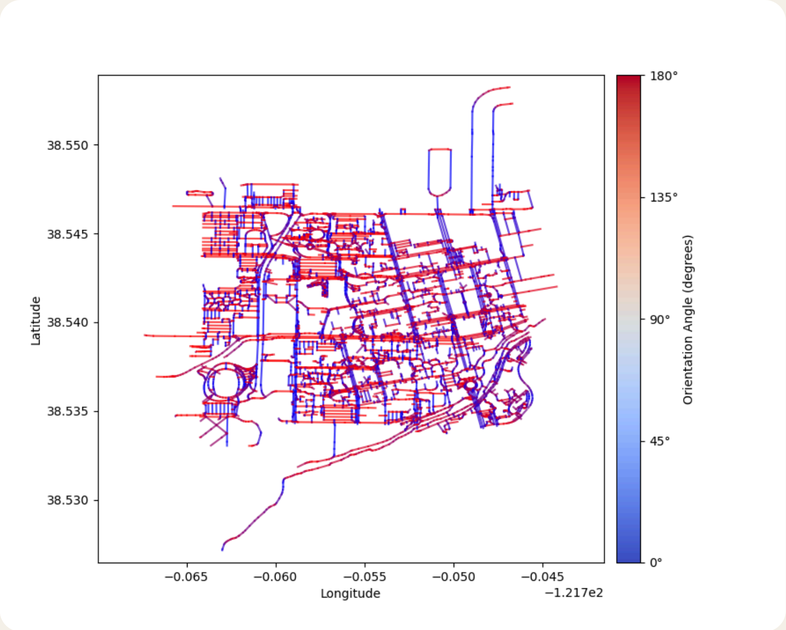
$$Ax = \lambda x$$

Key insight: for the distribution vector x to be stable it must adhere to the above (Whaaaat, it's an eigenvector problem!)

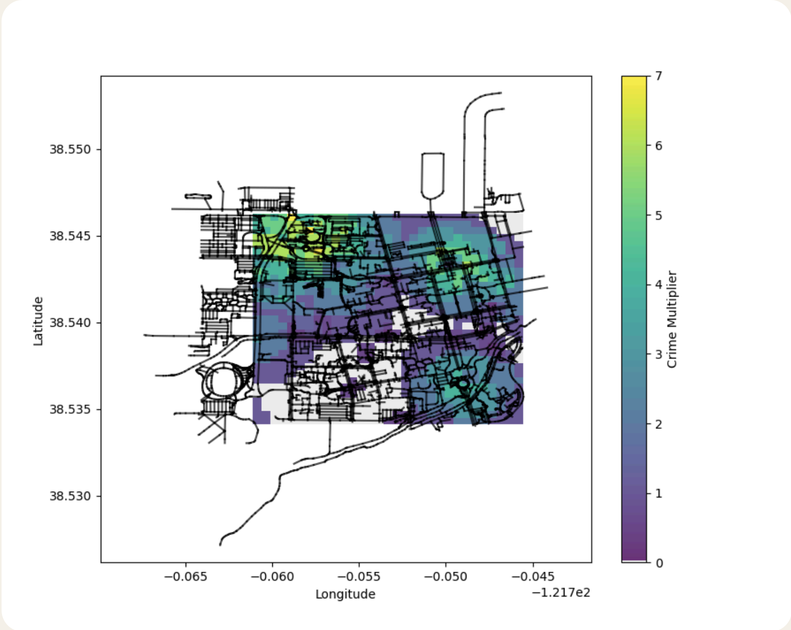


Then just simple scaling to match live traffic data

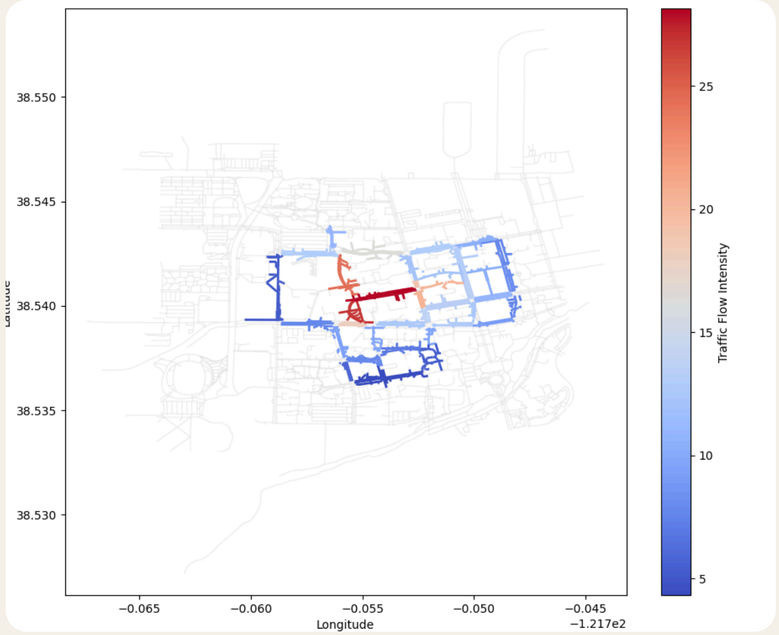
Results of outdoor pathfinding



A. EDGE DIRECTIONS



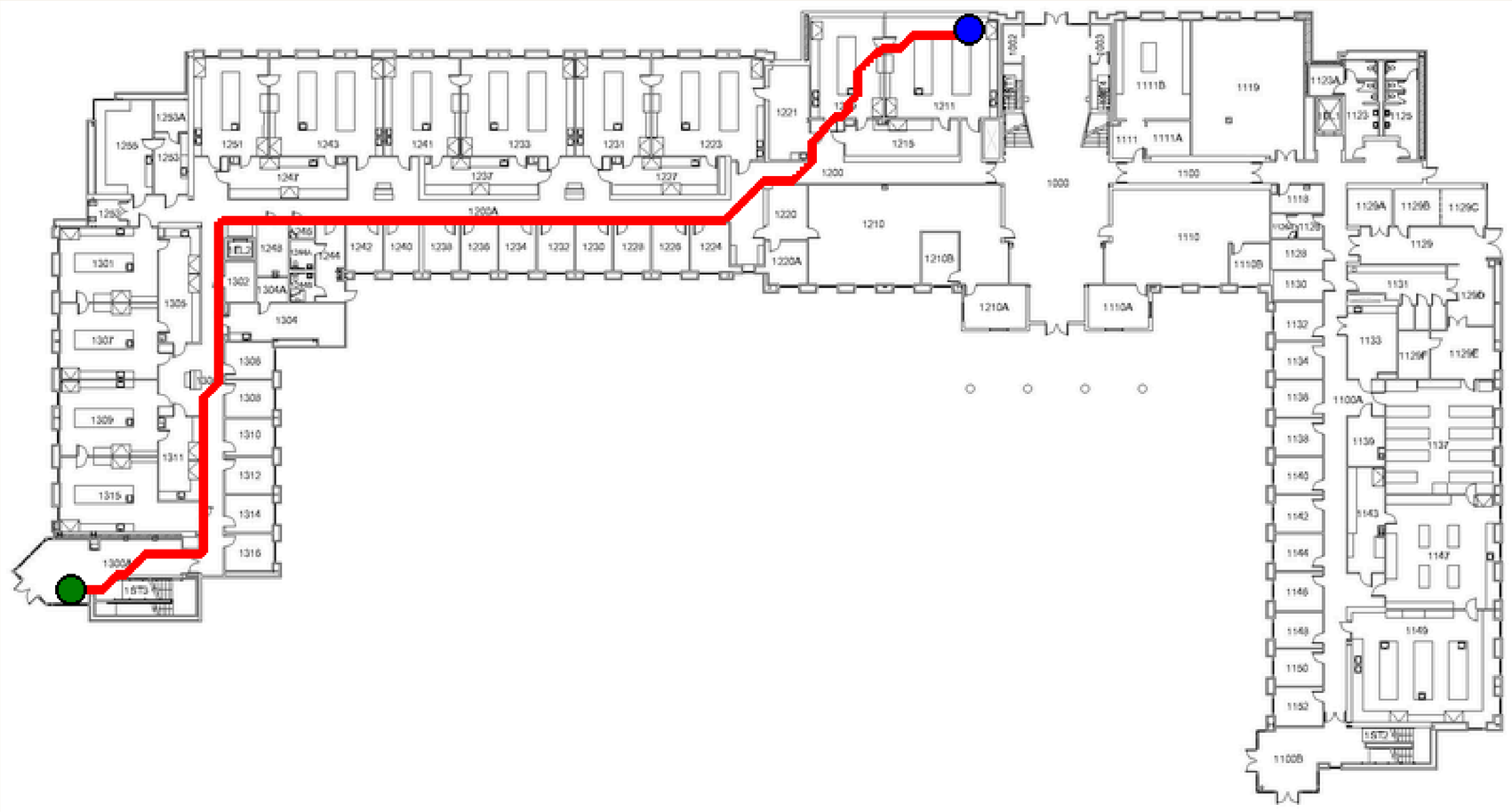
B. CRIME HEATMAP



C. STATIONARY DISTRIBUTION

INDOOR PATHFINDING

04



Run A* on the black and white pixels, but how are we going through the black doors?

Indoor Pathfinding Pipeline

- **Image cropping**

  Crop the images for preprocessing

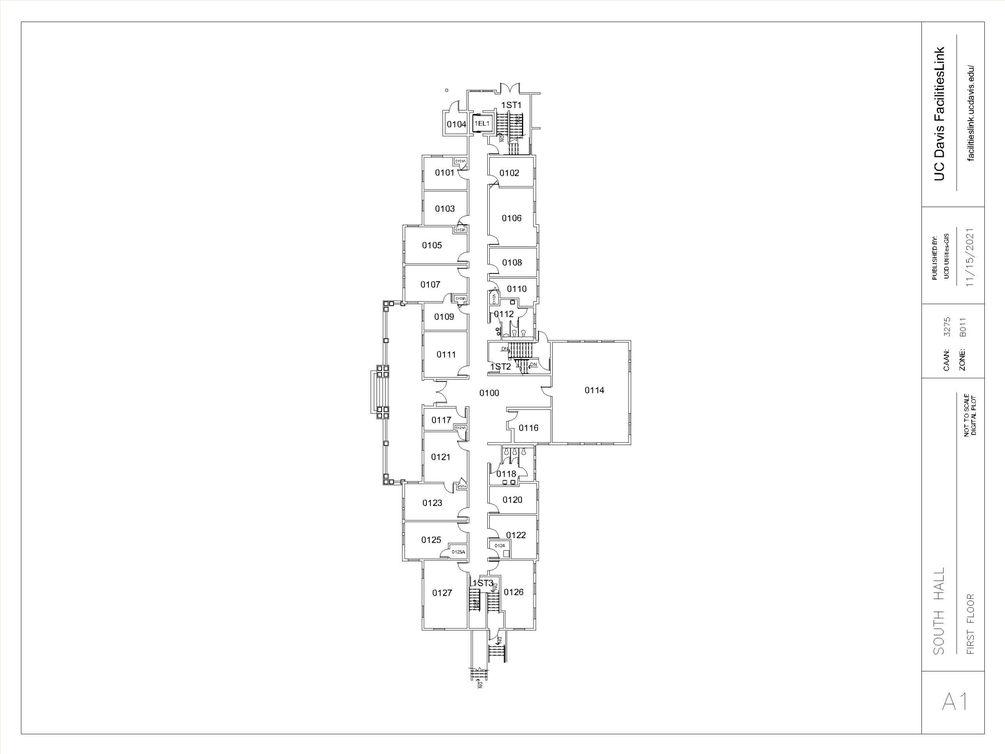
- **Door removal**

  Remove the doors to be able to pathfind through them

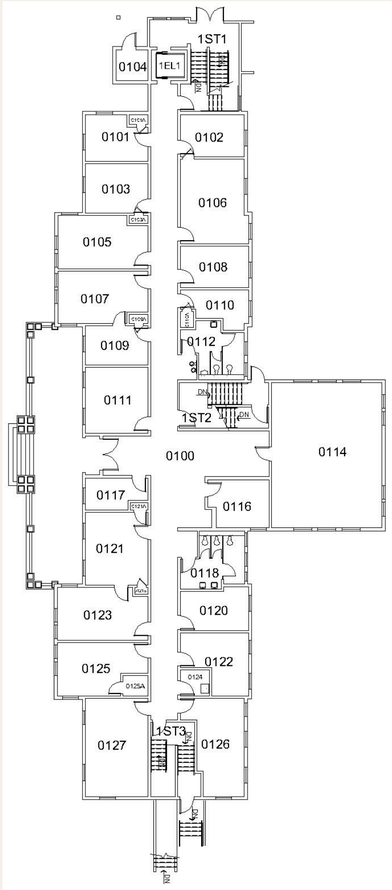
- **Entrance detection**

  Detect entrances and link to outdoor pathfinding

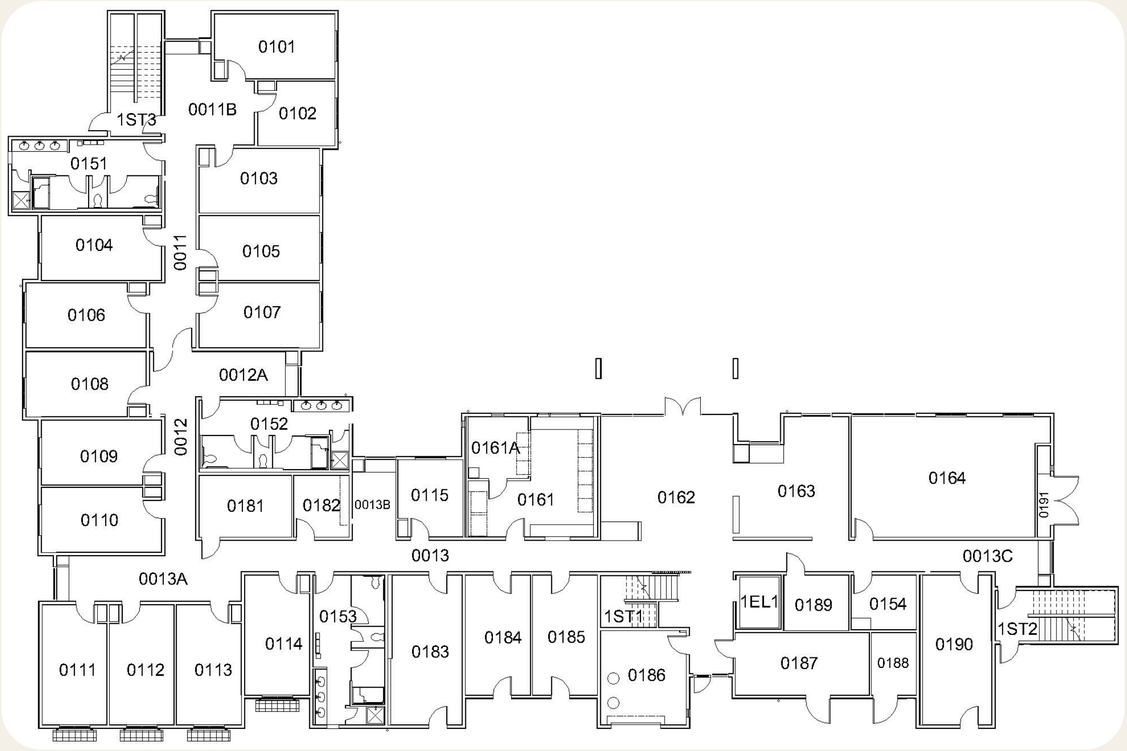
IMAGE CROPPING



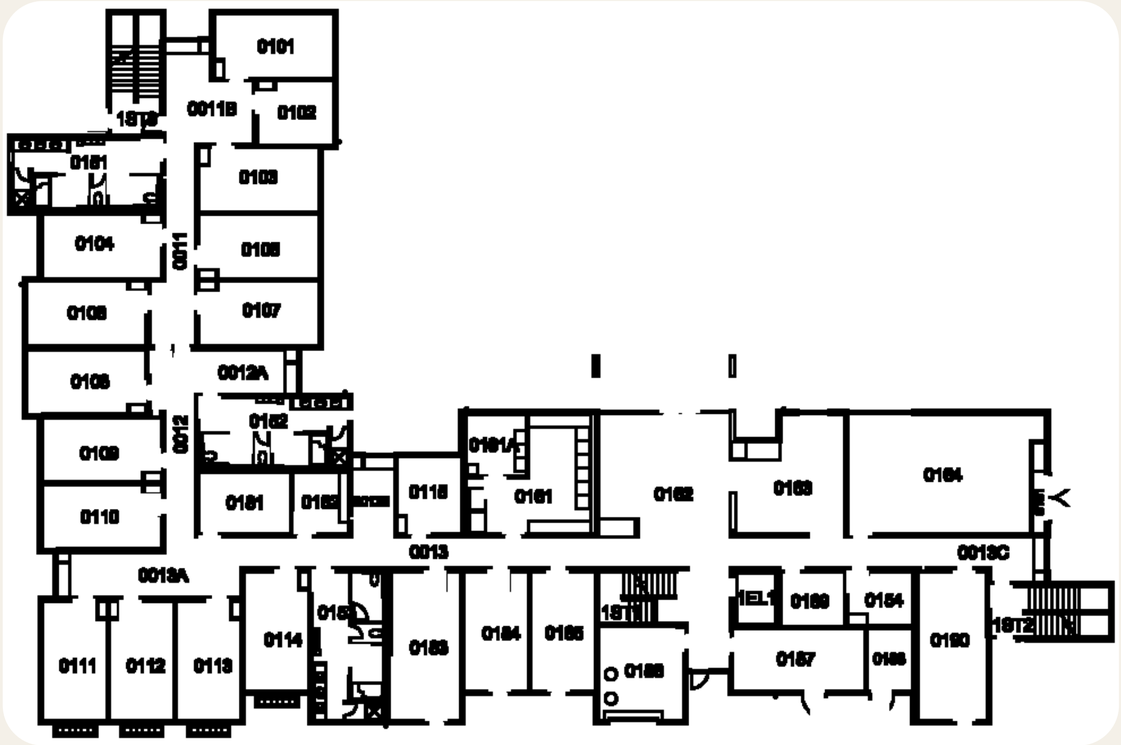
CROP THE IMAGES
USING SCANLINE



DOOR REMOVAL

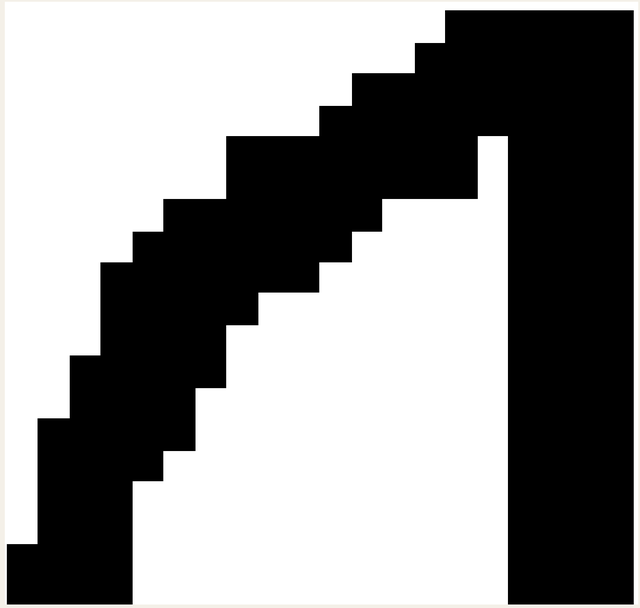


BEFORE DOOR REMOVAL



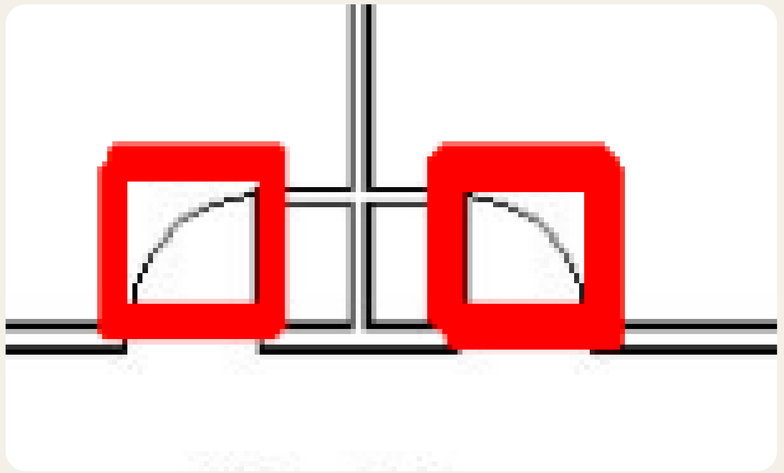
AFTER DOOR REMOVAL

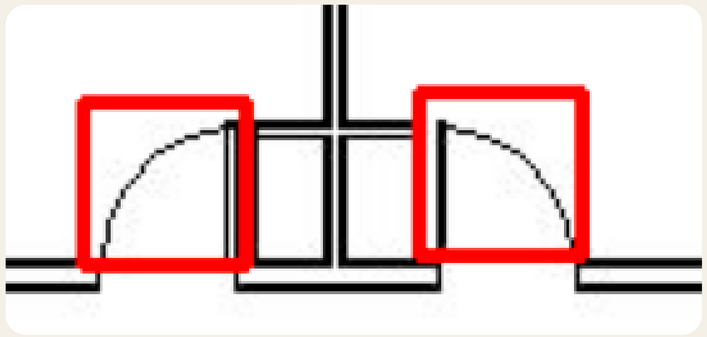
Detecting Doors Using a Template



- Use cv2.matchTemplate to find similar structures
- Test with 15 different rotations of template
- Test with template at scales 0.9, 1.0, 1.1
- Only select door if confidence is over a threshold

Getting Rid of Overlapping Detections





Remove overlapping detections because:

- They were removing walls
- Saves time by not needing to loop through each during door removal

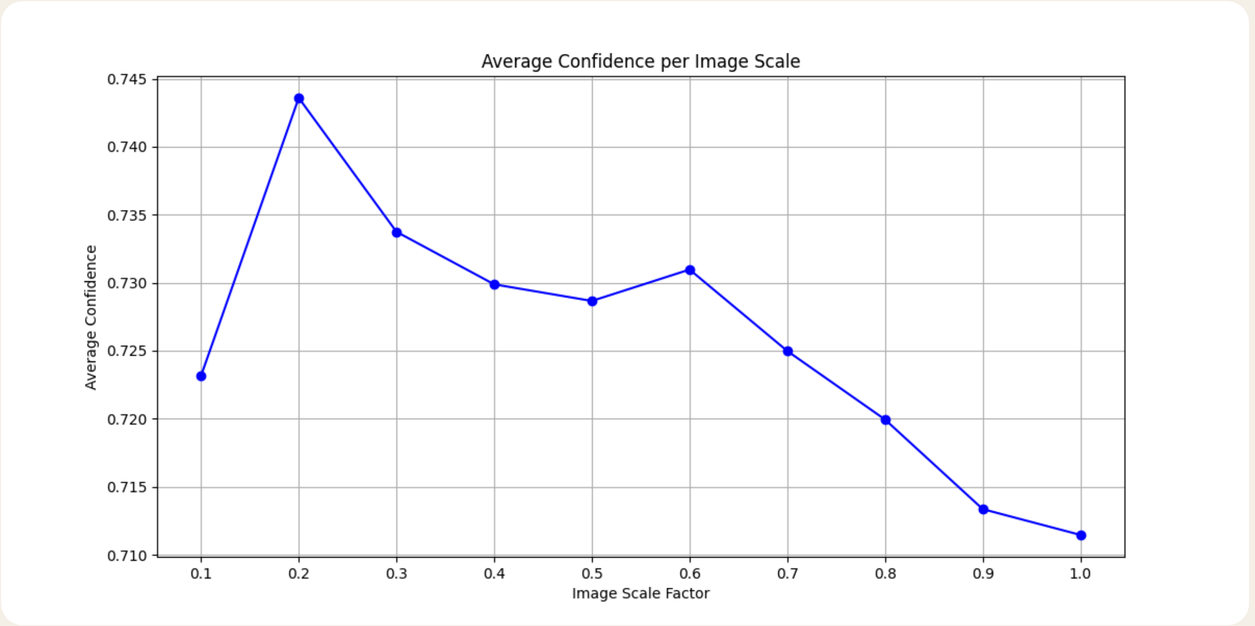
Method: Non-Maximum Suppression

1. Group the the detections with a KD-tree
2. Choose the one with the highest confidence

PROBLEM!!!

The template requires specific door size (10-15 pixels)

Optimal Scale Detection



Select the scale with the highest average confidence!

The door template is based on the smallest door size

↓

All floor plans should be scaled down

Non-convex building issues

↓

grid search

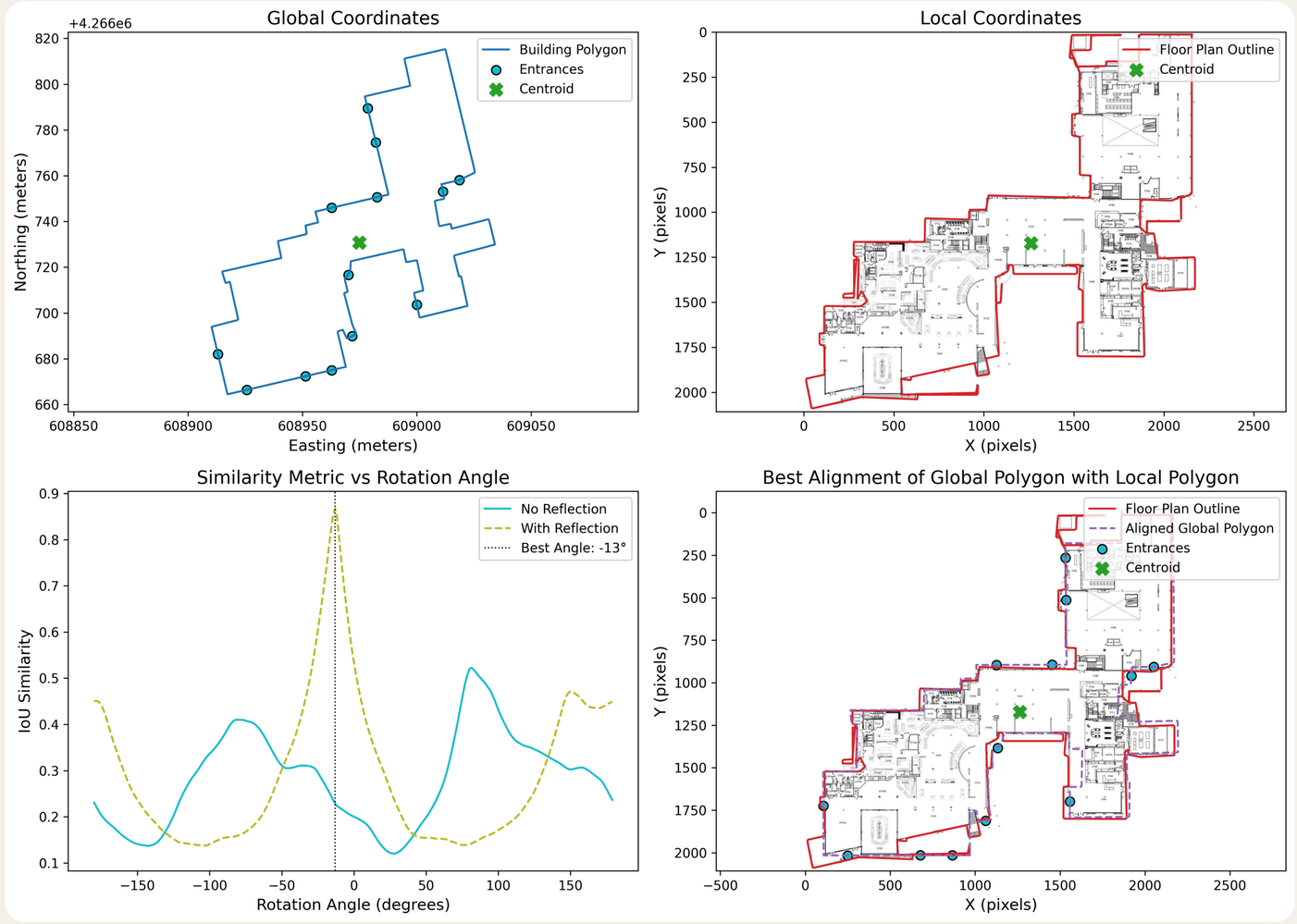
The results are pretty good

Entrance Mapping

We want to project the
outdoor entrances
onto the indoor floor plan to
get corresponding entrance

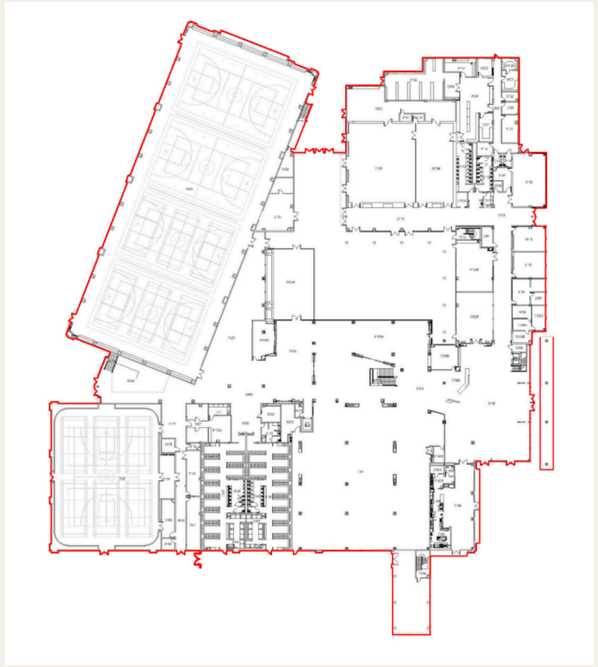
Steps:

1. Create polygons
2. Align centroids
3. Scale to local
4. Rotate to fit

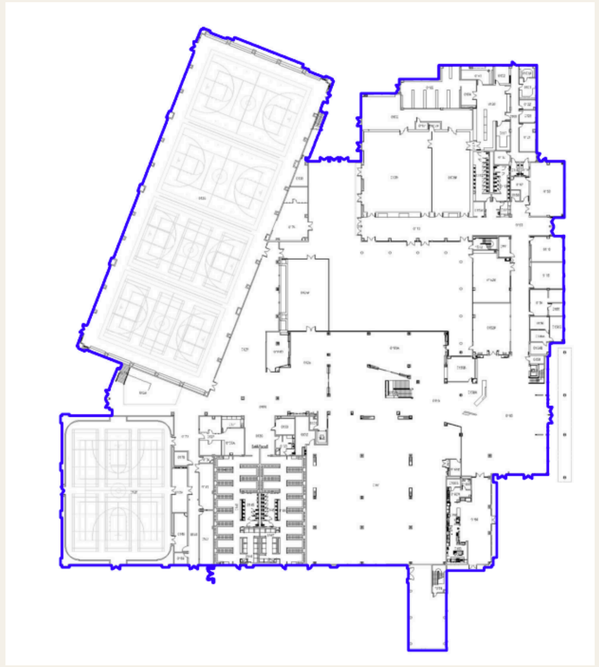


Step 1/4: Create Polygons

To be able to align indoor/outdoor we need the perimeter of the building!



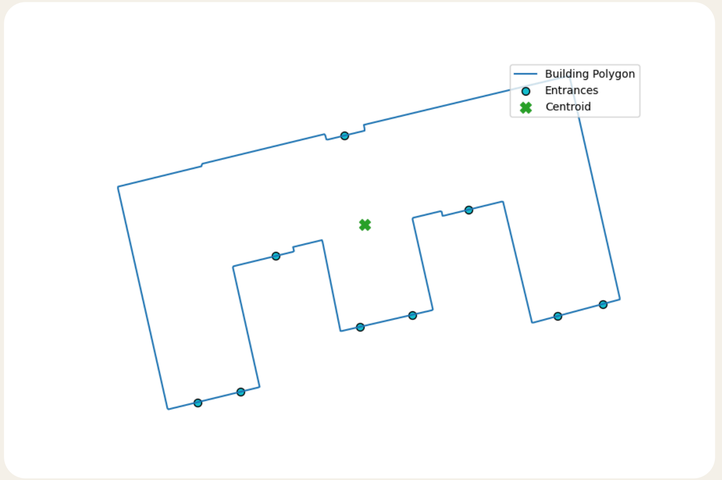
Using cv2.findCountours we get all of the cyclical loops in the image after closing gaps with cv2.morphologyEx



Select the largest loop as the perimeter

(outdoor polygons are trivial)

Step 2/4: Align Centers



To align the buildings we need the same centroid.

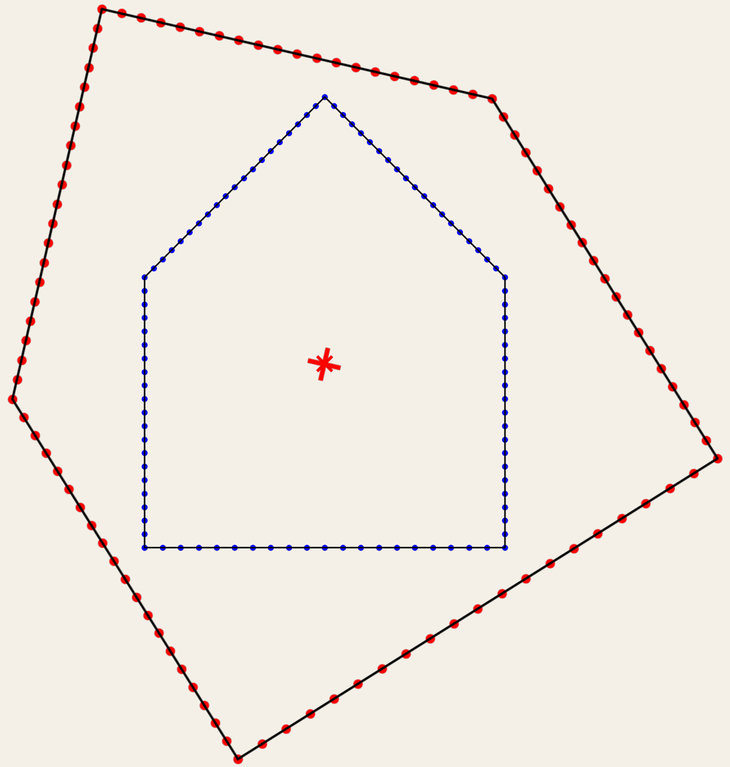
How to calculate?

1. Interpolate the perimeter
2. Calculate the center of mass (average position)

Step 3/4: Making the Polygons the Same Scale

Different scaling methods:

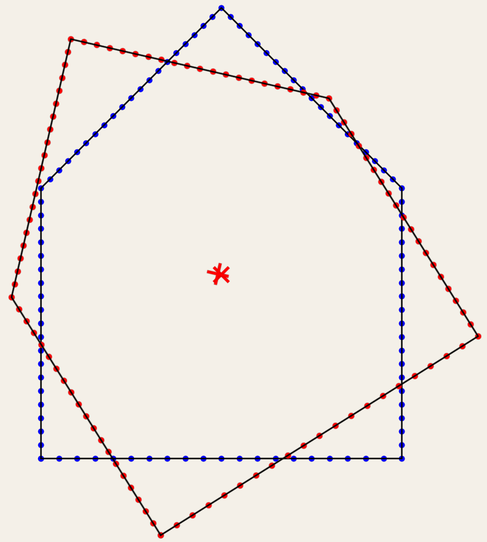
- Perimeter based
- Area based



Area based was better due to irregular floorplan polygons

Step 4/4: Making the Polygons the Same Rotation

Problem: What similarity metric to use?

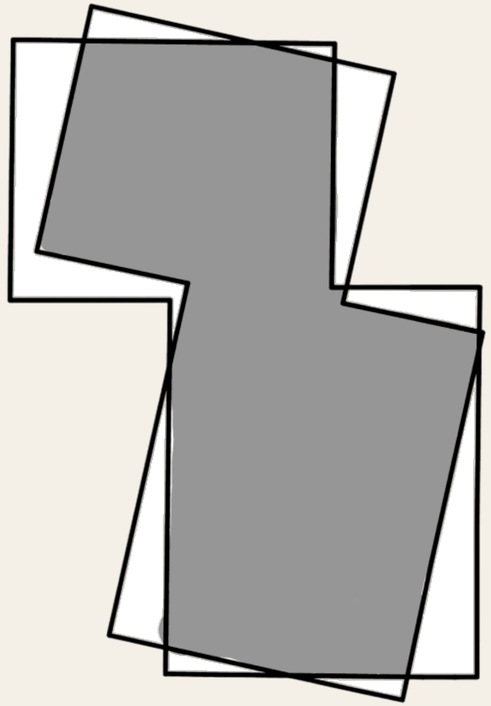


Alternatives:

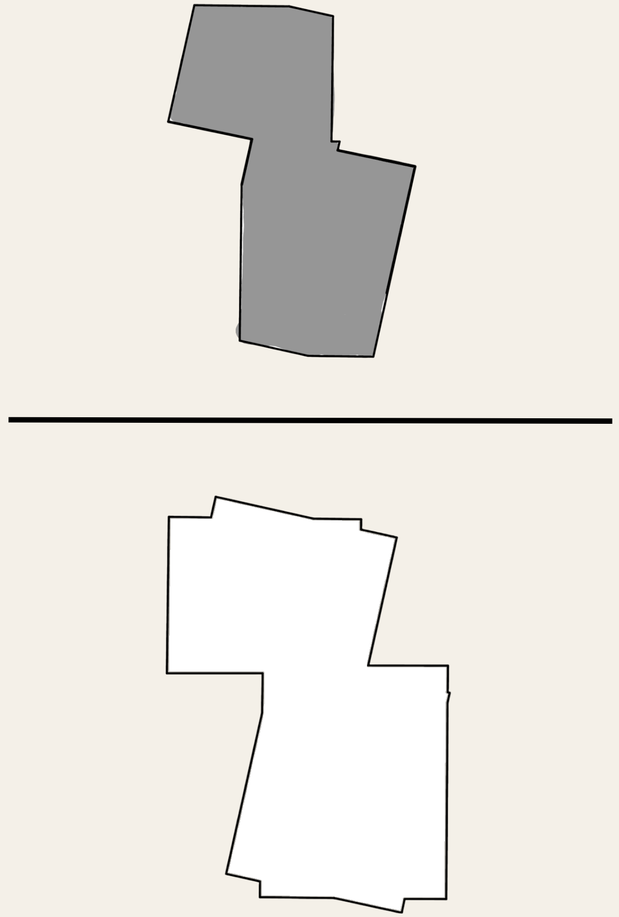
- Procrustes analysis (minimized sum of squares) - expensive
- Hausdorff distance (maximum of distance to nearest point) - inaccurate
- Best option: Intersection over Union (IoU)

Intersection over Union

Initial rotation



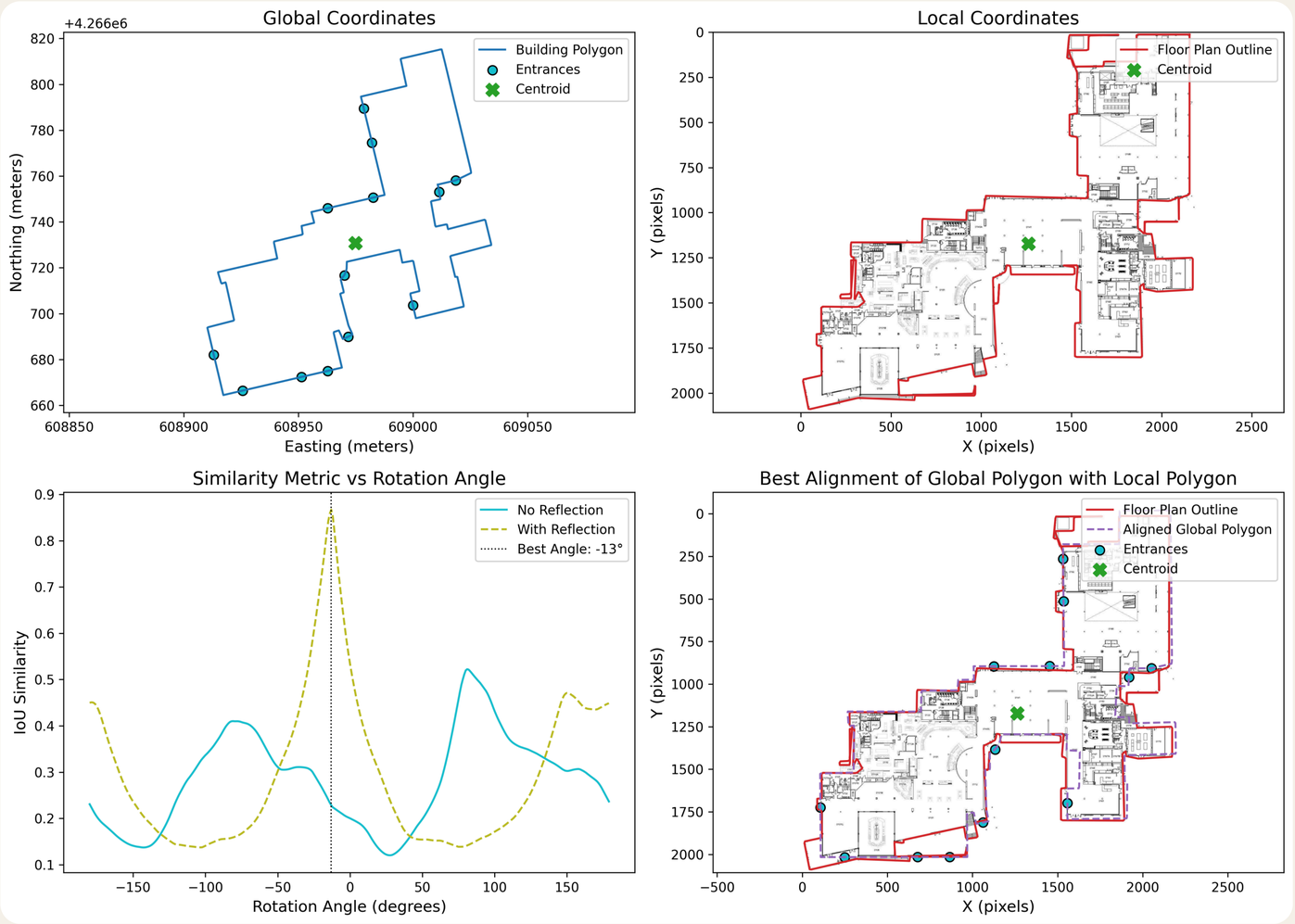
Similarity metric



Recap on the entrance mapping

We project the global entrances onto the local.

When pathfinding between buildings we find the nearest entrance and enter/exit



CONCLUDING THOUGHTS

05

CONCLUSION

Critical Need

Navify addresses the longstanding gap in seamless indoor and outdoor navigation for UC Davis.

Innovative Approach

Utilizes computer vision, graph theory, and real-time data to deliver accurate and efficient navigation.

Real-World Impact

Improves campus accessibility and safety by integrating features like room-to-room navigation and contextual adjustments.

Scalability

Provides a scalable solution applicable to other large campuses and complex environments.

Final Note

Navify demonstrates the power of interdisciplinary innovation to solve real-world challenges.