Image Recognition Error Log 1

Single Start and End Program

The reason for using image recognition is to simulate an interactive desktop. Real scene image recognition can be added.

First Attempt
Centroid Offset: My calculation of the centroid is based on the entire mask area. This means any extra noise or slight inconsistencies in the mask can cause the centroid to deviate from the expected position.

Scale Bar Blurriness: The Hough Transform I use for detecting lines can be sensitive to noise or slight changes. The scale bar looks a bit jagged or pixelated, which can pose a challenge for detection.

To resolve these issues:

For the Centroid: I improved the mask application to reduce noise and then calculated the centroid.
For Scale Bar Detection: I considered adopting a different method (such as contour detection) to reliably identify the scale bar.

Second Attempt
I applied morphological operations to clean the mask, hoping for a more accurate centroid.
I planned to use contour detection to find the scale bar, calculate its length, and then use it for distance calculations.

Third Attempt
Scale Bar Not Detected
Reason: The contour-based method seems to struggle in detecting the scale bar in new images. Since the scale bar is a prominent horizontal black line, I can revert to using the Hough Transform for its detection but with refined parameters for better accuracy.

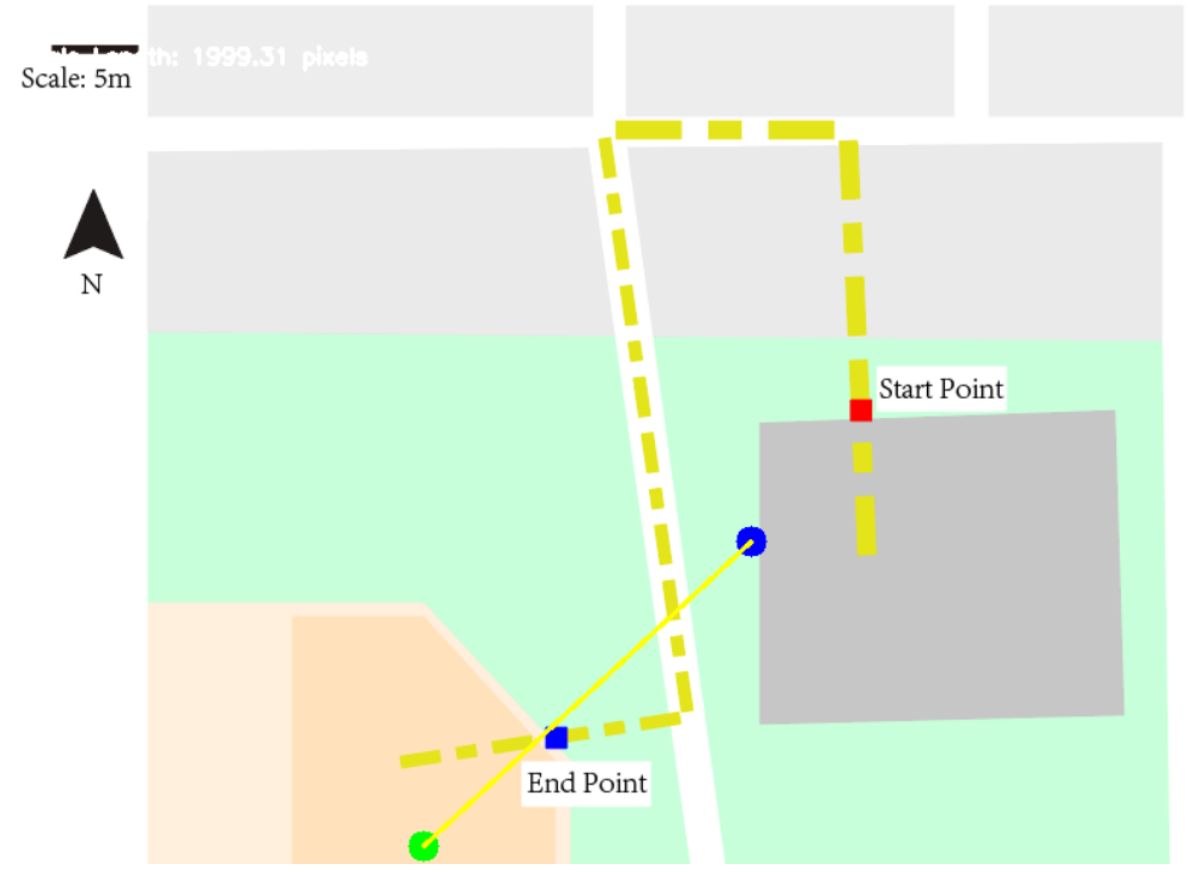
I modified the detector_scale_bar function to use the Hough Transform again for line detection but will adjust it specifically for the scale bar's characteristics.

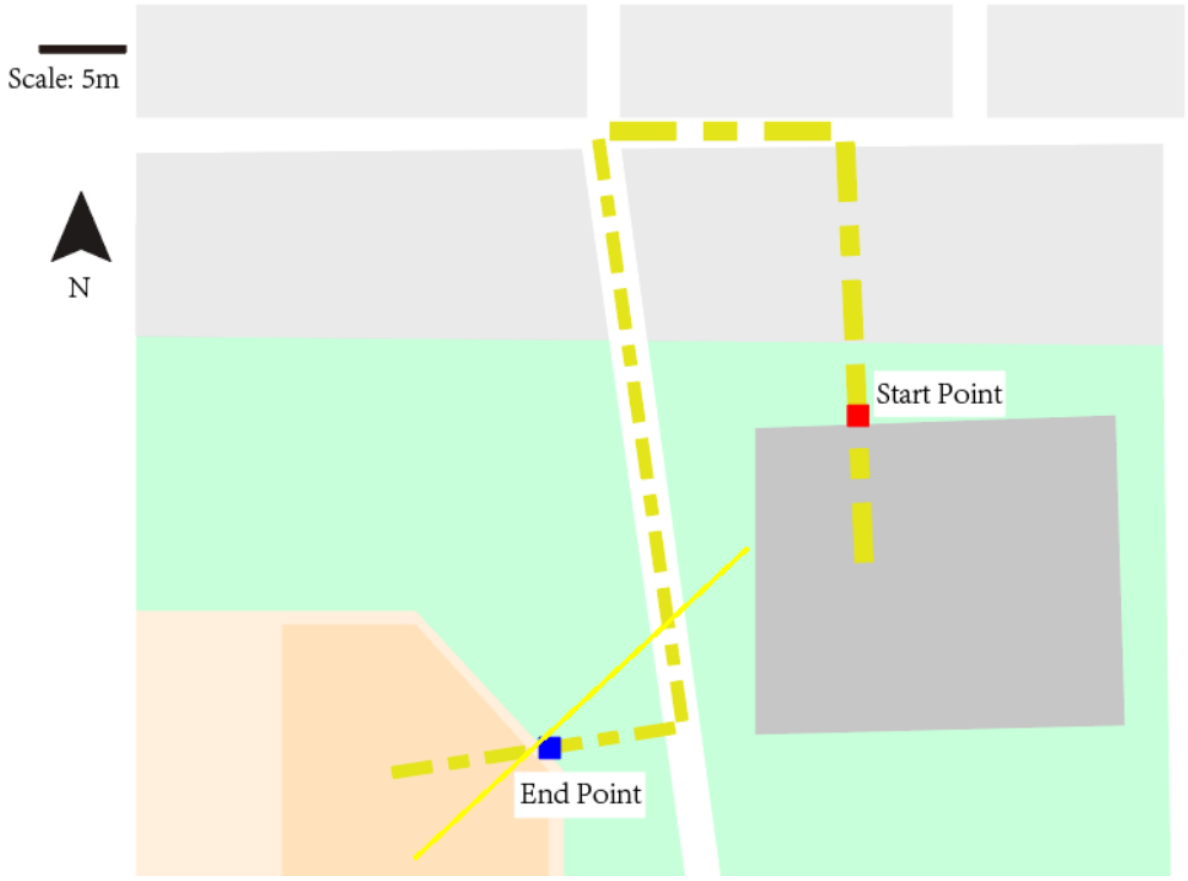
Fourth Attempt
Still no success. In Photoshop, I adjusted the size and removed the scale bar.

Fifth Attempt
Success

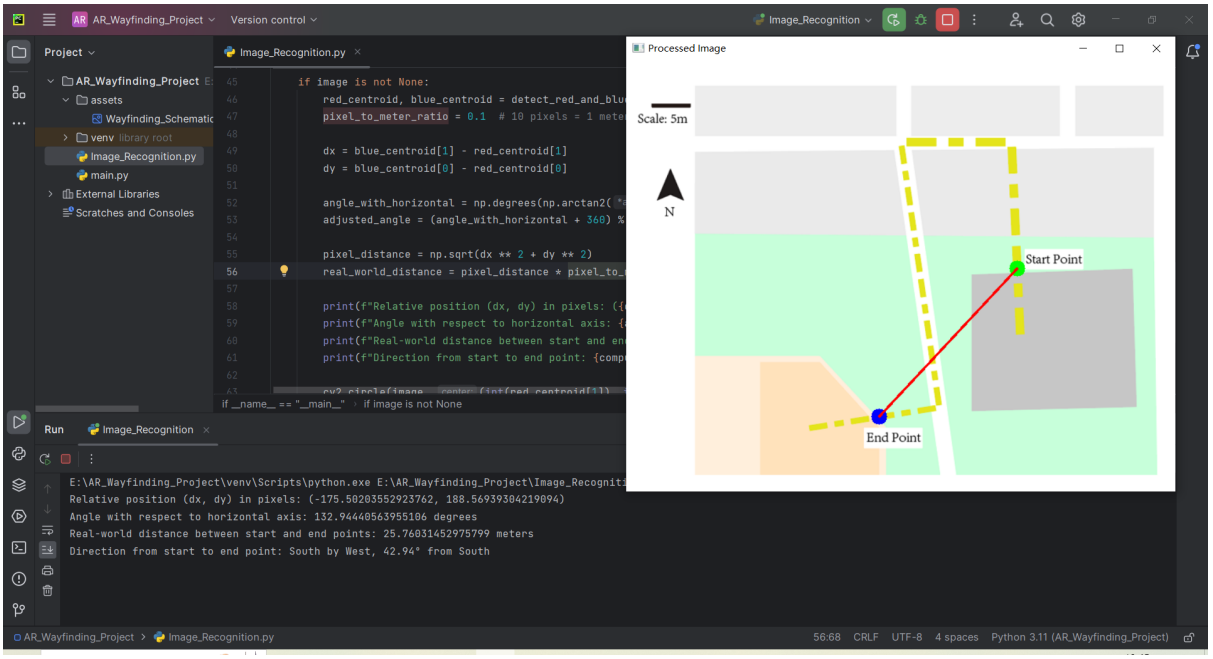
Define East as 0°.

Incorrect Illustration





Correct Illustration


AR_Wayfinding_Project
Version control
Image_Recognition
Project
Image_Recognition.py
AR_Wayfinding_Project
assets
Wayfinding_Schematic
venv library root
Image_Recognition.py
main.py
External Libraries
Scratches and Consoles
if image is not None:
red_centroid, blue_centroid = detect_red_and_blu
pixel_to_meter_ratio = 0.1 # 10 pixels = 1 mete
dx = blue_centroid[1] - red_centroid[1]
dy = blue_centroid[0] - red_centroid[0]
angle_with_horizontal = np.degrees(np.arctan2(
adjusted_angle = (angle_with_horizontal + 360) %
pixel_distance = np.sqrt(dx ** 2 + dy ** 2)
real_world_distance = pixel_distance * pixel_to_
print(f"Relative position (dx, dy) in pixels: ({
print(f"Angle with respect to horizontal axis: {
print(f"Real-world distance between start and en
print(f"Direction from start to end point: {comp
if __name__ == "__main__" › if image is not None
Run
Image_Recognition
E:\AR_Wayfinding_Project\venv\Scripts\python.exe E:\AR_Wayfinding_Project\Image_Recogniti
Relative position (dx, dy) in pixels: (-175.50203552923762, 188.56939304219094)
Angle with respect to horizontal axis: 132.94440563955106 degrees
Real-world distance between start and end points: 25.76031452975799 meters
Direction from start to end point: South by West, 42.94° from South
AR_Wayfinding_Project › Image_Recognition.py
56:68 CRLF UTF-8 4 spaces Python 3.11 (AR_Wayfinding_Project)
Processed Image
Scale: 5m
N
Start Point
End Point