Dynamic Multi-Agent Pathfinding

Student(s)

Deren Ege Turan, Emre Hilmi Songur, Farukhzhon Barotov, M. Sami Yavuz, Aysu Bogatarkan, Batuhan Yıldırım, Cem Yüksel

Faculty Member(s)

Esra Erdem, Volkan Patoğlu

Sabancı Üniversitesi

PURE PROGRAM FOR UNDERGRADUATE RESEARCH

ABSTRACT

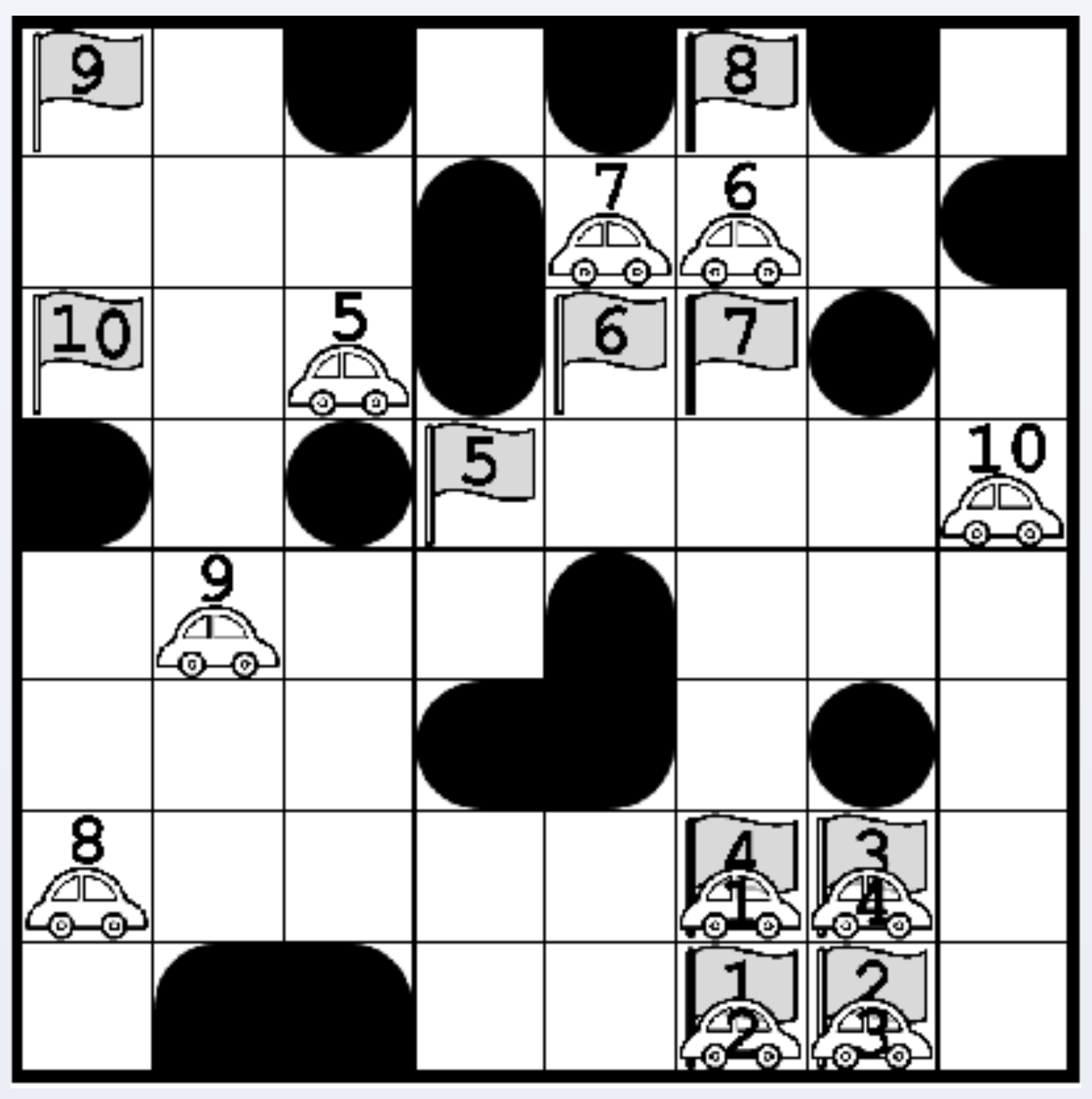


Figure 1: Pathfinding scheme with agents, obstacles and grids.

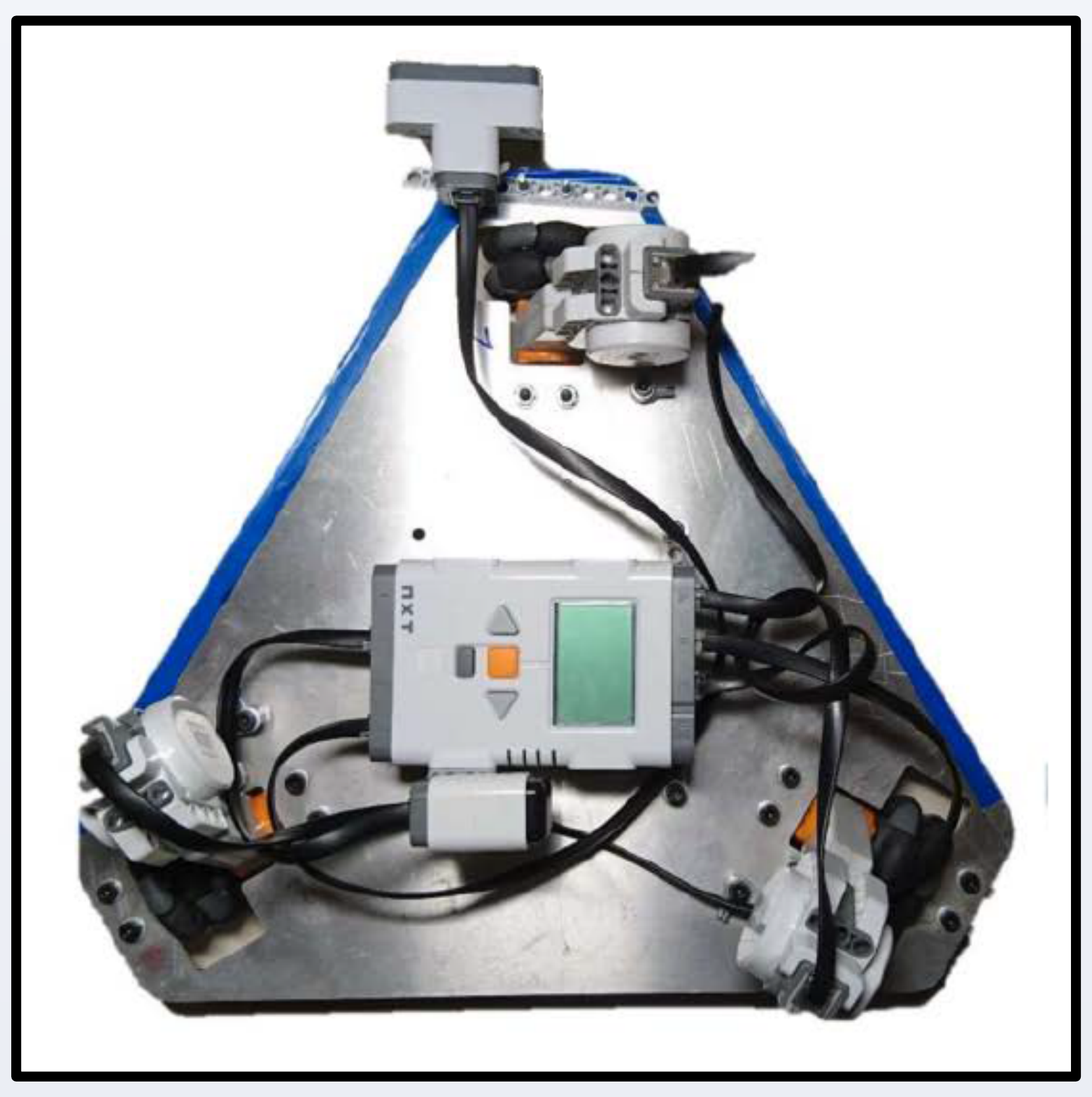


Figure 2: The omnidirectional LEGO robot's bird's-eye-view photograph.

In multi-agent path finding (MAPF), multiple agents need to find paths from their respective locations to their goal locations, ensuring that the paths do not collide with

- obstacles or
- other moving agents.

This problem is important for many applications, such as

- computer games,
- environmental monitoring,
- patrolling,
- multi-robot planning, etc.

This project involves studying MAPF in a dynamic environment, using graph theory and AI methods.

OBJECTIVES

Pathfinding for a single agent is the problem of planning a route from an initial location to a target location in an environment, going around obstacles. Pathfinding for multiple agents (PF) also aims to plan such routes for each agent, but subject to various constraints, such as no self-intersecting paths, no intersection of paths/plans, no crossing/meeting each other, no waiting idle, restrictions on the length of each path/plan and on the total length of paths/plans, and requirements on visiting multiple target locations/waypoints (Erdem et al., 2013).

Besides, we were supposed to design a compansetor that works on a globally-positioned camera by image recognition which is expected to compensate the loss from slippage on the next time step. Since the positioning system runs in real-time, we had to resolve the issue of running Simulink in external mode connected to the EV3 devices by TCP/IP wi-fi module with a wi-fi dongle.

DETAILS

During the project, we solved different versions of multi-agent path finding (MAPF) problem using answer set programming (ASP) and its solver Clingo. 1-Finding a path starting from an initial point to a goal point in a graph G = (V, E), an initial vertex s in V, a goal vertex g in V as inputs. 2- Finding a path starting from an initial point to a goal point in a directed graph G = (V, E), an initial vertex s in V, a goal vertex g in V, and a set W of waypoints as inputs. 3- In this step, the aim is to find plans for k robots, k is the number of robots. Now the environment is viewed as a special sort of graph: a grid of size n*m, and each robot has a delivery task to complete within u time steps. Inputs are given as: an initial location, a goal 3 location, set of waypoints and an upper bound on u on the plan length. The plan length should be at most u, such that no two robots collide with each other, no two robots can be at the same location at the same time, no two robots can swap their locations along the same edge. This problem is called the Multi-Agent Path Finding (MAPF) problem with waypoints.

Compensator Design: Detection of ArUco Markers

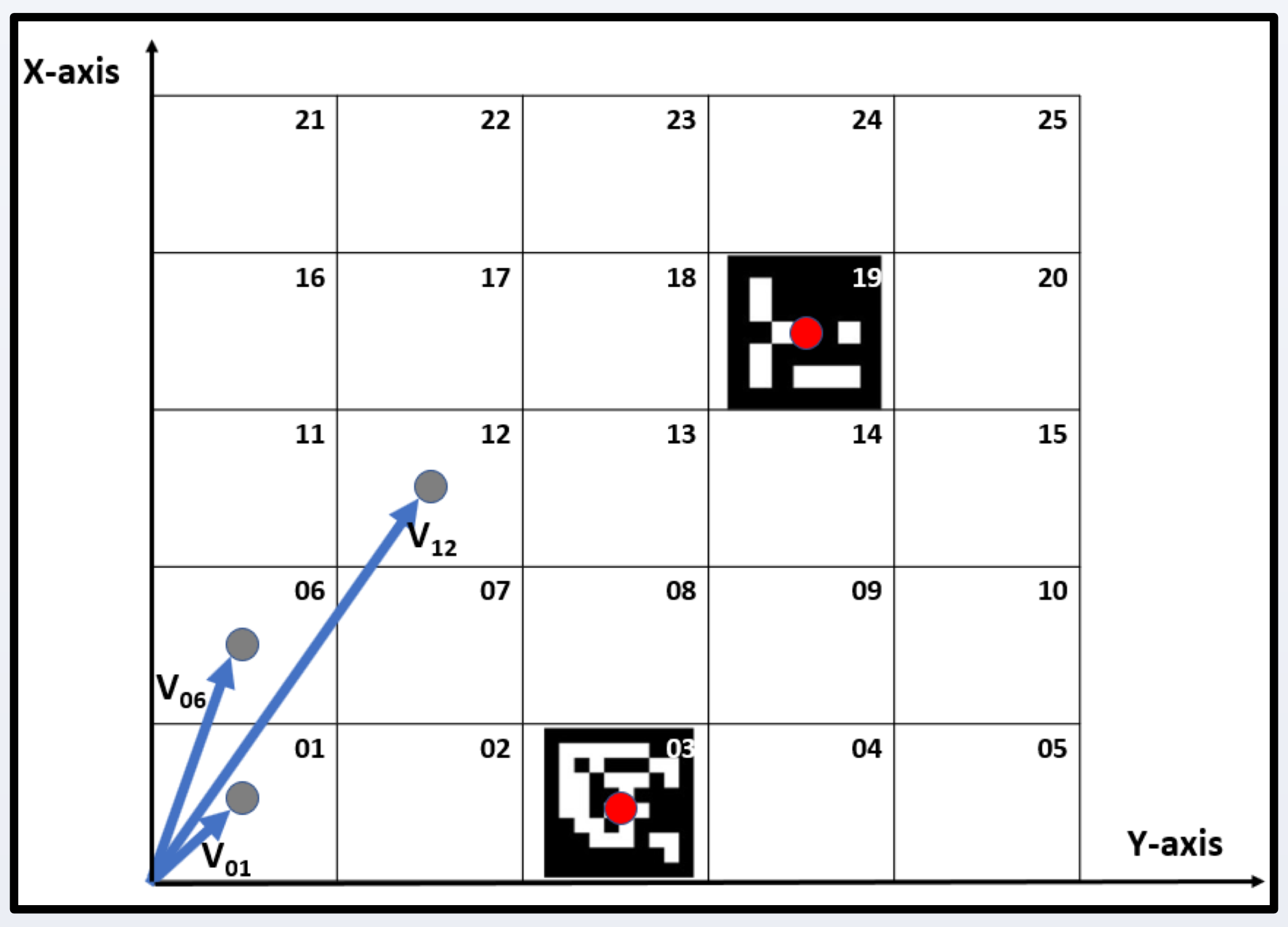
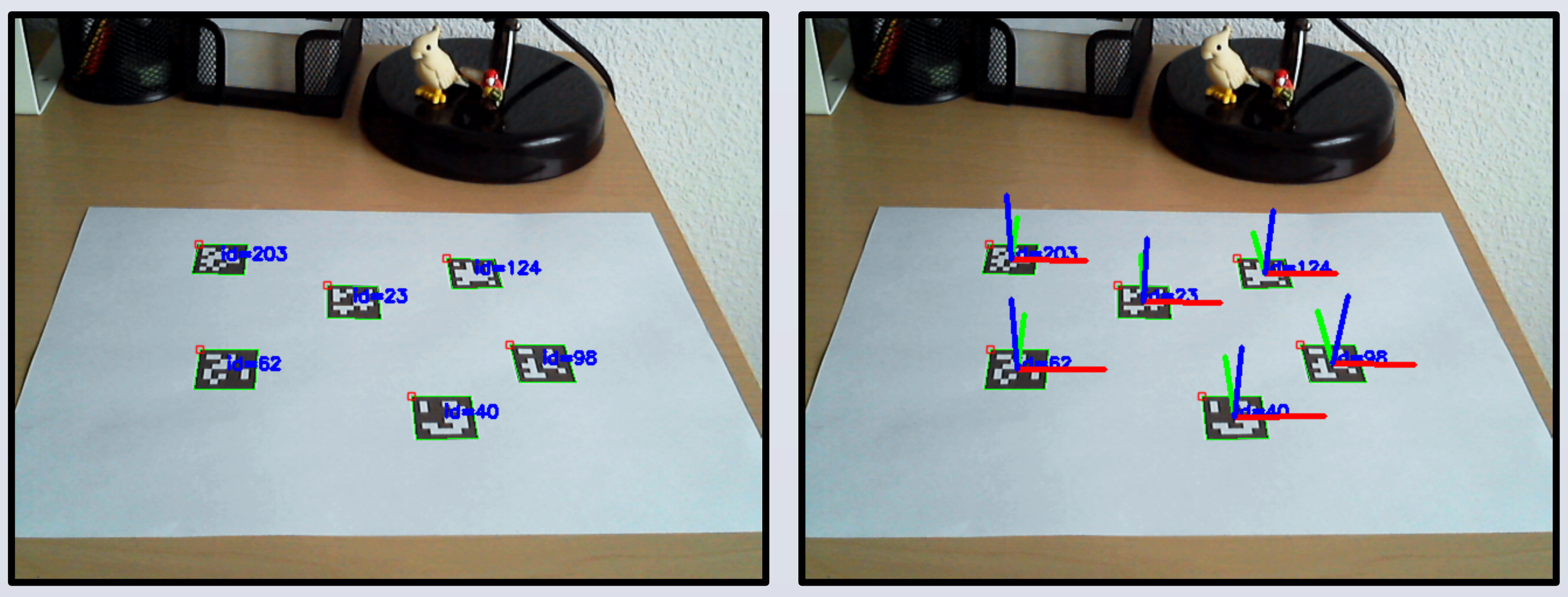


Figure 3: A bird's-eye-view representation of a global positioning camera.

- This figure is a bird's-eye-view representation of a global positioning camera (a webcam in our case), which gives the output of each label's coordinates as a vector $V_n = [x_n, y_n, z_n, \text{theta}_n]$ when prompted by "solvepnp" function. As an open-source computer vision library, OpenCV and detection of ArUco markers have supplied us these coordinate outputs at around 80 ms per detection.
- First, we have placed the markers in the middle of a grid and detected its mid-point coordinate represented by a red dot in the figure above. Then, the collection of these coordinates have supplied us a concrete map of our entire area grid by grid. The midpoint of each grid have been shown by the grey dots above.
- From this point on, all we had to do was a substraction between x_1, y_1, theta_1 of say the first grid and x, y, theta coordinates of the first markes which is very close to the midpoint of the first grid. We intented to substract this error from the input signal of the next step, hence, the next step movement will be for both taking the planned step and compensating the x, y, theta errors done in the previous time step.


Figures 4 and 5: Detection of ArUco Markers using OpenCV.

CONCLUSIONS

Single-agent path-finding and general form of multi-agent path-finding (MAPF) problem was solved in ASP with different approaches and each in terms of their running time efficiency was compared with sample algorithm presented in the paper (Erdem et al. 2013). A simple way to solve dynamic version of MAPF is to recompute solution for MAPF each time change in the environment occurs. Although, this will give a working solution, it requires too much computation power and processing time when there are many agents present in the environment.

We are still trying to come up with more efficient algorithms to solve the main problem and resolving the issues regarding Simulink integration.

REFERENCES

[1] ArucoCode - Google Drive. (n.d.). Retrieved from https://drive.google.com/drive/folders/0B1JICqP8TvqWVkJYeWFMeE9iX00

[2] Agent Pathfinding | www.picswe.com. (n.d.). Retrieved from https://www.picswe.com/pics/agent-pathfinding-0b.html

[3] Create and detect aruco markers. (2017, April 12). Retrieved from https://www.youtube.com/watch?v=y1GF2yscOoc

[4] lego rebot with EV3DEV don't show its ip address on the Screen. · Issue #967 · ev3dev/ev3dev. (n.d.). Retrieved from https://github.com/ev3dev/ev3dev/issues/967

[5] OpenCV: Detection of ArUco Markers. (n.d.). Retrieved from https://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html