A small, incomplete collection of projects I have undertaken over the years

Dr.-Ing (EE) Alexandros Philotheou, M.Sc.

MAR 2024

Contents

0.1 Research Projects—Emphasis on Robotics

0.1.1 RELIEF: Intelligent Repeaters and Robots for Fast, Reliable, Low-Cost RFID Inventorying & Localization

Context: NSRF Research Project, Aristotle University of Thessaloniki, Greece

0.2 Control Projects

0.2.1 Robust Decentralized Control of Cooperative Multi-robot Systems

Context: M.Sc. Degree Project, KTH Royal Institute of Technology, Stockholm, Sweden. `[Code] [Presentation] [Report]`

Problem I. In a given 3D physical workspace there are a number of agents whose motion is described by non-linear continuous-time dynamics, and a number of obstacles. All agents are constrained in (a) moving within the workspace boundaries and (b) avoiding collisions with each other and with the obstacles in the workspace. Additionally, some agents are constrained in keeping certain distance bounds from other agents. Given this setting and these constraints, the goal is for the whole multi-agent system to navigate itself from some initial configuration to a goal configuration and to stay there (be stable there), regardless of disturbances affecting the agents.

One solution to this problem is the design of a robust model predictive control regime for the entire team of cooperating robot systems. In this case there are two schools of thought: control laws are handed down by an all-knowing supreme authority, or calculated by each agent individually, with the incomplete knowledge that each possesses. The decentralised approach is more grounded in reality than the first approach, assumes an additional degree of independence between agents, and makes no further requirements of the overall system. While the problem involves agents whose dynamics are independent of one-another, the decentralised solution given here couples their constraints as the means of capturing the cooperative behaviour required. Figure 1 shows a three-agent system controlled by such a control regime, passing through a narrow gap between two objects before reaching its intended configuration, without additionally violating its prescribed constraints.

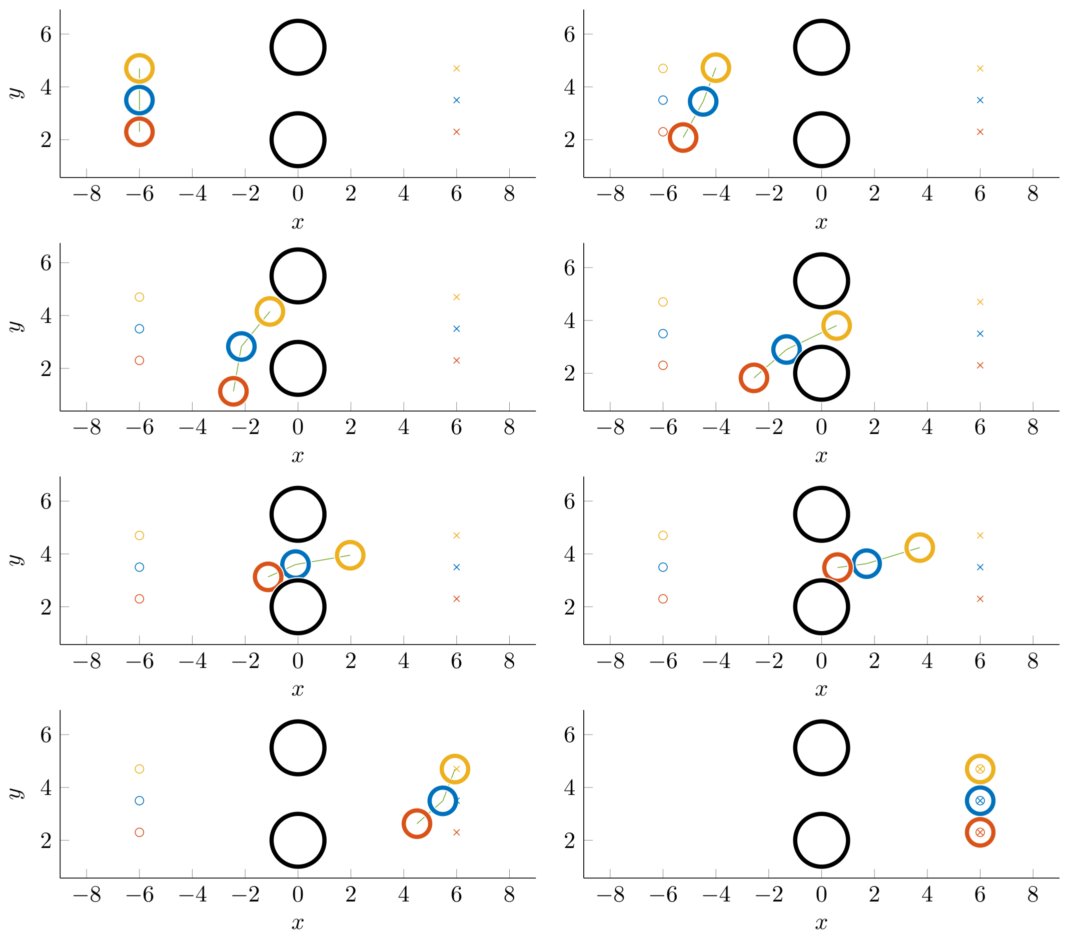

Figure 1: Trajectories of three agents in the $x - y$ plane. A faint green line connects agents who are to stay within certain distance bounds from one-another. Obstacles are marked with black. Mark $\circ$ denotes equilibrium configurations. Mark $\times$ marks desired configurations

Analytical proofs are given to show that under the proposed control regime: (a) Subject to initial feasibility, the optimization solved at each step by each agent will always be feasible, irrespective of whether or not disturbances affect the agents. (b) Each (sub)system can be stabilized to a desired configuration, either asymptotically—when uncertainty is absent—or within a neighbourhood of it when uncertainty is present, thus attenuating the affecting disturbance. In this context disturbances are assumed to be additive and bounded.

Notions/resources/tools used: Predictive Control, ISS stability, MATLAB, git

Publications resulted from this work: [FND18; FND20]

0.2.2 Tracking the circumference of a circle with a RC car

Context: Automatic Control Project Course EL2425, KTH Royal Institute of Technology, Stockholm, Sweden.

`[Code] [Wiki/Resources] [Video 1 | Video 2]`

Problem II. Assume a remotely controlled vehicle of Ackermann steering whose pose (x, y, θ) is measureable in 2D space, equipped with a computing unit. Given a circular path of appropriate radiue, the goal is for the vehicle to navigate the path as closely as possible.

The solution was sought after using Model Predictive Control, with variable reference poses within the horizon of the optimization problem. Figure 2 shows the trajectory of the vehicle from a top view during the transient phase; figure 4 depicts it during the steady-state phase. Accordingly, figures 3 and 5 show the respective errors in each phase as a function of time. These refer to actual experiments, not simulations. Notably, the steady-state error does not exceed 3.5cm, that is, the center of gravity of the vehicle does not diverge more than 3.5cm from the reference circular trajectory.

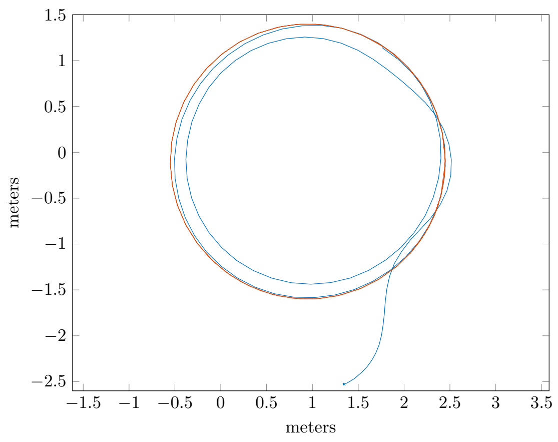

Figure 2: Reference trajectory (red) and trajectory of the vehicle (blue), in the transient phase

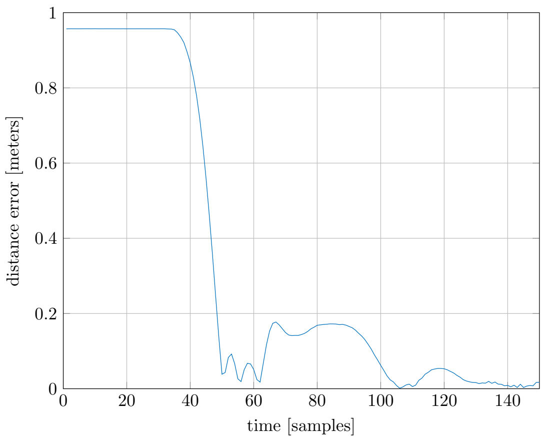

Figure 3: The discrepancy in distance between the trajectory of the vehicle and the reference trajectory in the transient phase, in meters

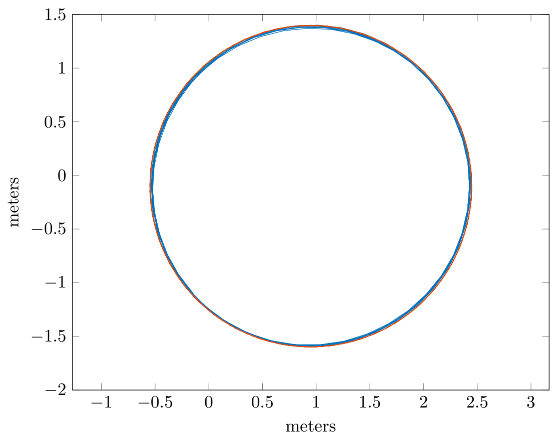

Figure 4: Reference trajectory (red) and trajectory of the vehicle (blue), in steady state

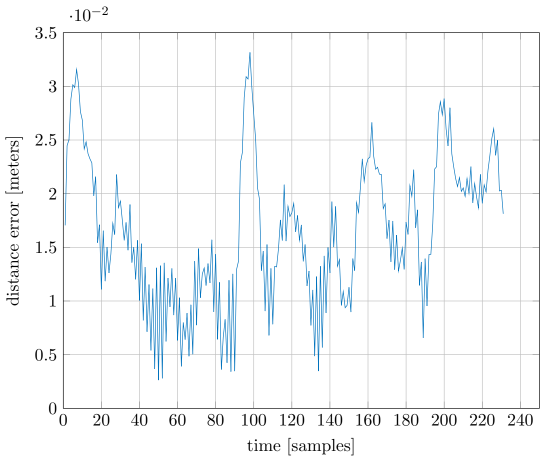

Figure 5: The discrepancy in distance between the trajectory of the vehicle and the reference trajectory in steady state, in meters

Notions/resources/tools involved: Predictive Control, ROS kinetic, Linux, Python, git, MATLAB

0.2.3 Tracking the centerline of a lane with a RC car

Context: Automatic Control Project Course EL2425, KTH Royal Institute of Technology, Stockholm, Sweden.

Problem statement: suppose a remotely controlled vehicle chassis of Ackermann steering, equipped with a laser rangefinder and a computing unit. Given a straight path (something that simulates a road lane, e.g. a corridor), the goal is for the vehicle to navigate the path while always staying in the middle of it. The solution to the problem involves solving two distinct and independent sub-problems, centered around the (a) translational error, and (b) the rotational error, with respect to the middle line of the path. Furthermore, the problem can be approached by two ways: through the design of a PID controller, which is easier to setup, faster in execution, but potentially difficult to tune, and through that of a MPC controller, which requires more work, is slower in execution, but is more robust than the PID controller. Figure 6 shows the evolution of the one-dimensional angular error of the vehicle under control by the PID controller.

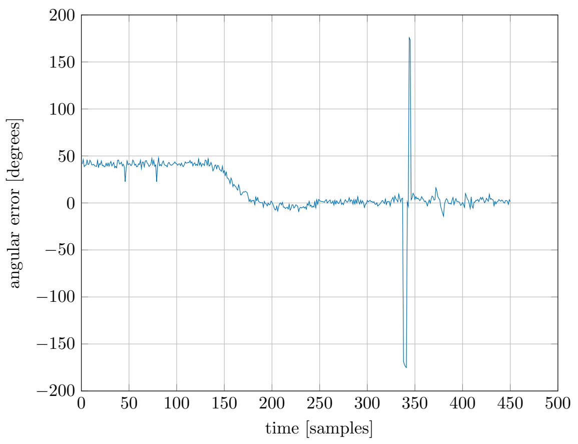

Figure 6: The angular error of the vehicle controlled via PID control in degrees.

Sample results can be found in the following video sequences:

`https://www.youtube.com/watch?v=w3Wnw5SLmss`

`https://youtu.be/937OZez1iN8?t=142`

Notions / resources / tools involved: {PID, Predictive} Control, ROS (Linux), Python, git, MATLAB.

0.2.4 Balancing a segway

Context: Systems and Control in Practice EL2222, KTH Royal Institute of Technology, Stockholm, Sweden.

Problem statement: suppose a two-wheeled motor-led contraption equipped with a computing unit and an IMU. The goal is to balance the system in an upright position using information from the gyro and accelerometer. A basic solution consists of first integrating and fusing the angular velocity and linear acceleration measurements to a filter that estimates the system's angular error with respect to the vertical. The second step employs a (in this case PID) controller that acts in a way that keeps this error at zero. Figures 7 and 8 show the real and idealized "shellfie" segway.

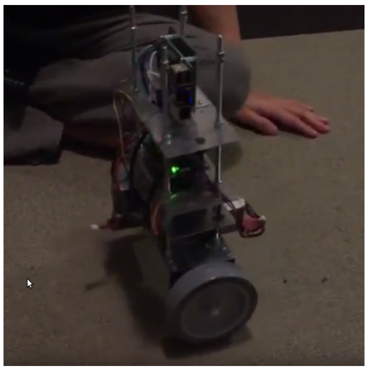
Figure 7: The custom-built "shellfie" segway. Image courtesy of Jatesada Borsub.

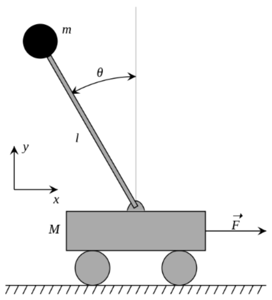

Figure 8: The shellfie is in principle an inverted pendulum.

Notions / resources / tools involved: PID Control, Raspberry Pi, IMU, ROS (Linux), Python

0.3 Estimation Projects

0.3.1 How one detects vehicles violating the speed limit when all one has is a stationary camera

Context: Applied Estimation Project EL2320, KTH Royal Institute of Technology, Stockholm, Sweden.

Problem statement: suppose a stationary camera overseeing a stretch of road on which vehicles travel freely. We would like to identify vehicles travelling at a speed larger than a given threshold.

One solution is comprised of two segments: (a) At first, a differentiation between what is considered background (the unvaried and vehicle-less road scene) and what is considered foreground (the objects “travelling” on top of the background) is in order. Since the data is in video form, we need to track the colour of each pixel through time, and therefore perform segmentation via Gaussian Mixture Models. (b) After vehicles have been detected, a separate Kalman filter (alternatively, a Particle filter) estimates the vector of the velocity of each detected vehicle. Given that the camera is stationary, information (or even guesstimates) about its placement with respect to the road it oversees can establish a correspondence between a vehicle’s real (road) velocity and represented (image) velocity, so that vehicles above the speed limit are identified. Image9 shows a frame drawn from a video sequence where the position and velocity of two vehicles are being tracked.

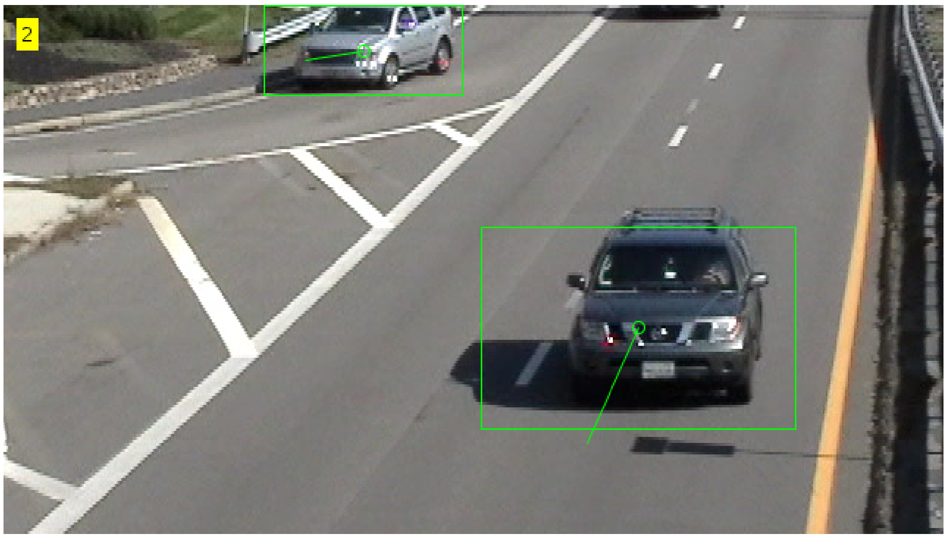

Figure 9: Two vehicles are successfully detected, with their velocity vector being estimated through time, using one Kalman filter per vehicle.

Notions / resources / tools involved: video segmentation, Kalman filter, Particle filter, MATLAB.

0.4 Computer Vision Projects

0.4.1 Hole detection via RGB-D camera within the RoboCup Rescue competition

Context: Voluntary work at PANDORA Robotics group, Aristotle University of Thessaloniki, Thessaloniki, Greece.

`http://pandora.ee.auth.gr/`

`https://github.com/pandora-auth-ros-pkg`

"The RoboCupRescue Robot League is an international league of teams with one objective: Develop and demonstrate advanced robotic capabilities for emergency responders using annual competitions to evaluate, and teaching camps to disseminate, best-in-class robotic solutions."[1]. Within the context of the RoboCup Rescue competition, robotic rescuers have to be able to locate simulated victims in closed spaces, simulating what is to happen during emergency situations. A rescuer has to first detect the "holes" in walls behind which these victims are assumed to be located. The unmanned ground vehicle PANDORA competes in the autonomous class and uses a RGB and a Depth camera for such purposes. Each of the images of the two cameras undergo independent analyses so as to make locating the precise outline of holes more probable. Figure 10 shows an example outcome of such analyses after cross-referencing RGB-derived holes to Depth-derived holes and vice versa, and after further processing. In terms of development and testing time, the resulting ROS package took around seven months to build; in terms of size it is comprised of around 8K lines of code and 4K lines of comments.

[1] `http://wiki.robocup.org/Robot_League`

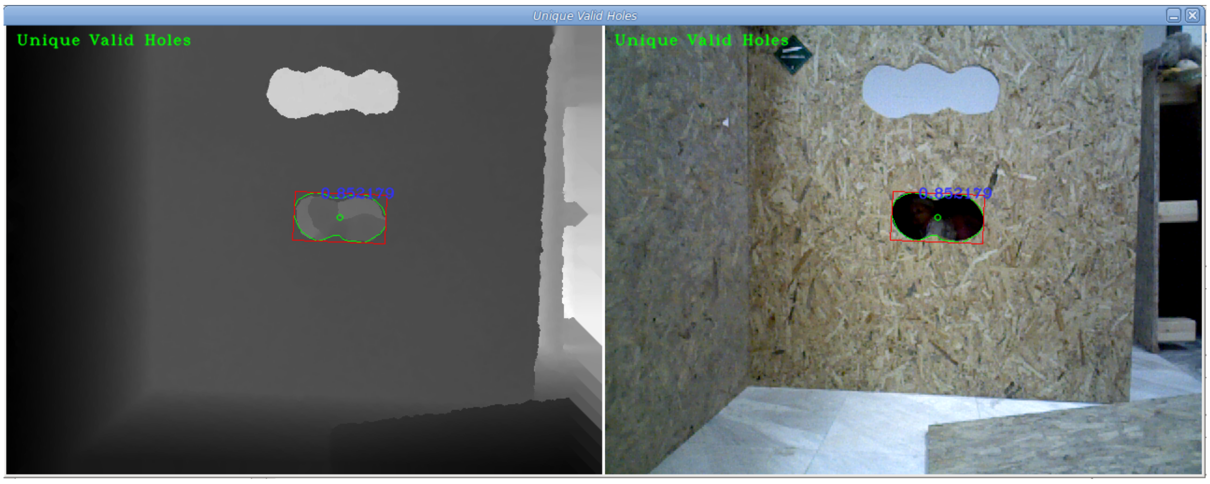

Figure 10: Although the wall has two holes, only the one closer to the ground is valid: the hole above it is through-and-through and therefore cannot contain possible victims.

Resources / tools involved: C++, ROS, git(hub)

0.5 Machine Learning Projects

0.5.1 Multi-label classification using Learning Classifier Systems

Context: Diploma Thesis, Aristotle University of Thessaloniki, Thessaloniki, Greece.

Problem statement: suppose a set of annotated data of multiple labels (not multiple classes: each instance is simultaneously classified to more than one labels). Given this set, we would like to build a classifier which, trained with this set, can classify unseen data to multiple labels with the highest accuracy possible.

The solution was sought after through extending the framework of Learning Classifier Systems to the multi-label dimension. LCS's are systems that represent data instances with rules of the form "antecedents ⇒ consequents". Within this frame, rules are treated as a population evolving through time according to the principles of evolution mapped out by Charles Darwin. Rules undergo (a) crossover with other rules depending on a measure of their fitness (their "quality"), and (b) mutations indiscriminately during crossover, through time, iteratively.

The resulting classifier, called GMl-ASLCS, was compared against the state-of-the-art classifiers on a number of datasets and its performance was found to be (a) overall the highest in terms of accuracy, and (b) not statistically significantly divergent than those it was compared against. Table 1 shows the accuracy of the designed LCS called GMl-ASLCS and that of its rivals on 6 standard multi-label datasets.

Algorithms / Datasets	music	yeast	genbase	scene	medical	enron	Rank
GMl-ASLCS	**60.47**^{1}	**51.67**^{1}	**98.63**^{1}	63.15^{2}	51.58^{3}	40.35^{2}	**1.67**^{1}
GMl-ASLCS $_0$	50.05^{4}	45.51^{4}	87.90^{5}	42.39^{5}	40.14^{5}	39.40^{3}	4.33^{5}
BR-J48	46.23^{5}	43.95^{5}	98.62$^{2.5}$	51.34^{4}	**74.26**^{1}	36.71^{4}	3.58^{4}
RA*k*EL-J48	50.91^{3}	48.74^{3}	98.62$^{2.5}$	57.76^{3}	72.84^{2}	**41.04**^{1}	2.42^{2}
Ml*k*NN	53.26^{2}	51.62^{2}	94.11^{4}	**66.14**^{1}	41.77^{4}	31.84^{5}	3.00^{3}

Table 1: Comparison between GMl-ASLCS and rival multi-label classifier algorigthms. The exponents refer to the rank of each algorithm on each specific dataset. Column "Rank" summarises the overall rank of each classifier.

Notions / resources / tools: Genetic algorithms, Java, git.

0.6 Other

0.6.1 Play Simon with a Raspberry Pi

Context: Systems and Control in Practice EL2222 Project, KTH Royal Institute of Technology, Stockholm, Sweden.

The game “Simon” is used as a measure of the ability to retain memories, and in particular, memories of events and their sequence. The simplicity of the game can be captured by the equal simplicity with which the game as an artefact can be set up using a Raspberry pi, a small breadboard, four or more leds and some cables.

[FND18] Alexandros Filotheou, Alexandros Nikou, and Dimos V. Dimarogonas. “Decentralized Control of Uncertain Multi-Agent Systems with Connectivity Maintenance and Collision Avoidance”. In: *2018 European Control Conference (ECC)*. IEEE, June 2018, pp. 8–13. ISBN: 978-3-9524-2698-2. DOI: `10.23919/ECC.2018.8550343`. URL: `https://ieeexplore.ieee.org/document/8550343/`.

[FND20] Alexandros Filotheou, Alexandros Nikou, and Dimos V. Dimarogonas. “Robust decentralised navigation of multi-agent systems with collision avoidance and connectivity maintenance using model predictive controllers”. In: *International Journal of Control* 93.6 (June 2020), pp. 1470–1484. ISSN: 0020-7179. DOI: `10.1080/00207179.2018.1514129`. URL: `https://www.tandfonline.com/doi/full/10.1080/00207179.2018.1514129`.