AALBORG UNIVERSITY

STUDENT REPORT

XNav - From Pixels to Paths: Explainable Social Navigation with Cost Maps using Object Detection and VLMs

Endri Dibra, Joel Thamby, Daniel Jonatan Bank Dharampal, Nicklas Alexander Hougaard Deding

AAU - Department of Electronic Systems - Robotics Msc - ROB7-163 - November 2025

7th Semester mini Project

Electronic Systems
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
XNav - From Pixels to Paths

Theme:
Explainable Social Navigation with Cost Maps using Object Detection and VLMs

Project Period:
Fall Semester 2025

Project Group:
163

Participant(s):
Endri Dibra, Joel Thamby, Daniel Jonatan Bank Dharampal, Nicklas Alexander Hougaard Deding

Supervisor(s):
Professor Rikke Gade

Copies: 1

Page Numbers: 45

Date of Completion:
3rd of November 2025

Abstract:

This project concerns the implementation of an Explainable Socially Aware Navigation approach, using technologies like YOLOv11n Object detector and SmolVLM Visual language model, as well as a cost map to produce more robust paths, generated by a global path planner such as A* algorithm. The images used to test the effectiveness of the pipeline, have different scenes, sizes and background noises, as a result, its testing generalization becomes more intuitive overall. In addition, the results/paths on the image are compared with results from other industry-defined VLMs/LLMs. Ultimately, this pipeline leverages smolVLM's abilities to provide a social cost and an evaluation score, to explaine why the certain produced path respects social norms.

Contents

Preface

Aalborg University, 3rd of November, 2025

Endri Dibra
<edibra25@student.aau.dk>

Daniel Jonatan Bank Dharampal
<ddhara22@student.aau.dk>

Joel Thamby
<jthamb25@student.aau.dk>

Nicklas Alexander Hougaard Deding
<ndedin22@student.aau.dk>

Double Outline

Table 1: Double outline

Chapter	Section	Purpose
Introduction	1	To introduce and set up the initial problem, highlighting motivation and research focus for the problem analysis, leading to an initial problem statement.
Implementation	2	To describe the core components, methods and functionality of the proposed idea for research, elaborating it more with tests and results and image illustrations, based on backgrounds with no, light and heavy noise application, different resolution sizes in pixels to detect possible limitations of the object detector, VLM and path planning algorithms, additionally testing our pipeline on different background scenes and finally to compare our results, thus the generated paths on the image with other large and with strong potential VLMs, such as ChatGPT, in order to check the robustness and generalization of our approach.
Challenges and Limits	3	To refer to the most important challenges and issues that our team was encountered with, and how we managed to solve them.
Future Scope	4	To mention how we would like to continue this project, in what environments and with which robots.
Conclusion	5	Ultimately, to discuss the alternative approach that this project's pipeline follows to achieve an Explainable Socially Aware Navigation, respecting social norms and explaining its chosen decisions for trust and credibility, as well as its future contribution into the field.

Introduction

This project concerns the research and implementation of an approach inspired by the VLM-Social-Nav paper, for explainable socially aware navigation, incorporating technologies such as, Path planning algorithm [A*], Object detection [YOLOv11n] and Visual Languange Models [VLM - smolVLM], all in one stack, pipeline, and tested on static images with different background clearness, such as cases with no, light and heavy noise, as well as non-identical background scenes, for instance, scenes like Urban road, Laboratory, Nature and playground and distinctive image pixel sizes, for the purposes of testing the limits and strengths of this particular approach and way of handling navigation on 2D occupancy grid maps.

Furthermore, the results [Generated paths] from our final algorithm incorporation all the tech stack used for this project, are tested with the likes of larger and more powerful VLMs/LLMs, like ChatGPT5, Gemini-2.5-Flash, Grok-4-Fast and Claude-Sonnet-4.5, and the end results are really interesting and informative, showcasing their limits too.

Moreover, the concept of costmap is introduced, which consists of crucial cost safety parameters, such as object distance, object type [Humans have higher cost for safety reasons], wall distance, social cost and other inflations. The end purpose of it, is to offer a path solution that is at the same time distance optimal and socially aware, respecting people's space and other social norms.

Ultimately, the output provided by the tech pipeline will be evaluated and compared to determine its robustness and strong foundational bases.

Implementation

The implementation steps needed to build this project are as follows:

1. **Image creation** Images or an occupancy grid maps are created using design tools like CANVA and LLMs, containing people and obstacles like chairs, cars, bicycles and other, to simulate simple real-life scenes in order to use them for testing the navigation tech pipeline.
2. **Geometric Cost Map** So, to create a robust cost map, penalties such as image wall boundaries inflation and avoidance are applied, as well as path turn and curvature limits. This way, the first limited areas to pass are generated and set. Finally, its contribution to total cost map is 40 percent or 0.4/1.0
3. **Object Detection** After the image scenes creation object detection is applied to these images to detect people and obstacles. Then their detected zones have yellow bounding boxes and are inflated by a high cost [higher cost for people] radius, to avoid A*'s decision path planning to be around these areas, as a result an improvement in obstacle avoidance and safety functionality is accomplished. The object detector used in this project is YOLOv11n from Ultralytics.
4. **VLM:** Additionally, as far as the cost map is concerned, the tech pipeline applies an extra crucial cost, the social cost, created from the smolVLM-Instruct model, which takes as input a prompt and an image of each detected obstacle. The risk of that particular obstacle is asked to be evaluated with a score of 0.0 to 1.0. That final score is then multiplied by a fixed number depending on obstacle's type [Person by 1.0 or object by 0.3], to give higher priority on people's safety. Furthermore, its contribution to the final cost map is 60 percent or 0.6/1.0. Ultimately, smolVLM is given a prompt and the final generated path by the algorithm, to evaluate it on how well respects social norms and fast it is, by providing a score from 1 to 5. The output is quite useful for us to check smolVLM's reasoning and final algorithm's generated path.
5. **Path Planning** Finally, now that there is a proper image scene with detected and inflated objects and obstacles set up, the A* algorithm is activated and ready to make its decision. In this case, A* starts to adjust its optimal path finding mode to these new conditions, trying to find the shortest path without touching or routing near obstacles and at the same time respecting social norms like not passing through people, thus achieving a socially aware planning process.

The GitHub repository of this project is this: Repository

2.1 Path Planning - A*

The path planning algorithm used in this project is A*. The A* algorithm finds the correct balanced path by minimizing a cost function, f(n), which guides the search efficiently. The f(n)function is calculated as f(n) = g(n) + h(n). Then the g(n) function is the accumulated actual cost from the start to the current node, including the movement distance (baseCost), the total penalties from the cost map, and penalties for turning and curvature. The cost map is where the social and obstacle avoidance costs (VLM output and distance transform) are incorporated to push the path away from inappropriate zones. The h(n) function is the Euclidean heuristic, a direct line estimate of the remaining distance to the goal, which ensures the search is goal directed. The algorithm prioritizes nodes in a priority queue based on the lowest f(n) value, guaranteeing that the resulting path, if one exists, represents the most efficient and socially acceptable route according to the defined costs. [1] [2] [3]

2.1.1 Cost Map

In A* implementation, for start, three base cost penalties are applied to ensure a safe and smooth path. First, is the intrinsic Movement Distance, which is the fundamental cost of travel (1 for orthogonal moves, sqrt of 2 for diagonal). Second, a fixed Turn Penalty is added if the robot changes its movement direction from the previous step, as a result discouraging sharp, jerky turns. Then, a Curvature Penalty, weighted by curvature Weight, is calculated based on the angle between the two previous path segments, which promotes smooth, continuous movement, which ideal for this project's navigation path planning. Finally, an explicit Wall Cost is applied, which steeply increases the cost map near the map boundaries, ensuring the path maintains a safe distance from walls and avoids getting stuck and colliding. [4] [5] [6] [7]

2.2 Object Detection - YOLOv11n

This project uses YOLOv11n (yolo11n.pt) as the object detection component to identify objects that pose a risk to navigation, such as people and chairs. The map image is first resized to a fixed 320 x 320 resolution for stable detection before being fed into

the model. The model outputs bounding boxes for detected obstacles, which are then scaled back up to the planner's 250 x 250 grid size. These scaled boxes are used to mark occupied cells in a binary grid, forming the foundation of the physical obstacle map and contributing to the final cost landscape. [8] [9] [10] [11]

2.2.1 Cost Map

The costs directly resulting from YOLO detections are primarily the Obstacle Cost and its resulting impact on path eligibility. First, detected obstacles are subjected to a strong main dilation step, effectively inflating the bounding boxes by a safety radius to prevent the path from clipping corners. This inflated binary obstacle map is then used to calculate the distance transform, which measures the distance of every free cell to the nearest obstacle. This distance is converted into the Obstacle Cost by an inverse power function, severely penalizing paths that approach detected objects and walls.

2.3 Visual Language Model - smolVLM and Social Cost Map

The SmolVLM-Instruct model used for this project is a small specialized Vision-Language Model of a size of 1.7 billion parameters developed by Hugging Face company, used to inject social intelligence into the path planner. Its core task is to analyze and provide a risk score for each specific obstacle depending on its type. Finally, this VLM output, after being cleaned and normalized from 0.0 to 1.0 scale, becomes the final social cost, which serves as a semantic penalty layer in the path planning cost function. [12] [13] [14] [15]

2.4 Tests and Results

The original image, has two other versions, one with light noise and another one with heavier noise. They will be used to test the robustness of path planning and the limits of the pipeline. As is shown below, the object detection accuracy and costs strength depend extremely on image resolution, thus size and noise. Less resolution and more noise, lead to worse social norms, detection and prediction accuracy results. We have also tested images with different background scenes as well, such as Urban road, Laboratory, Nature and Playground, which contains different scenarios and obstacles, challenging even further ours algorithm's generalization. In the end, there are some resulted paths

generated by large LLMs/VLMs like ChatGPT-5, Gemini-2.5-Flash, Grok-4-Fast, Claude-Sonnet-4.5. The results from them are quite intriguing and unexpected.

So, we start by displaying the original image and its noise versions [Light and Heavy].

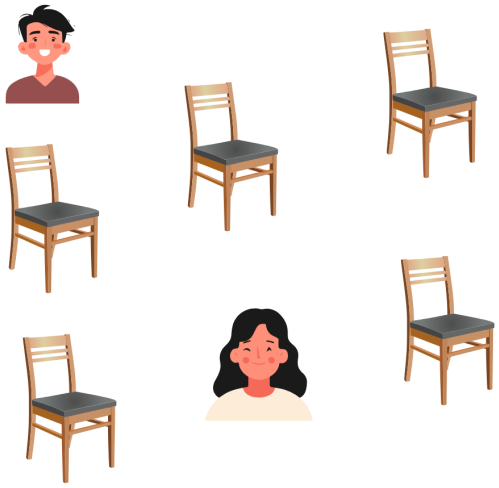

Figure 2.1
Original Image

Figure 2.2
Light Noise Image

Figure 2.3
Heavy Noise Image

2.4.1 Different size resolutions on the original image

In the no noise case, path planning was not correct in 50 x 50 resolution, but when the resolution got bigger, thus clearer, then we noticed that the algorithm was able to produce better paths, respecting social norms and restrictions. The most computationally and logically effective resolution size was 250 x 250 pixels. Meanwhile, object detection was wrong and false until the image got a 100 x 100 resolution. The other image resolutions had correct predictions and detections.

Original Image - No noise - Different sizes:

Below, there are presented the results from YOLOv11n object detection application on the image but with different sizes.

50: 2 traffic lights, 1 fire hydrant

100: 5 chairs

150: 2 persons, 5 chairs

200: 2 persons, 5 chairs

250: 2 persons, 5 chairs

450: 2 persons, 5 chairs

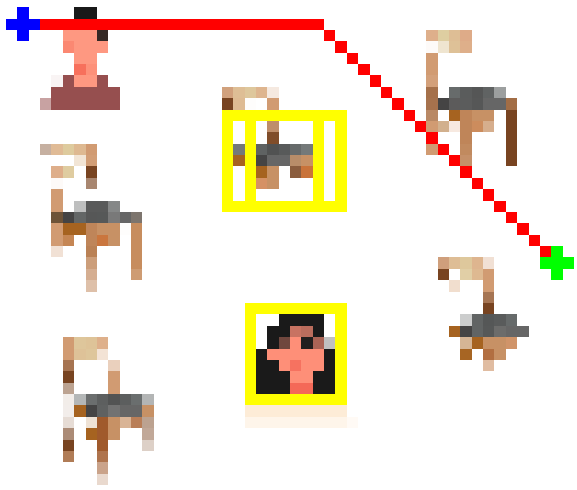

Figure 2.4
Image 50 pixels

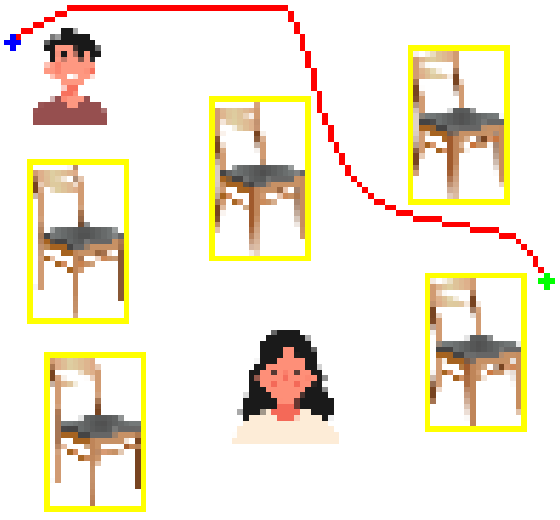

Figure 2.5
Image 100 pixels

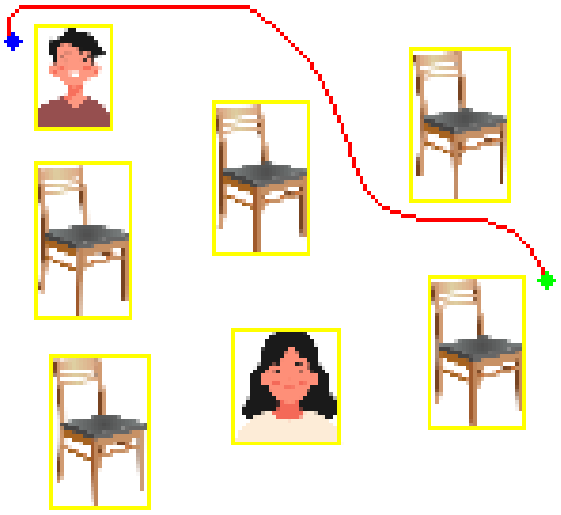

Figure 2.6
Image 150 pixels

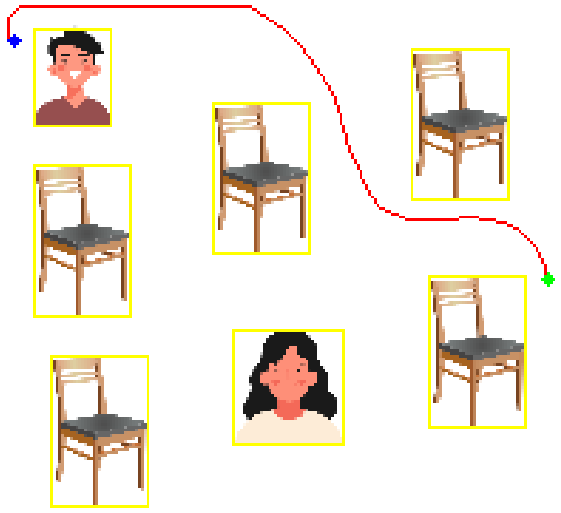

Figure 2.7
Image 200 pixels

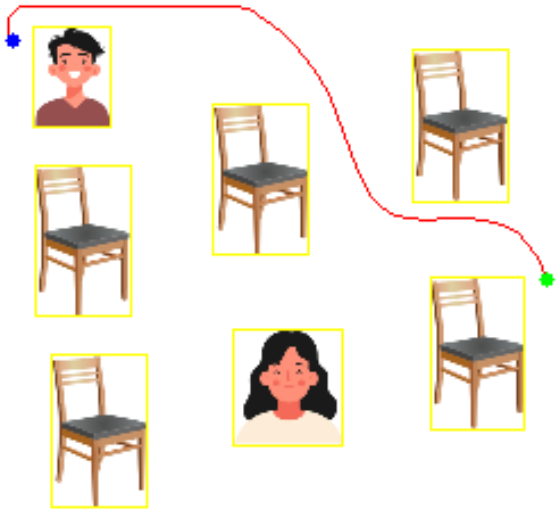

Figure 2.8
Image 250 pixels

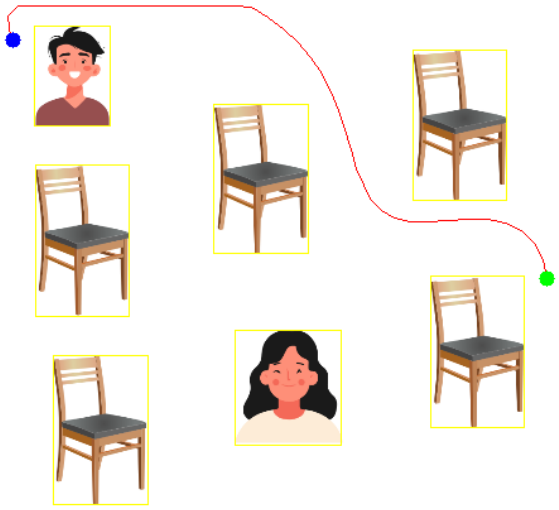

Figure 2.9
Image 450 pixels

The end result from the image with the most effective resolution size [250 x 250] is clearly showing that the generated path is following and respecting the geometric and social constrains. The path does not touch any person or obstacles, is smooth and reaches the

goal.

2.4.2 Different size resolutions with Light Noise

The results regarding light noise case, are that up to the 100 per 100 resolution both detection and paths are wrong, but from 150 x 150 to 250 per 250 we notice that path planning starts to improve and calibrate to the restrictions and rules set, as well as the detections occurs with better accuracy results. Finally, it is noticed that at 450 per 450 resolution both path and detections are correct. So, in light noise, the higher resolution helped to improve the robustness and effectiveness of the pipeline.

Original Image - Light noise - Different sizes:

50: (no detections)

100: 1 person,

150: 2 persons

200: 2 persons, 4 chairs

250: 2 persons, 2 chairs

450: 2 persons, 5 chairs

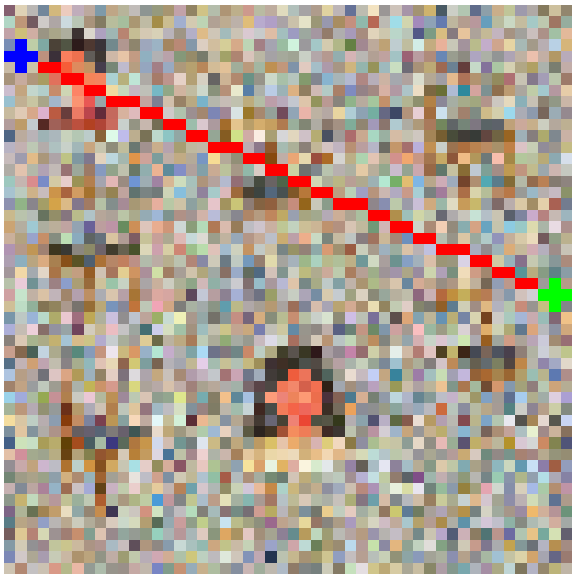

Figure 2.10
Image 50 pixels

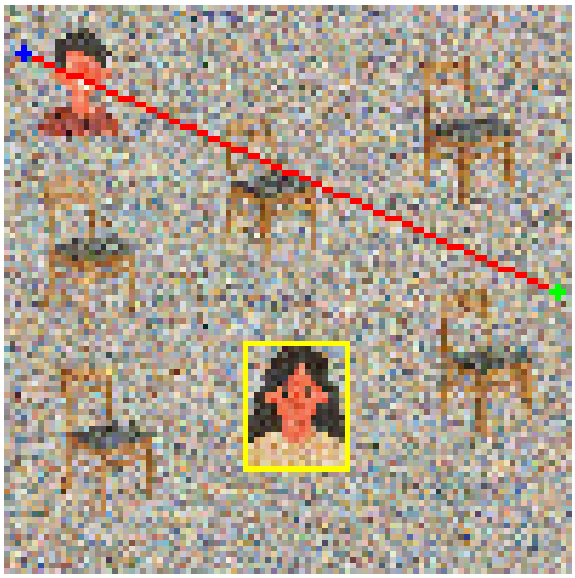

Figure 2.11
Image 100 pixels

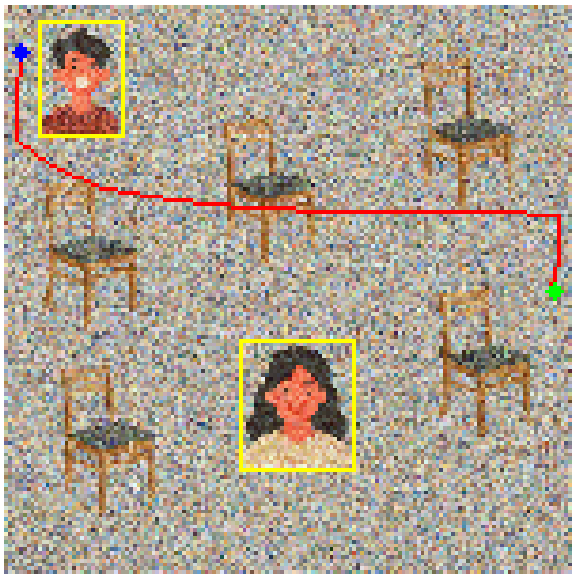

Figure 2.12
Image 150 pixels

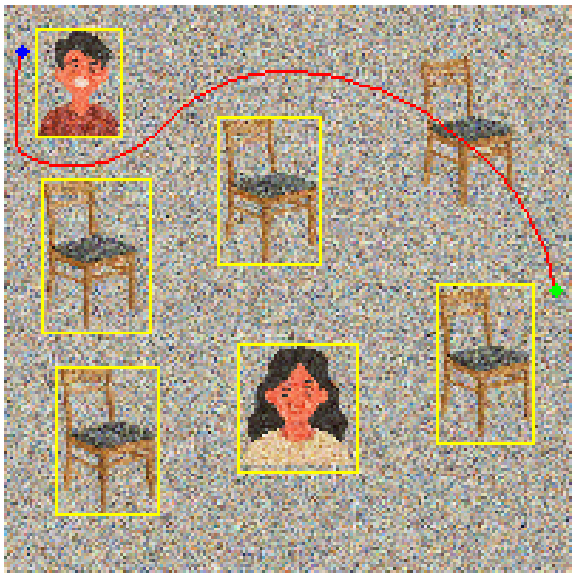

Figure 2.13
Image 200 pixels

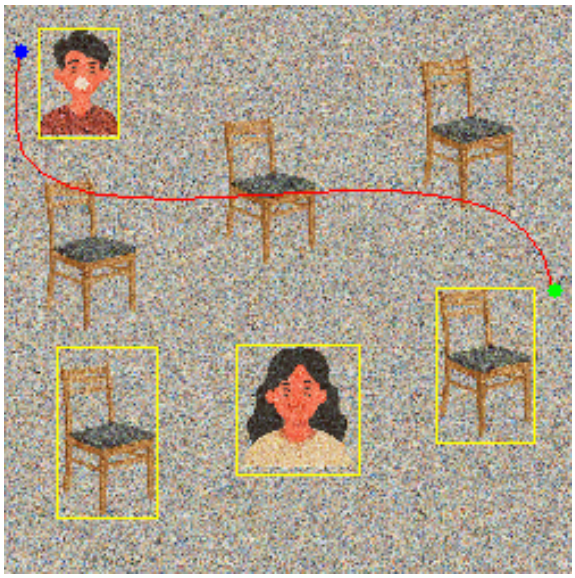

Figure 2.14
Image 250 pixels

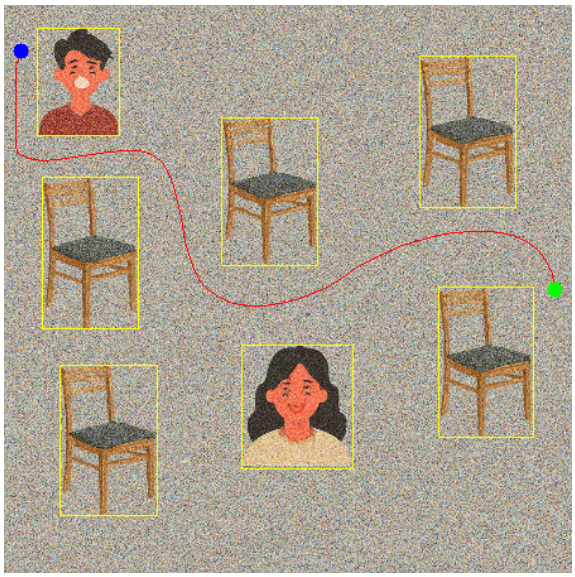

Figure 2.15
Image 450 pixels

The end result from the image with a resolution size of [450 x 450] is clearly showing that the generated path is following and respecting the geometric and social constrains. The path does not touch any person or obstacles, is smooth and reaches the goal. But the

disadvantage in this case, is that the size should be 450 x 450, thus high computational load, constraining real-time process in low-resource computers.

2.4.3 Different size resolutions with Heavy Noise

The results regarding heavy noise case, are indicating that all detections are either wrong or no detections at all. Similar behavior is noticed by the path planning system, which ignores all the set restrictions and goes directly to the goal[straight line], except the case where resolution is 400 per 400 pixels, there the path started to impove its robustness, not perfectly tho.

Original Image - Heavy noise - Different sizes:

50: (no detections)

100: (no detections)

150: 1 fire hydrant

200: (no detections)

250: (no detections)

400: 1 person, 2 chairs

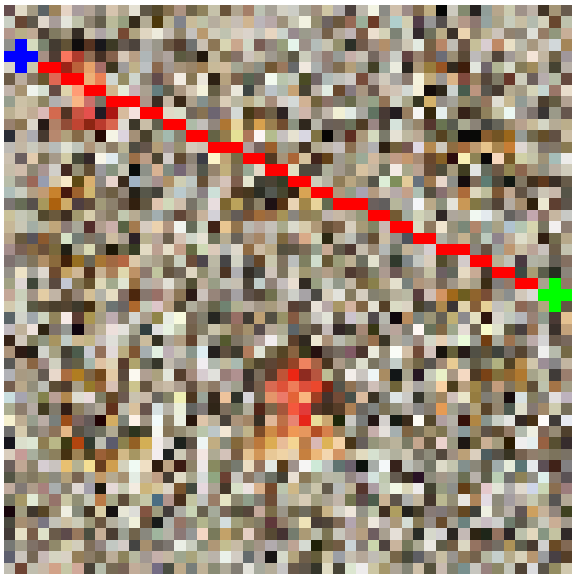

Figure 2.16
Image 50 pixels

Figure 2.17
Image 100 pixels

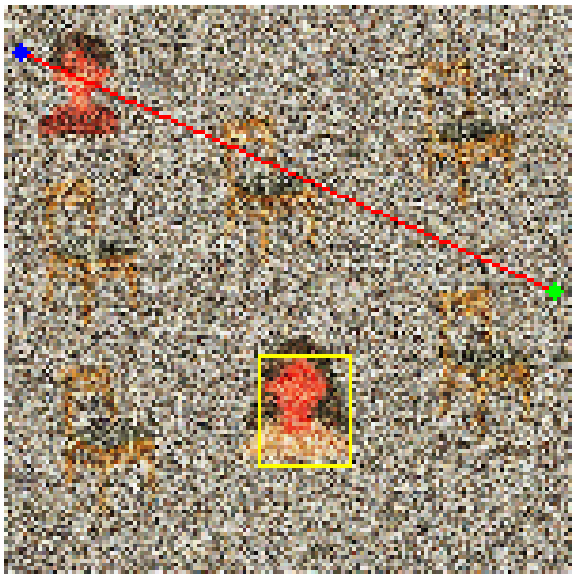

Figure 2.18
Image 150 pixels

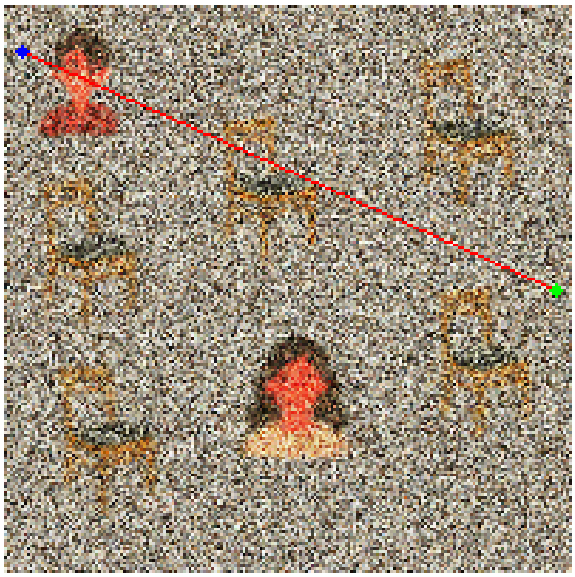

Figure 2.19
Image 200 pixels

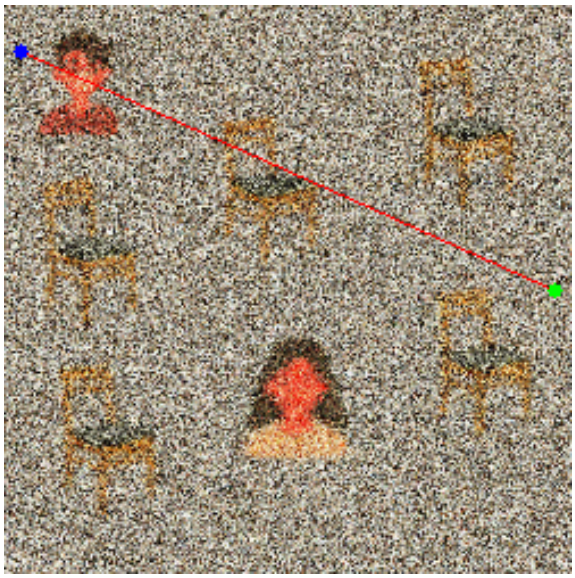

Figure 2.20
Image 250 pixels

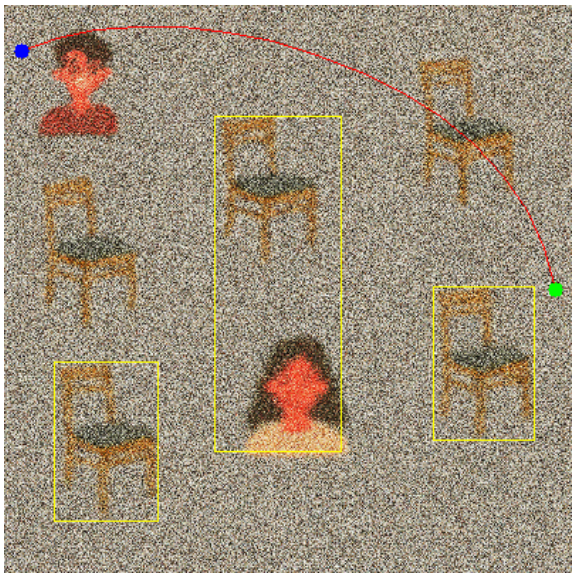

Figure 2.21
Image 400 pixels

The end result from the image with a resolution size of [400 x 40] is indicating that the generated path is improving by starting to make a curve, but because the object detector was not able to detect all the obstacles, it is passing through people and obstacles, thus,

that size displayed improvement but not full correctness. Similarly, the disadvantage in this case, is that the size is 400 x 400 pixels, thus high computational load, constraining real-time process in low-resource computers.

2.4.4 Different Positions - No Noise

The results for the image with different goal and start points, are that in no noise case there are 2 people, 5 chairs detected, which is correct. Similar results were generated by the case with light noise. On the other hand, when the image is with heavy noise, the detection is wrong, it displays that there are 3 people and 2 chairs detected. When the image is rich in heavy noise, it makes false predictions even in 450 per 450 resolution. The path planning was correct in all noise cases, showing that respects geometric and social restrictions.

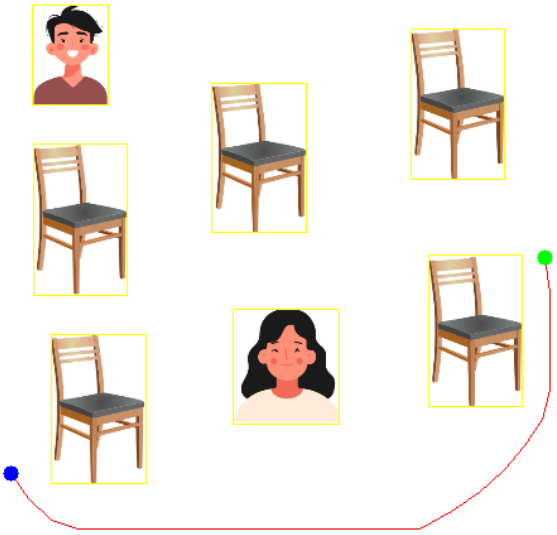

Figure 2.22
Original Image

2.4.5 Different Positions - Light Noise

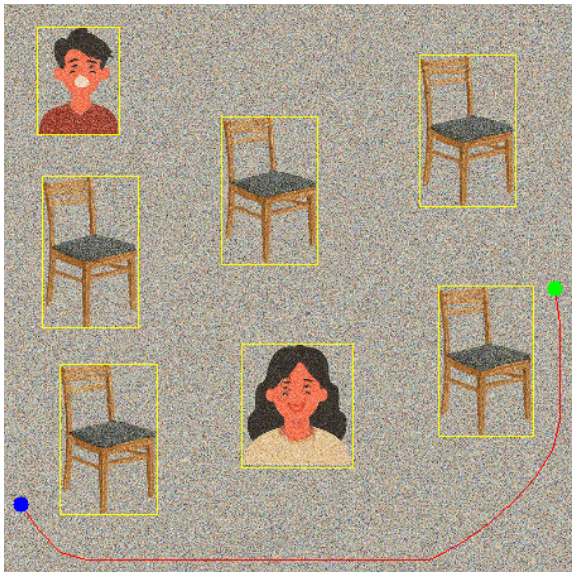

Figure 2.23
Light Noise Image

2.4.6 Different Positions - Heavy Noise

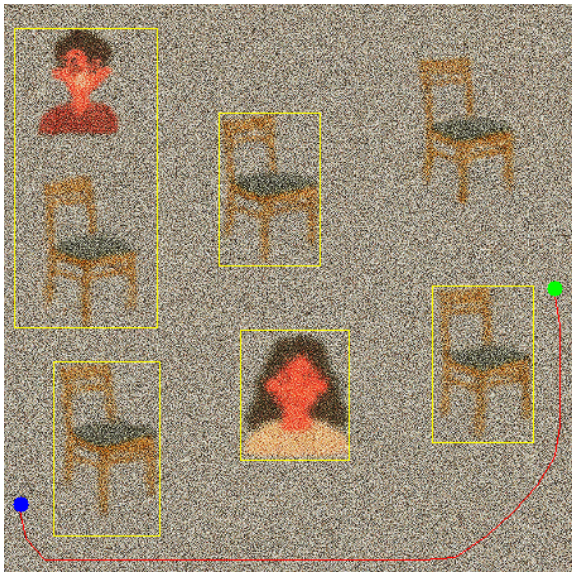

Figure 2.24
Heavy noise Image

2.4.7 Different Background Scenes

To test even more algorithm's robustness and scene generalizations on 2D static images, we used four scenes distinctive from each other. The four scenes are an Urban road, a Laboratory, a Nature and a Playground. So, starting with the results regarding the urban road, all people and obstacles are correctly detected and the path is avoiding collisions with them. The only flaw is that the red path is closer than it should be with the woman top right the blue car. Then, about the laboratory image, same people and obstacles are detected, and the path in this case it is carefully avoiding all of them. The red path as it can be seen is passing on the desk as well, but this happens because there is no depth in 2D images, and the desk is not perceived as an obstacle. Finally, same correct aware paths are generated on the images with nature and playground scenes.

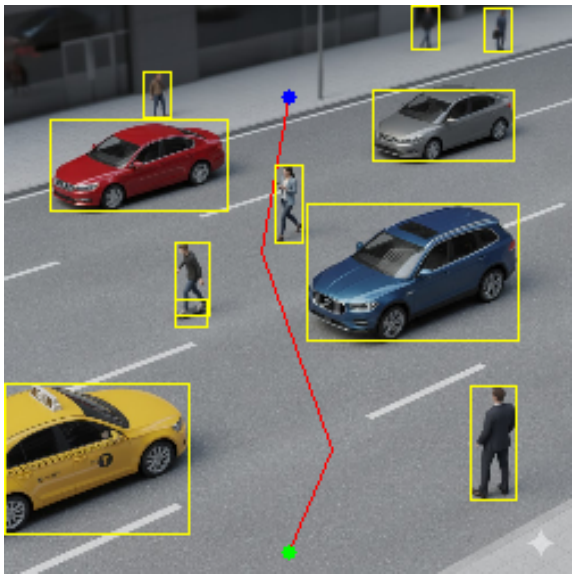

Figure 2.25
Urban Road Scene

Figure 2.26
Laboratory

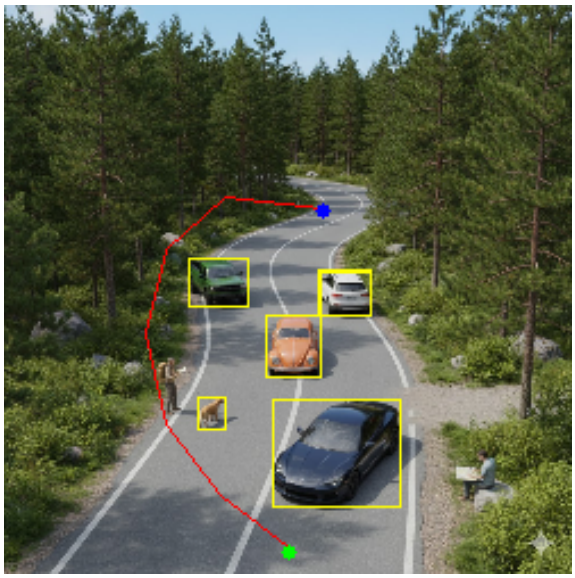

Figure 2.27
Nature

Figure 2.28
Playground

2.4.8 Comparing results on the original image with other LLMs/VLMs

Below, are the generated paths from some large LLMs/VLMs like ChatGPT-5, Gemini-2.5-Flash, Grok-4-Fast, Claude-Sonnet-4.5. As it can be seen, the paths are neither effective nor socially aware, showcasing that even these systems trained on the largest datasets with an all around data content, were not able to properly generate and provide a path that follows the rules and constrain.

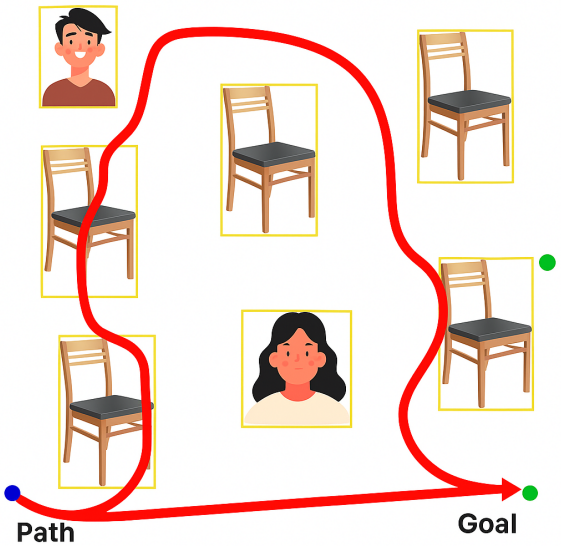

Figure 2.29
ChatGPT-5

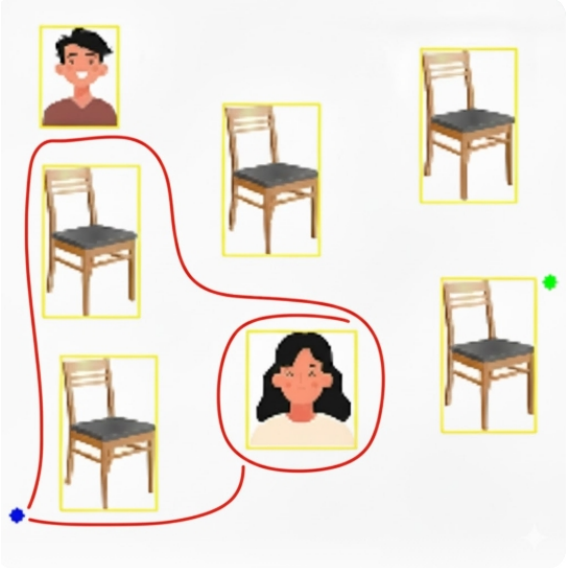

Figure 2.30
Gemini-2.5 Flash

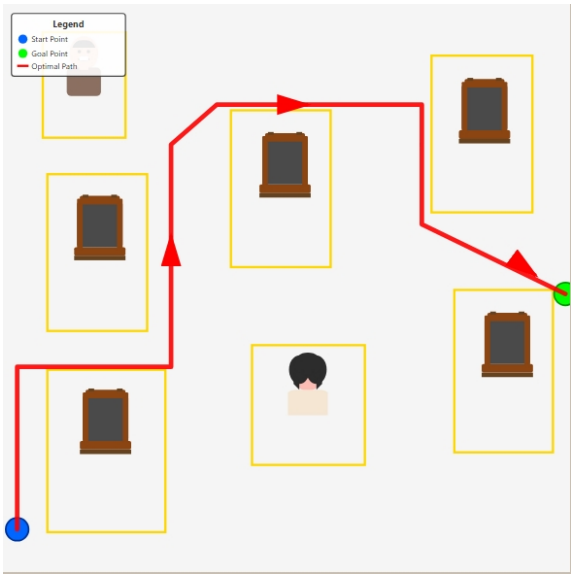

Figure 2.31
Claude-Sonnet-4.5

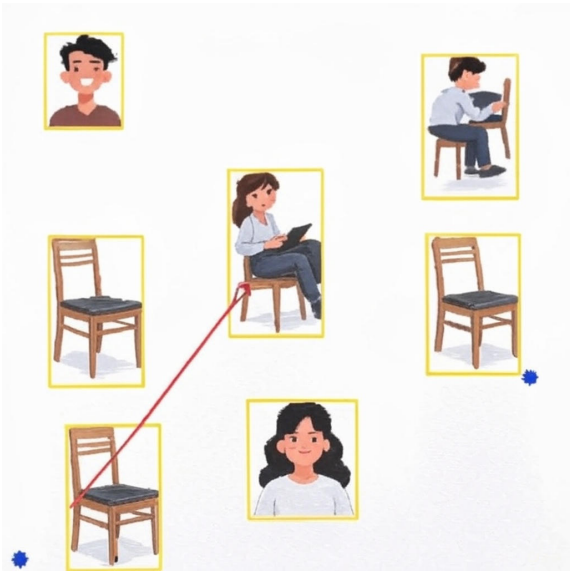

Figure 2.32
Grok-4-Fast

2.4.9 Comparing results on the different background sceneimages with other LLMs/VLMs

Below, are the generated paths from Gemini-2.5-Flash. The results are identical as previously mentioned. Again, the paths are neither effective nor socially aware, showcasing that they were not able to properly generate and provide a path that follows the rules and constrain on these different scenes as well. So, the logical output here from this experiment was that we cannot entirely depend on LLMs/VLMs for path planning reasoning and generation.

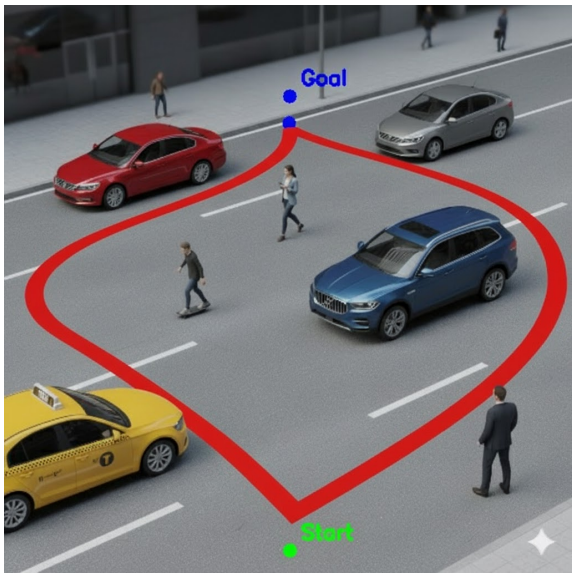

Figure 2.33
Urban Road

Figure 2.34
Laboratory

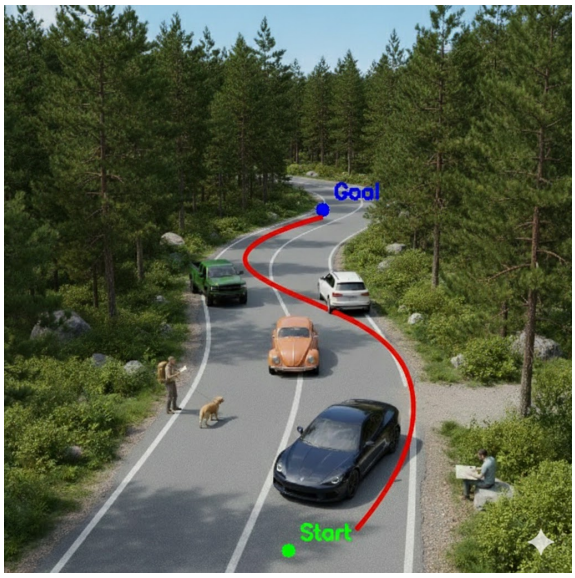

Figure 2.35
Nature

Figure 2.36
Playground

2.5 Path Explainability and Evaluation

In the path planning framework, SmolVLM provides an intuitive evaluation of the robot's trajectory by analyzing a visualized map image, where the route appears as a red line linking a green start marker to a blue goal. Guided by a concise prompt, the model assigns ratings on a 1 to 5 scale, with 1 being inadequate and 5 excellent for social awareness, emphasizing avoidance of pedestrian zones and seating to promote courteous navigation, and for optimality, focusing on route directness and efficiency to minimize travel time.

Finetuning SmolVLA for predicting Social Cost

Our chosen paper uses a general-purpose VLM model for estimating its social cost. We wanted to try a different approach using a vision—language—action model pretrained for controlling robots. This model will generate action trajectories based on image and text-based instructions. This enables more fine-grained social cost functions on the trajectory rather than the more discretised single action space they use in the paper (left, straight, right, and various discrete speed steps).

To enable a general-purpose robotics VLA model to estimate socially compliant trajectories, we needed to fine-tune the model to this behaviour. For this purpose, we used SCAND. It contains around 10 hours of teleoperated trajectories and sensor data. We first reformatted the data into a format that SmolVLA could read and be fine-tuned on. This involved resampling and resizing the videos captured by the robot. The data also had to be chopped into episodes consisting of 30 frames.

3.1 Finetuning a model on teleoperated data from a differential drive robot

An initial fine-tuning run was done on a small subset of the data with only the Jackal differential robot. This data only contains around 1 hour of teleoperation. A graph of the loss function over training epochs can be seen in Figure 3.1. The dataset was designed so that the input is a static instruction: "Continue along the path", an RGB image collected by a front-facing camera, and the current state described by a linear and angular velocity. The output of the model is 50 action vectors sampled at 1 Hz. One inference run of the model will produce a trajectory spanning 50 seconds.

In Figure 3.2 we see how the model performs without being finetuned much. This indicates the behaviour it would have without finetuning, showing why this is necessary. In Figure 3.3, the first finetuned model shows less variance and for angular velocity a better absolute error. There is a conversion error in the linear velocity. To produce this plot, we have tried to account for the normalisation and conversion back to actual units,

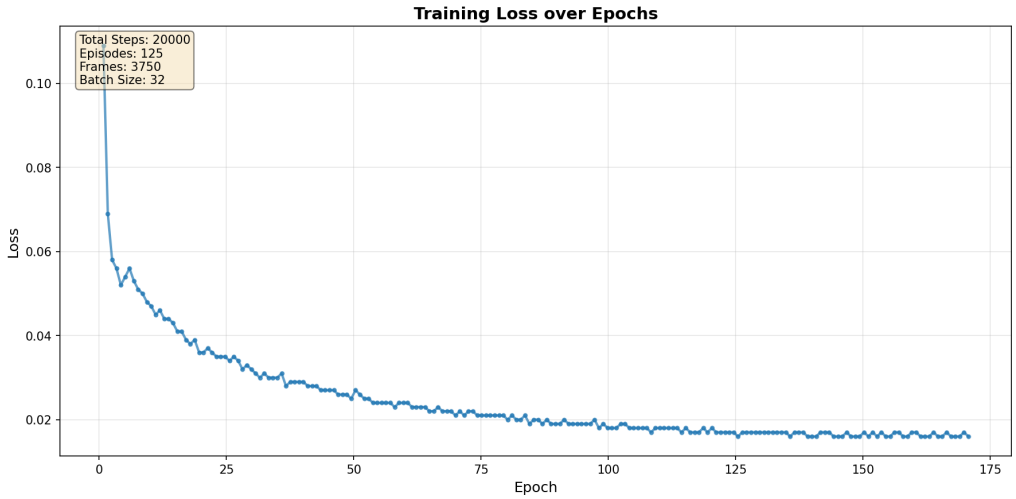

Figure 3.1: Initial finetuning loss with 20,000 training steps with the Jackal robot.

but there seems to have been some issue with that.

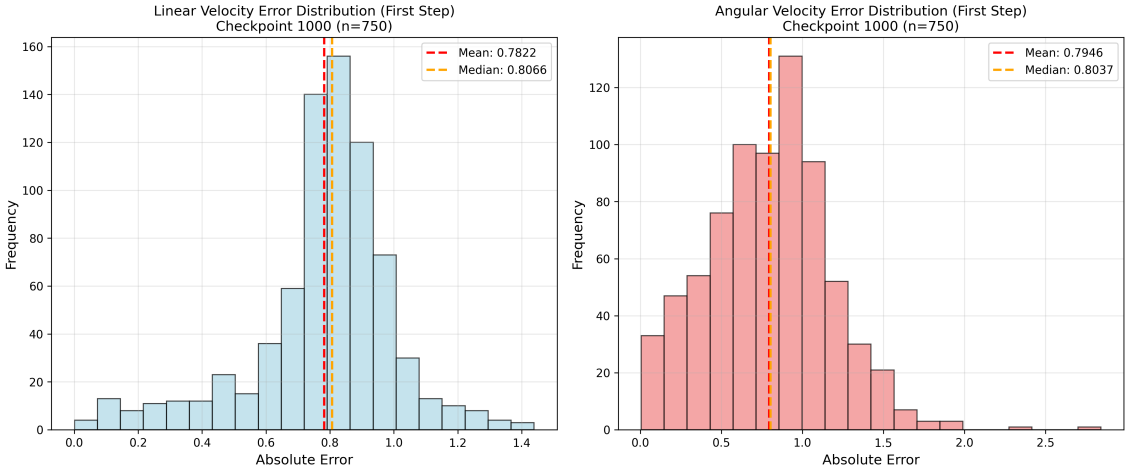

Figure 3.2: This is the model taken at 1,000 training steps. This is meant as a benchmark of how the untrained model performs. We see a large variance in the histogram. The error is calculated as the difference between the first predicted step and the actual step of the robot.

To further illustrate the importance of finetuning, an error plot of the predicted action vectors shows how the pretrained model uses 6 numbers for prediction in Figure 3.4 while our finetuned model successfully limits itself to 2 numbers in Figure 3.5, with just some small residual errors on the other dimensions.

Some other interesting plots can be seen in Figure 3.6 and Figure 3.7. These show 20 random trajectories generated by the 1,000-step model and the 20,000-step model.

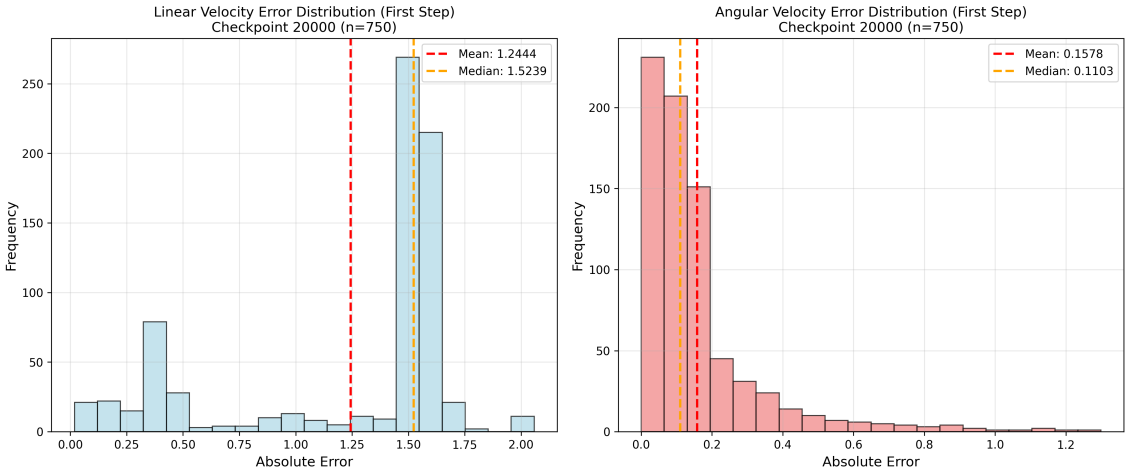

Figure 3.3: The 1st finetuned model. The error is calculated as the difference between the first predicted step and the actual step of the robot. The plot shows less variance and is centred better at 0 for the angular velocity. The linear velocity seems to have a shifted prediction.

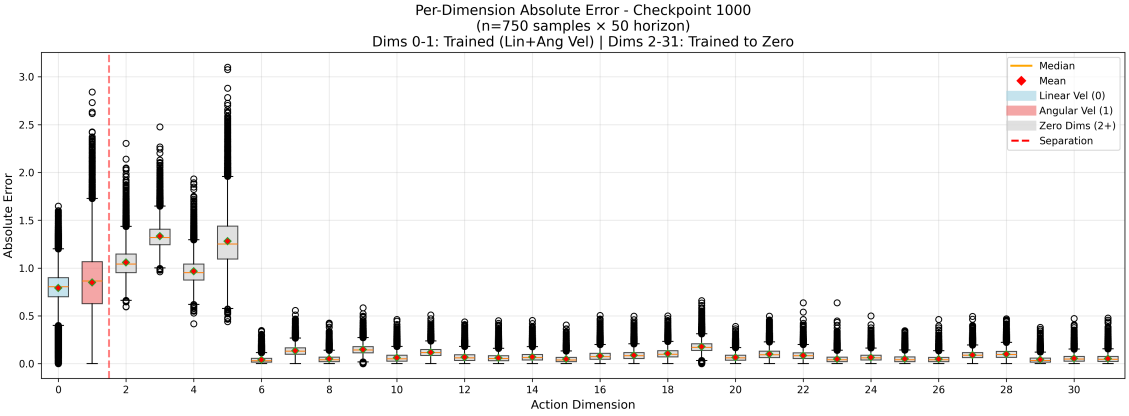

Figure 3.4: The dimension error of the first action vector in a predicted trajectory compared to ground truth labels by the model trained for only 1,000 steps.

Notice how the 20,000-step model generates smoother and more plausible trajectories, but seems to start spiralling towards the end of the trajectory.

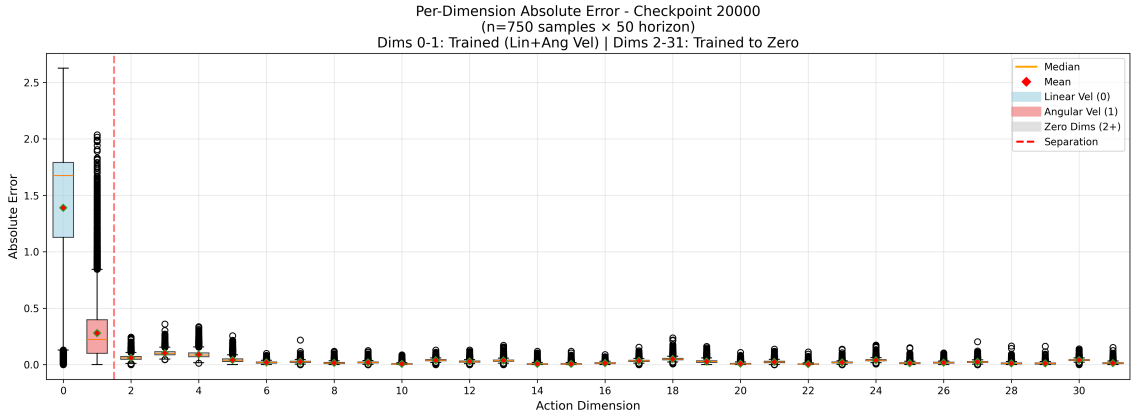

Figure 3.5: The dimension error of the first action vector in a predicted trajectory compared to ground truth labels by the model trained for 20,000 steps.

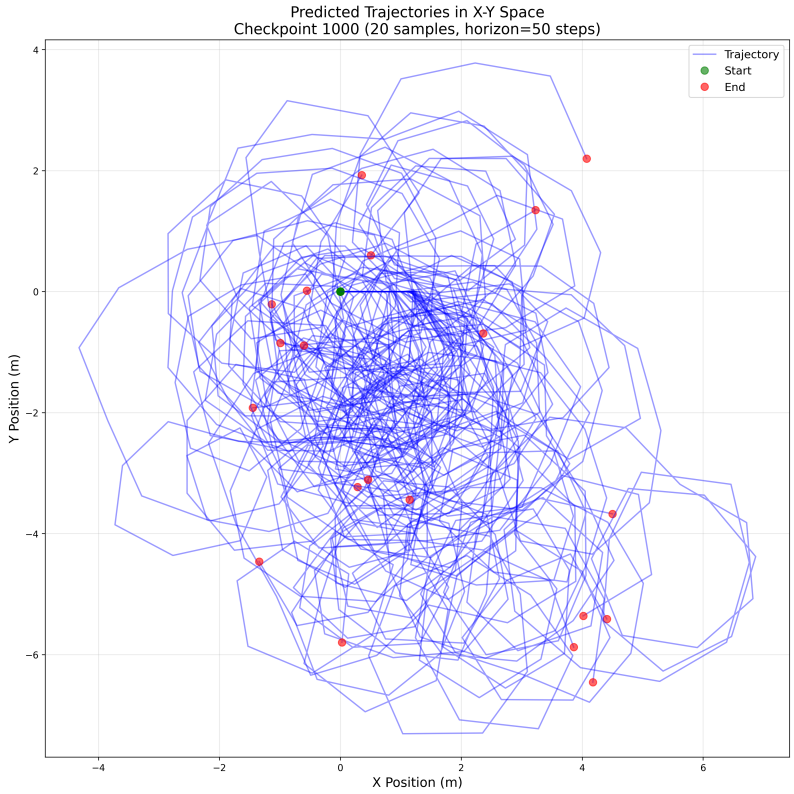

Figure 3.6: 20 randomly sampled trajectories from the 1,000-step model.

3.2 Finetuning a model on teleoperated data from a quadroped drive robot

For the second training run, we trained exclusively on the data from SCAND collected on the quadroped robot: Spot. 40% of this data is the spot going backwards. This means

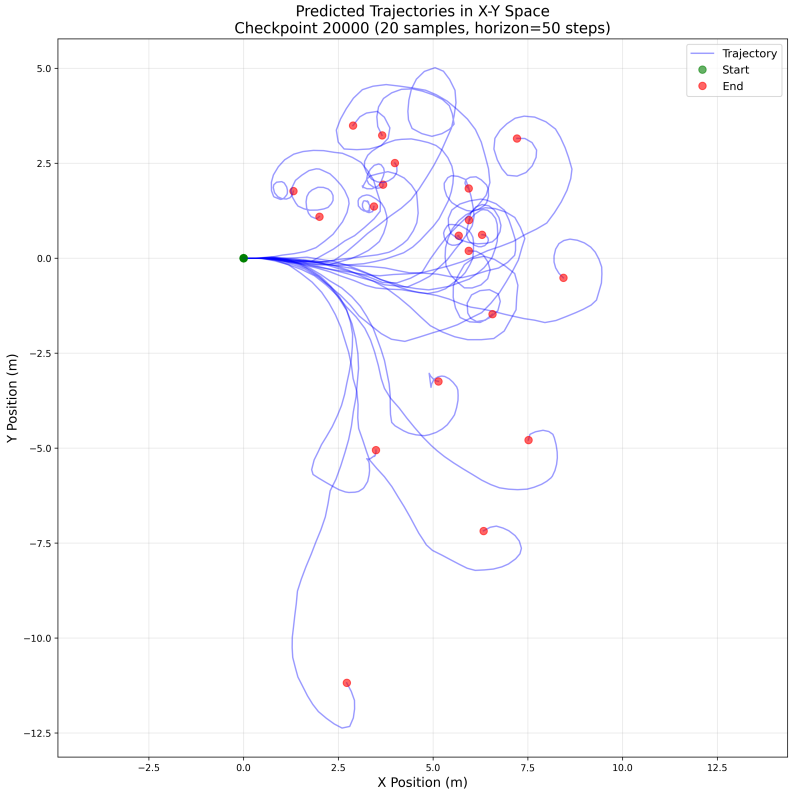

Figure 3.7: 20 randomly sampled trajectories from the 20,000-step model.

that when we tested on the Jackal data, the VLA had a hard time generalising. The test set was all of the data collected from the Jackal. This becomes a comparison of how embodiment impacts performance. A few other things were changed in this training run.

The prediction horizon is still 50 steps, but they are now sampled at 4 Hz, so that means an inference run produces 12.5 seconds of trajectory.

The model no longer predicts trajectories based on a static instruction but now knows more about where it is supposed to go. An instruction can be something like: "Go 8.2m forward and 0.4m to the left". The motivation for this is to be able to steer the model better.

The dataset now also contains temporal data. For each inference run, the model will use the current image and state, as well as the same information from 0.1 and 0.2 seconds ago. The motivation for this is to enable the model to learn better object dynamics and permanence.

This model was trained with a batch size of 16 compared to earlier training with 32, since VRAM requirements for this training run were higher. The model was trained for 30,000 steps. More details can be seen in Figure 3.8

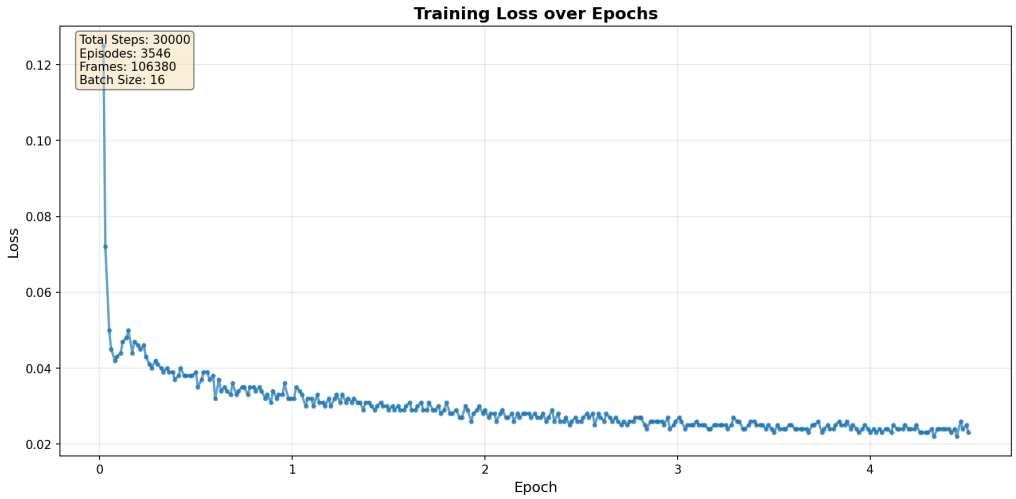

Figure 3.8: Loss over epoch for finetuning on the Spot teleoperation data.

[!h]

As can be seen in Figure 3.9, the first step error shows similar but worse performance for this cross-embodiment task. There is still the substantial error of the linear velocity.

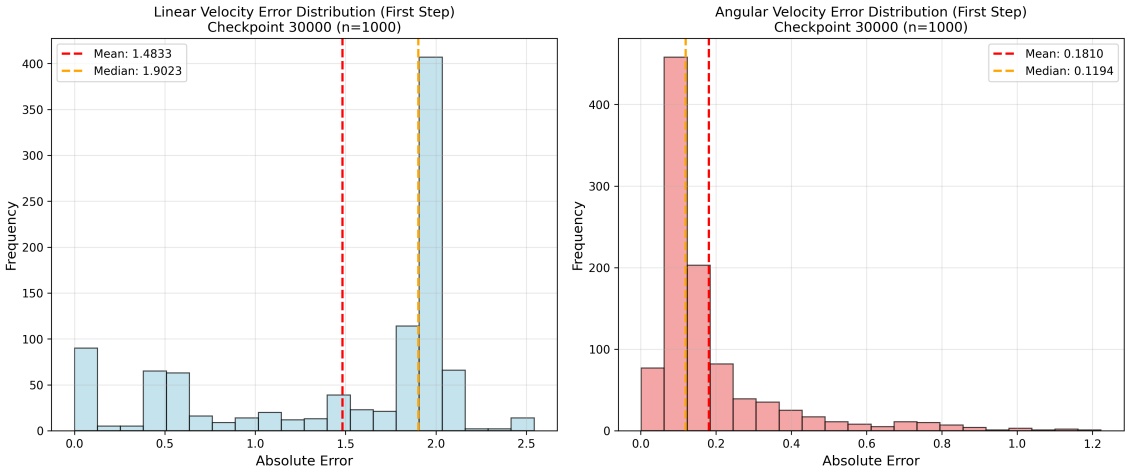

Figure 3.9: Error distribution from the first step compared to the ground truth labels.

In Figure 3.10 and Figure 3.11, we can see some generated trajectories. The trajectories seem more well-behaved than before, but are also not as long. Also, there seems to be a

clear trend in the trajectories, even though they are randomly sampled.

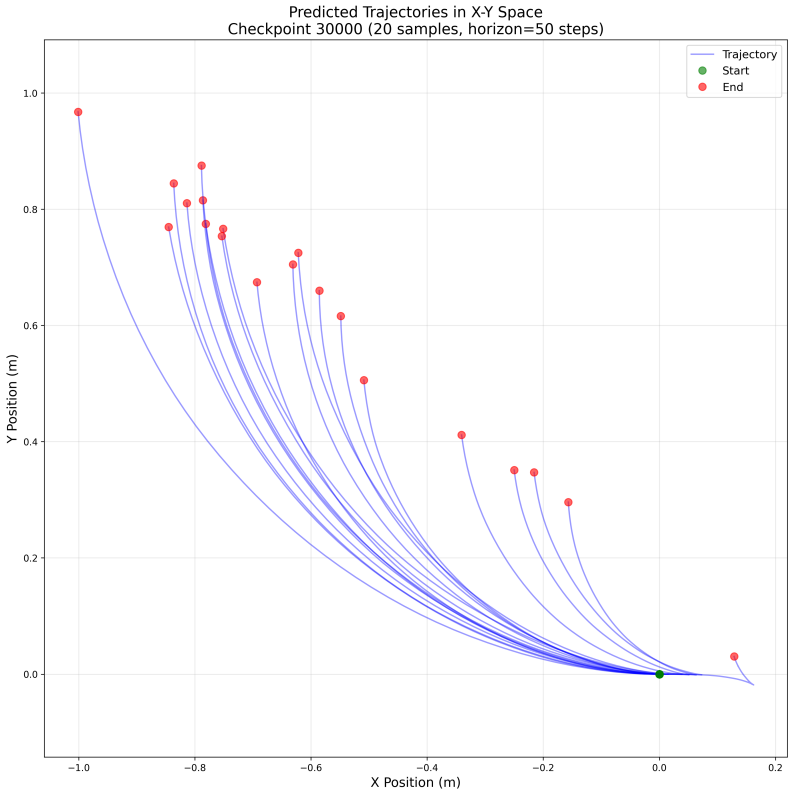

Figure 3.10: Trajectories from 20 randomly sampled instances from the test set.

To make the results more relatable, a random pick of 3 samples from both the first and second finetuned models on their respective test datasets. This can be seen in Figure 3.12 and Figure 3.13. We can see that there is still some way to go to make this reliable. Predicted velocities are too low, and the Spot finetune shows how it tends to either stand still or move backwards.

The most dominant error seems to be the shifted linear velocity. We suspect it stems from a normalization or conversion error. For a more robust model, it might be worth adapting the data to better fit the embodiment of a specific robot, or training with more varied data across more embodiments with better data distributions. [16] [17] [18]

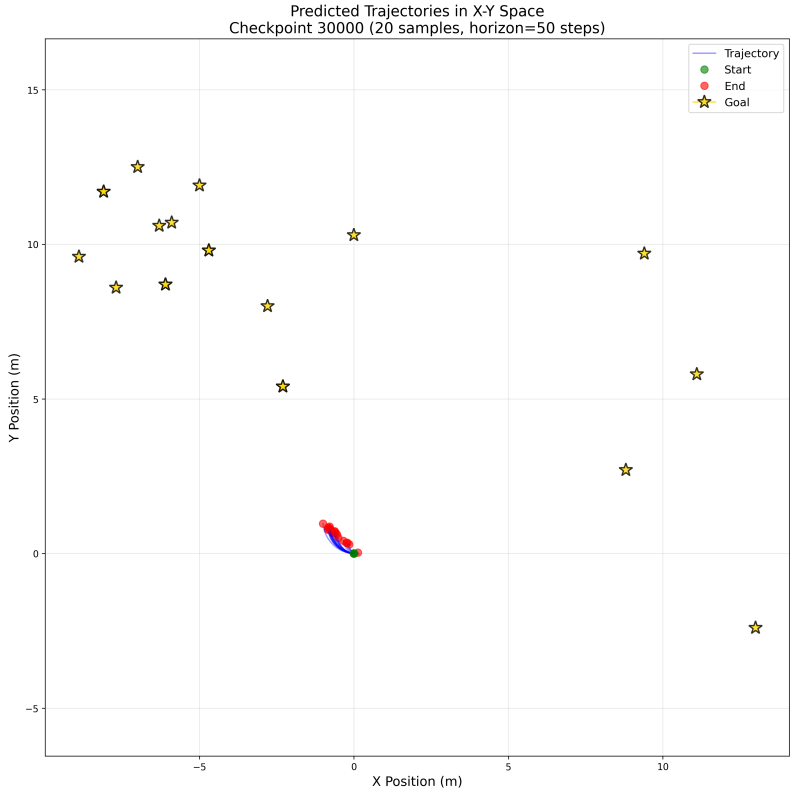

Figure 3.11: Trajectories and goals from 20 randomly sampled instances from the test set.

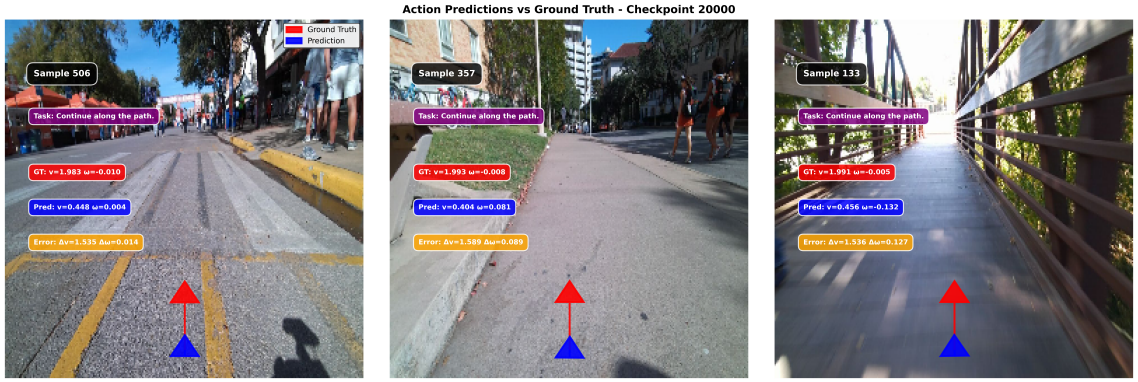

Figure 3.12: 3 random samples showing the ground truth and predicted movement vector for the model trained and evaluated on Jackal data.

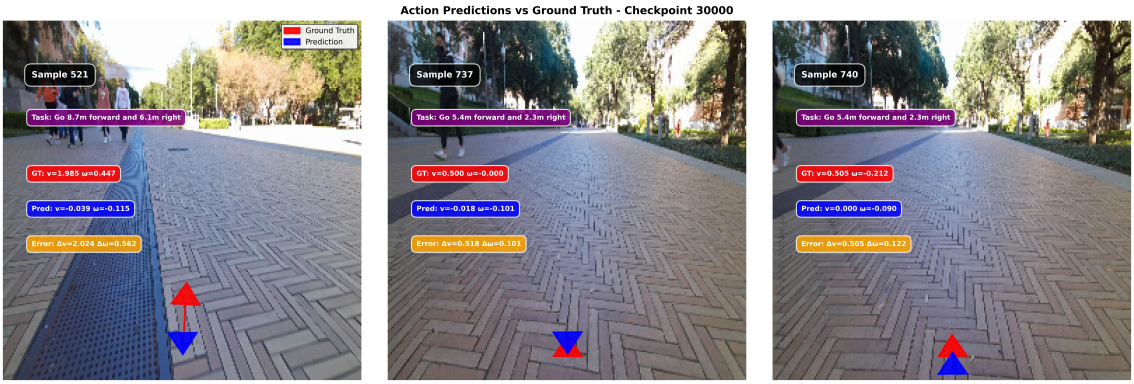

Figure 3.13: 3 random samples showing the ground truth and predicted movement vector for the model trained on only Spot data and evaluated on Jackal data.

Challenges and Limits

Below are presented some of the most important challenges and limits that we have been dealing with.

1. **Tuning:** The final full algorithm has a lot of parameters needed to be tuned, so a small change in values can change and disrupt the effectiveness of the generated paths.

2. **Prompts:** The VLM [smolVLM] used for this project needed careful and precise prompt engineering in order to provide and generate the desired output. Sometimes, changes of words in the sentence were able to competely flactuate the path, because smolVLM is also used for the social cost, which has a 60 percent [majority] contribution on the final cost map, as a result affecting the most the generated path.

3. **Latency:** Because of the use of a CPU only computer and many computationally hungry technologies like object detectors and VLMs, the gerated path was provided after 7 to 8 minutes. Therefore, establishing this project for that computer, impossible for real-time processing.

4. **2D World:** The current algorithm implementation is utilized only for a 2D world and scenes, it cannot comprehend depth and other 3D world needs and attributes.

5. **Hardware Constrain** We notice that when smolVLM was asked to provide elaborated explanations and outputs, it was not able to achieve that, instead it was proving non-sense data/results. We understood that the main cause of this issue was computer's hardware limits. A GPU is a crucial feature needed for a computer system to handle heavy computing loads and in the end offer useful and meaningful output. Ultimately, We solved this limit by asking smolVLM to offer a single value for each query and not sentences as reslults/output.

Future Scope

The next step for this project is to rearrange all of its core functionalities and set it to be able to perform in a 3D environment. In addition, we aim to use it on a robotic system such as Tiago Pro from a company named PAL. It is a great opportunity to experiment with challenges and solutions of a 3D world, as well as to see in live demonstration a real-life robot utilizing the concept and methods of the new algorithm and system regarding socially aware navigation and low cost path planning.

Conclusion

Taking everything into account, this project showcases that by making use of state-of-the-art technologies such as object detectors and visual language models, the process of path planning by a search algorithm, for instance, in this case A*, is able to deliver and provide a more robust and generalized end-result, respecting the geometric restrictions as well as the social norms between people, where at the same time choosing the least costly, thus the shortest path possible. These combinations and results have a great chance to increase the trust and credibility between people and robots, as well as offer better debugging opportunites to operators and developers. The insights, knowledge and skills that our team learned and gathered from this adventure are the absolute stepping stones for a new further research on these topics and cases.

Bibliography

[1] MIT Center for Bits and Atoms, "Robotic path planning," Course Material: How to Make (Almost) Anything - MAS.865, 2021, accessed: 3 Nov. 2025. [Online]. Available: https://fab.cba.mit.edu/classes/865.21/topics/path_planning/robotic.html

[2] GeeksforGeeks, "A* search algorithm," Online Tutorial, 2024, accessed: 3 Nov. 2025. [Online]. Available: https://www.geeksforgeeks.org/dsa/a-search-algorithm/

[3] V. K. Cheedella, "Robotic path planning: A*," Online Blog, Medium, 2024, accessed: 3 Nov. 2025. [Online]. Available: https://medium.com/@cheedellavamsikishore/robotic-path-planning-a-star-96a2c8f9467d

[4] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," ICRA 2008 Workshop Paper, 2008, accessed: 3 Nov. 2025. [Online]. Available: https://www.cs.cmu.edu/~maxim/files/costmapplan_icra08ws.pdf

[5] MathWorks, "Vehicle costmap (automated driving toolbox)," Online Documentation, 2025, accessed: 3 Nov. 2025. [Online]. Available: https://www.mathworks.com/help/driving/ref/vehiclecostmap.html

[6] B. Yang, "Icra 2021 project page," Project Website, 2021, accessed: 3 Nov. 2025. [Online]. Available: https://byangw.github.io/projects/icra2021/

[7] L. E. Kavraki *et al.*, "Arches: Toward agile robot motion planning architectures," Motion Planning Workshop Paper, 2019, accessed: 3 Nov. 2025. [Online]. Available: https://motion-planning-workshop.kavrakilab.org/papers/arches.pdf

[8] S. Slazebni, "Lecture 7: Object detection," University of Illinois Urbana-Champaign Lecture Notes, 2017, accessed: 3 Nov. 2025. [Online]. Available: https://slazebni.cs.illinois.edu/spring17/lec07_detection.pdf

[9] Unknown, "Preprint arxiv:2504.18136," *arXiv preprint*, vol. arXiv:2504.18136, 2025, accessed: 3 Nov. 2025. [Online]. Available: https://arxiv.org/pdf/2504.18136

[10] Ultralytics, "Yolo11 models documentation," Online Documentation, 2025, accessed: 3 Nov. 2025. [Online]. Available: https://docs.ultralytics.com/models/yolo11/

[11] ——, "Object detection task documentation," Online Documentation, 2025, accessed: 3 Nov. 2025. [Online]. Available: https://docs.ultralytics.com/tasks/detect/

[12] Hugging Face, "Smolvlm: A compact vision-language model for efficient multimodal reasoning," Official Blog, 2025, accessed: 3 Nov. 2025. [Online]. Available: https://huggingface.co/blog/smolvlm

[13] ——, "Smollm: Small, efficient open llms," GitHub Repository, 2025, accessed: 3 Nov. 2025. [Online]. Available: https://github.com/huggingface/smollm

[14] H. F. R. Team, "Smolvlm: Small vision-language models," *arXiv preprint*, vol. arXiv:2504.05299, 2025, accessed: 3 Nov. 2025. [Online]. Available: https://arxiv.org/pdf/2504.05299

[15] Hugging Face, "Smolvlm-instruct model card," Model Repository, 2025, accessed: 3 Nov. 2025. [Online]. Available: https://huggingface.co/HuggingFaceTB/SmolVLM-Instruct

[16] ——, "Smolvla: Efficient vision-language-action model trained on lerobot community data," Online Blog, 2025, accessed: 3 Nov. 2025. [Online]. Available: https://huggingface.co/blog/smolvla

[17] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti, S. Alibert, M. Cord, T. Wolf, and R. Cadene, "Smolvla: A vision-language-action model for affordable and efficient robotics," *arXiv preprint*, vol. arXiv:2506.01844, 2025, accessed: 3 Nov. 2025. [Online]. Available: https://arxiv.org/pdf/2506.01844

[18] H. Karnan, A. Nair, X. Xiao, G. Warnell, S. Pirk, A. Toshev, J. W. Hart, J. Biswas, and P. Stone, "Socially compliant navigation dataset (scand): A large-scale dataset of demonstrations for social navigation," *arXiv preprint*, vol. arXiv:2203.15041, 2022, accessed: 3 Nov. 2025. [Online]. Available: https://www.cs.utexas.edu/~xiao/SCAND/SCAND.html