Obstacle Avoidance onboard MAVs using a FMCW RADAR

Master of Science Thesis

Nikhil Wessendorp

Delft University of Technology
12 April 2021

TU Delft

Obstacle Avoidance onboard MAVs using a FMCW RADAR

Master of Science Thesis

by

Nikhil Wessendorp

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday April 23, 2021 at 09:00.

Student number:	4432290	
Project duration:	March 18, 2020 – April 23, 2021	
Thesis committee:	dr. J.J.G. Dupeyroux,	TU Delft, Supervisor
	Prof. dr. ir. G.C.H.E. (Guido) de Croon,	TU Delft, Supervisor
	dr. F. Fioranelli,	TU Delft, TT Assistant Professor

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Acknowledgements

I would like to thank my supervisors Julien Dupeyroux and Dr. Guido de Croon. Julien has been a tremendous help guiding me throughout the whole project, to solve all sorts of problems and keeping motivation up, and Guido thank you for sharing your knowledge and wisdom. Further thanks to Nilay Sheth and Raoul Dinaux for their collaboration with my thesis. Lastly, I would like to thank my family and friends, especially Giammy, Val, and Lex for their motivational support.

Contents

List of Figures

Introduction

Micro air vehicles (MAVs) are increasingly being considered for aerial tasks such as delivery of goods and surveillance due to their lightweight, compact design and manoeuvrability. To safely and reliably carry out these tasks and navigate to its objective, especially in complex and cluttered environments, the MAV is also required to sense and avoid (S&A) obstacles. Due to the MAVs limitations in weight, power and processing power, vision systems usually prove ideal for sensing the environment, being a cheap, lightweight, power efficient and a rich source of information. They do however require adequate computational resources and most importantly, good visibility. When the environment does not host these conditions, for instance when flying though dust, smoke or fog, other sensors need to be utilised that can provide more robust sensing to ensure safe and reliable operation.

Radar sensors are mostly unaffected by atmospheric conditions and have been used extensively in the aerospace industry for this purpose. These sensors were traditionally heavy and power hungry, only applicable on ground or in large craft. However other radar sensors have since come about that are more suited for use in small MAVs. Specifically, lightweight, power efficient and compact frequency modulated continuous wave (FMCW) radars have increasingly been used in advanced driver assistance systems as auxiliary sensors, however there has been little work to integrate them on MAVs. This sensor provides the range, horizontal bearing and radial velocity (Doppler shift) of any objects in the field of view, which can then be used for multi-target tracking (MTT) [38]. The major disadvantage of the sensor is the limited field of view (approximately 80 degrees horizontal) and noisy nature of the sensor, especially in cluttered environments.

The challenge is to explore filtering, tracking and avoidance algorithm pipelines to extract meaningful information from the raw data and investigate the sensor's effectiveness with respect to obstacle avoidance on MAVs. This will include algorithms such as data association, estimation and avoidance, as well as an investigation of neural networks to aid in processing the raw data and provide some filtering. This will be accomplished by integrating the sensor on a MAV and testing and tuning the algorithms both in real life (in the cyberzoo flying arena of the aerospace faculty), and using data gathered as part of an obstacle detection and avoidance dataset that was generated during this project. This will hopefully allow MAVs to operate safer, either using a standalone radar or integrated with other sensors.

This report is the final document of a master thesis. The first part is a scientific paper, outlining the related work, radar sensor, development and results of the study, and the second part is the preliminary report containing the literature study and initial research objectives and questions. Note that the preliminary report has already been graded as part of the course AE4020.

I

Scientific Paper

Obstacle Avoidance onboard MAVs using a FMCW RADAR

Nikhil Wessendorp, Raoul Dinaux, Julien Dupeyroux and Guido C. H. E. de Croon*

***Abstract*— Micro Air Vehicles (MAVs) are increasingly being used for complex or hazardous tasks in enclosed and cluttered environments such as surveillance or search and rescue. With this comes the necessity for sensors that can operate in poor visibility conditions to facilitate with navigation and avoidance of objects or people. Radar sensors in particular can provide more robust sensing of the environment when traditional sensors such as cameras fail in the presence of dust, fog or smoke. While extensively used in autonomous driving, miniature FMCW radars on MAVs have been relatively unexplored. This study aims to investigate to what extent this sensor is of use in these environments by employing traditional signal processing such as multi-target tracking and velocity obstacles. The viability of the solution is evaluated with an implementation on board a MAV by running trial tests in an indoor environment containing obstacles and by comparison with a human pilot, demonstrating the potential for the sensor to provide a more robust sense and avoid function in fully autonomous MAVs.**

I. Introduction

Micro Air Vehicles (MAVs) are very well suited for navigation in complex environments such as indoor buildings as a result of their lightweight, compact design and manoeuvrability, making them ideal for tasks such as search and rescue in hazardous environments and surveillance. To ensure safe flight in such environments, the MAV is usually required to reach a destination while also sensing and avoiding (S&A) obstacles or people. Apart from dealing with cluttered, GPS-denied environmental conditions, closed tight spaces and limited visibility, MAVs are also constrained by computational, power and weight limitations. For these reasons, the use of cheap, lightweight, and passive vision systems are among the most popular methods. Although cameras are a rich source of information, they also demand an adequate amount of computational power and sufficient visibility conditions. In the absence of these requirements, for example when providing aid and assistance in a smoke-filled building, other sensors need to be considered for a more robust solution to guarantee operation and safety.

In low-light conditions, event-based cameras, laser-based sensors and illumination can compensate for the deficit left by ordinary cameras. However, these systems quickly break down in the presence of dust, fog or smoke. To combat this, ultrasound (sonar) sensors or radar sensors can be utilised instead. Ultrasound sensors are however point-based and have difficulty sensing soft or curved edges at large incidence angles [1]. Radar sensors on the other hand have been used extensively in the last century for object tracking in the aerospace industry. However these sensors have traditionally been expensive, complex, heavy and power hungry. Only recently have all these factors been improved upon to produce cheap, lightweight sensors that are typically used as auxiliary sensors for applications such as advanced driver assistance systems and ground based applications. These new sensors come in the form of compact millimetre-wave (MMW) frequency-modulated continuous-wave (FMCW) radars. These radars provide the range, bearing and radial velocity of detections [2], which can be used for the purpose of multi-target tracking (MTT) and ultimately avoidance. Usage of the radar sensor on small MAVs for indoor obstacle avoidance has not properly been established yet, and the rare existing work does not allow for proper bench-marking.

This study will attempt to fill this gap. While physically ideal for use on small MAVs, the major drawback lies with the fact that the sensor is noisy, and thus requires fine-tuning and filtering to extract meaningful data and perform S&A functions reliably. The challenge that this paper addresses is to implement such a filtering and tracking pipeline appropriate for real-time use on a MAV. We evaluate the avoidance capabilities in a flight arena equipped with the OptiTrack motion tracking system, and test is on a dataset[1] acquired with ground truth positions in the same arena, containing obstacle avoidance trials.

Section II brings into view the existing work done on FMCW radars and integration onto Unmanned Aerial Vehicles (UAVs). Section III will explain the processing pipeline and avoidance algorithms used, followed by Section IV giving the results and performance of the implementation.

II. Related Work

The underlying technology and processing pipeline for radars are well explored. While radar sensors are preferably not used independently due to their lack of fidelity, a lot of research has been done to fuse target information with vision systems. Long et al. 2019 [3] for instance develop a system to aid the visually impaired, which utilises a particle filter to both fuse information and track objects using FMCW radar, a normal camera (using a convolutional neural net) and a stereoscopic IR camera setup, combining the advantages of vision (object classification and identification) with the accurate range, bearing and radial velocity measurements provided by the FMCW radar. Kim et al. 2014 [4] and Ćesić et al. 2016 [5] also fuse vision and FMCW radar for the

*All authors are with Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, 2629HS Delft, The Netherlands. j.j.g.dupeyroux@tudelft.nl

[1]https://github.com/tudelft/ODA_Dataset

purpose of MTT by means of an association algorithm (joint probabilistic data association filter - JPDAF) in combination with a Kalman filter (KF). Both methods perform similar functions, however a particle filter takes a fully probabilistic Bayesian approach, while still demonstrating computational tractability and convergence [6].

With regards to airborne applications, little has been explored concerning close proximity obstacle avoidance, although some studies address the use of a (often heavier and more complex) radar as a complimentary sensor to the Traffic Collision and Avoidance System (TCAS) and Flight Alarm (FLARM) for integrating UAVs into the local airspace [7]–[9]. Eric et al. 2013 [10] achieve this by only utilising a stand-alone FMCW radar sensor using beamforming for an increased field of view (FOV), however outdoor domain of this application is quite different in nature from an indoor environment. Scannapieco et al. 2015 [11] instead use a gimbaled 94GHz FMCW radar for mapping an indoor environment using Interferometric Synthetic Aperture Radar, and subsequently using that for path planning, however requiring significantly more time and processing power. Scannapieco et al. 2015 [12] evaluate the FMCW radar sensor itself rather than implementation for obstacle avoidance of MAVs, only performing outdoor ground tests, and Yu et al. 2020 [13] fuse information from a camera and FMCW radar for obstacle avoidance, however are still very reliant on the vision system.

III. Signal Processing

A. Detection

Specifically, the sensor in use is a fast-chirp FMCW radar and operates by transmitting a saw-tooth FM carrier wave from its transmitter antenna and listening for the returns reflected by objects in its two receiver antennas (one period is referred to as a 'chirp'). The received signals (one per object) are shifted to the right (a delay in time) with increasing range, as shown in Figure 1. The transmitted and reflected signals are then mixed to produce the intermediate frequency (IF) signal (or beat frequency) and is passed through a low pass filter followed by an ADC to produce the raw data of the radar, which consists of the (I, Q) values representing the electromagnetic wave. Fast-chirp FMCW radars feature reduced range but better resolution compared with the traditional FMCW radars, where the chirp duration is one order magnitude longer. Note that because of this fast-chirp nature, the Doppler shift in frequency is negligible compared to the shift in frequency due to range, and is not accounted for in this step.

Following from this raw data, the IF signal is passed through a fast-Fourier transform (FFT) (with zero padding), which highlights the peaks representing range produced by all objects in the FOV. Once a threshold is applied, the detections are distinguished with their associated range using Equation (1):

$$R = \frac{cT_c f_b}{2B} = \frac{cf_b}{2S} \tag{1}$$

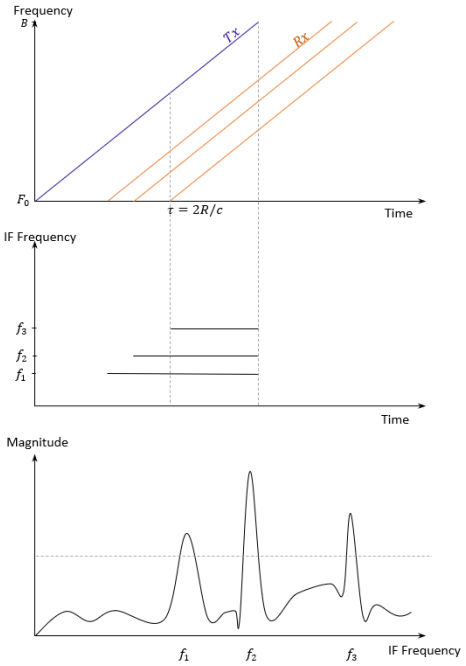


Fig. 1. The basic principle behind FMCW radar. Top: the transmitted (purple) and received (orange) waves of one chirp. Middle: the mixed IF signal, showing the distinct frequencies that different objects produce. Bottom: the first range FFT (only the magnitude shown) applied to the IF signal, delineating the peaks.

where c is the speed of light, B is the bandwidth, f_b is the beat frequency, S is the slope of the frequency modulated ramp and T_c is the up-chirp time. To determine the horizontal bearing of the detections, the phases of the two antennas in the FFT (where only magnitude is shown in Figure 1) are compared with the use of Equation (2):

$$\theta = \arcsin\left(\frac{\lambda \Delta\omega_d}{2\pi d}\right) = \arcsin\left(\frac{\Delta\omega_d}{\pi}\right) \tag{2}$$

where d is the antenna spacing, λ is the wavelength and $\Delta\omega_d$ is the phase difference between the two antennas. Note that $d = 0.5\lambda$ gives the largest FOV of $\pm 90°$.

The radial velocity of the detected objects can also be extracted by taking a second set of FFTs over multiple chirps: this essentially compares the change of phase over 2 consecutive chirps, since the phase is very sensitive to small changes in distance (essentially, a phase change is a Doppler frequency shift). This change in phase $\Delta\omega$ is given by Equation (3), where V is the radial velocity of the target and λ is the wavelength.

$$\Delta\omega = \frac{4\pi V T_c}{\lambda} \tag{3}$$

By taking a number M of IF samples instead of 2 (number of chirps in a frame), a velocity estimate can be computed for each individual object by taking a Doppler FFT of $\Delta\omega$ over the different chirps, creating a 2D FFT matrix shown in Figure 2. Here objects can be resolved by both their range and radial velocity.

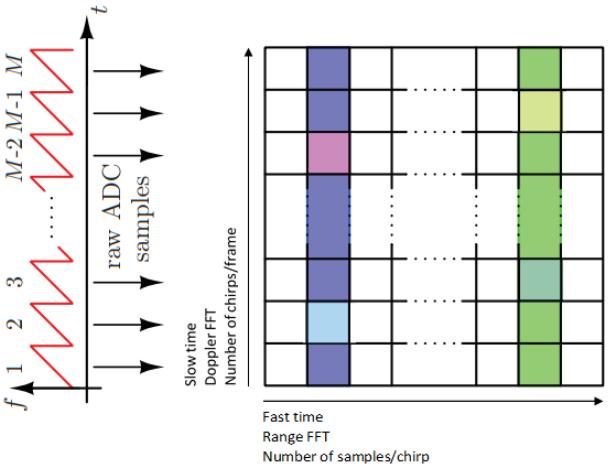


Fig. 2. A 2D FFT, where the horizontal axis represents fast time (one chirp) and the vertical axis represents slow time (one frame). After performing FFTs on the fast time axis identifying any peaks (shaded regions), an FFT is performed along the slow time axis to determine the phase change (Doppler shift) of same range bin can be compared to distinguish one or multiple peaks (objects with different velocities at same range, coloured squares). Adapted and modified from [2].

B. Filtering

As the sensor is rather noisy, both data association and tracking have to be employed. Data association involves handling the detections and objects that are being tracked, that is, first: assigning detections to existing objects and discarding detections from clutter, second: creating new objects when detections indicate there is a new object in the FOV, and third: deleting objects when they leave the FOV (when there are no new detections). These operations are done in conjunction with the Kalman filtering (KF) process.

The first process boils down to calculating a cost matrix which indicates the cost of associating a detection to an object or clutter. For this, a simple global nearest neighbourhood (GNN) optimisation algorithm is used [14], which defines the cost as being proportional to the square of the distance between the detection and prediction of the object position (using the KF). The cost matrix also contains the cost associated with misdetecting an object, that is, no detections associated with the object (right side of Equation (4) which shows the cost matrix L). Additionally, gating is used, whereby any detections that are greater than a threshold distance to a particular object are immediately discarded ($-\ell^{n,m} = \infty$). The GNN algorithm is a greedy yet computationally efficient approach that works well for simple scenarios. This cost matrix is then converted to an assignment matrix by minimising the cost using an algorithm such as the Hungarian algorithm [15].

$$L = \left[\begin{array}{ccc|ccc} -\ell^{1,1} & \dots & -\ell^{1,m} & -\ell^{1,0} & \dots & \infty \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -\ell^{n,1} & \dots & -\ell^{n,m} & \infty & \dots & -\ell^{n,0} \end{array}\right] \quad (4)$$

$$\ell^{i,0,h} = \log\left(1 - P^D\right) \quad (5)$$

$$\ell^{i,j,h} = -\frac{1}{2}\left(z^i - \hat{z}^{i,h}\right)^\top \left(S^{i,h}\right)^{-1}\left(z^j - \hat{z}^{i,h}\right) \quad (6)$$

where n is the number of objects being tracked, m is the number of radar detections, $-\ell^{n,m}$ represents the association cost, and $-\ell^{n,0}$ represents the cost of misdetecting the object. P_d is the probability of detection, $z^i - \hat{z}^{i,h}$ is the distance between the measurement and predicted location of the object, and $S^{i,h}$ is the innovation covariance of the KF. The second step is done by keeping track of all detections within the FOV (associating them to new candidate tracks which are also tracked with a KF). When the covariance of the position (in the $P^{i,h}$ matrix of the KF) drops below a threshold, the object is initiated (track birth) and considered valid. Likewise, when a tracked object's covariance rises above another threshold (when it is misdetected multiple times) it is removed (track death).

Once a detection has been associated with an object, the detection is used as the measurement input (range, bearing, radial velocity) to an ordinary KF that is run for every object to filter out noise and estimate the tangential velocity as well, thereby obtaining the range, bearing and their derivatives. The KF assumes both observation noise, to account for the sensor noise, and process noise, to account for any non-linearities in the motion of the objects or MAV, as a constant acceleration model is assumed.

C. Avoidance

The obstacle avoidance control method used is Velocity Obstacles (VO), which finds the set of velocity vectors of the MAV that will result in a collision with the object, taking into account the radius of both the object and MAV.

The following explanation is retrieved from Fiorini et al. 1998 [16]. Consider a robot A and an obstacle B with velocities V_A and V_B and radii r_A and r_B, as shown in Figure 3. Mapping B onto the configuration space of A means enlarging object B by the radius of A to form object $\hat{B}$, and reducing A to a point $\hat{A}$ and computing the relative velocity of robot A with respect to object B, $V_{A,B} = V_A - V_B$. The collision cone $CC_{A,B}$ can then be formed, in which any *relative* velocity $V_{A,B}$ will result in a collision with object B. The radar sensor will yield the relative position and $V_{A,B}$. Since the ego-velocity V_A is known, V_B can also be determined.

By accounting for the velocity of robot A and its limitations in maximum velocity and direction change, a desired $\hat{V}_{A,B}$ can be computed by adjusting $V_{A,B}$ to lie on one of the edges of $CC_{A,B}$ (also taking into account any safety margins). An absolute desired velocity of robot A, $\hat{V}_A$, can then be found by addition with V_B. In the case in Figure 3, it is most beneficial to slow down and adjust the velocity vector to the right.

D. Sensor Characteristics

The sensor used in this study is the Infineon XENSIV^{TM} 24GHz Position2Go kit, a small 10g fast-chirp FMCW radar that features human target detection at a range of 1-12m and a horizontal-vertical half-power beamwidth (HPBW) FOV

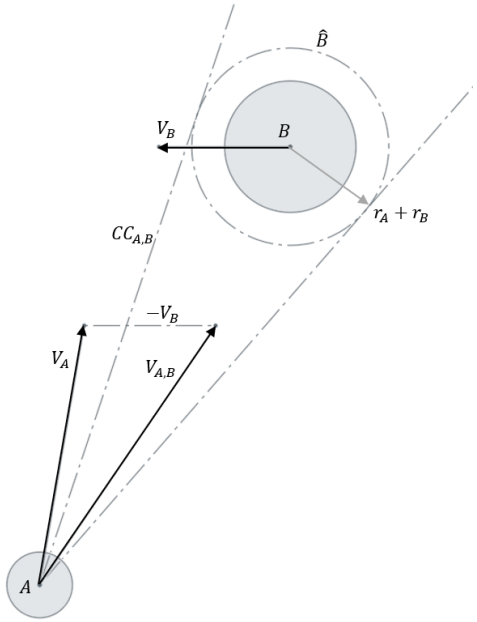


Fig. 3. Robot A and moving obstacle B will collide as $V_{A,B}$ lies within $CC_{A,B}$, which is formed by enlarging object with the radius of robot.

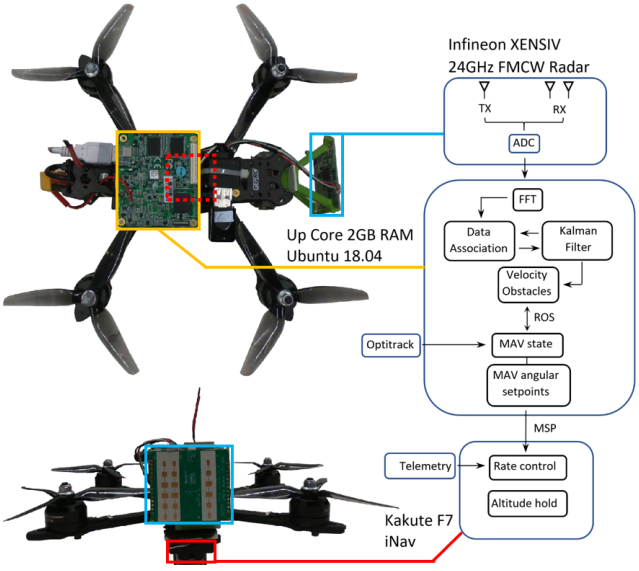


Fig. 4. Top and front view of the MAV. Light blue indicates the FMCW radar, yellow the Up Core companion computer, and red the flight controller (underneath the companion computer) and the LiDAR altimeter.

of 76°x19°. Although this is sufficient for frontal obstacle avoidance, objects that are moving faster than the MAV outside the FOV still pose a collision threat, albeit less likely. With a maximum bandwidth of 200MHz, it is able to resolve objects $0.75m$ apart in range, with a range accuracy of $\pm$ 15cm and an angular accuracy of $\pm 2°$ from $0-20°$, and up to $\pm 8°$ from $20-65°$. Strict filtering of clutter detections is required due to the noisy nature of the sensor. Furthermore, as the sensor only features 2 receiver antennas, and is thus only able to detect 2 objects at a time in the same range bin. However taking this into account in the association algorithm, all objects in the FOV can be detected over multiple frames (although decreasing the update frequency).

IV. Performance Evaluation

A. Implementation

The FMCW radar sensor was integrated and tested on a custom made 5-inch MAV, as shown in Figure 4. Two processing boards are integrated. The first is the Kakute F7 flight controller running iNav 2.6.0 firmware, the second companion computer is the Intel Up Core (1.44GHz 64bit processor with 2GB RAM) running Ubuntu 18.04 LTS. The latter runs the radar driver, processes the raw data, performs the MTT and runs a custom made autopilot using ROS (Robot Operating System) to communicate with the radar sensor, computing the avoidance manoeuvre and relaying the desired orientation of the MAV (pitch, roll and yaw angles) to the flight controller (using MSP protocol), which in turn handles the low level rate and altitude control using the TFMini LiDAR rangefinder facing down. The radar sensor is fixed to the front of the MAV at a slight upward tilt of 10° to reduce reflections from the ground.

Testing was done in the flying arena of the TU Delft, equipped with the OptiTrack motion capture system for positioning, which is relayed through UDP to the UP Core, although concerning the avoidance algorithm, only velocity control was implemented. Furthermore, the avoidance algorithm only considers the nearest obstacle, both for simplicity and to stimulate different avoidance scenarios. A simpler avoidance manoeuvre was implemented whereby the MAV simply translates approximately 1m to the side to better approximate the flying behaviour displayed in the dataset in which a human is flying to avoid 1 or 2 obstacles in the flying arena. The obstacles are cardboard poles roughly 0.5m in diameter placed in the centre of the flying arena, and avoidance was carried out from all sides and corners.

B. Results

Looking at Figure 5 and Figure 6, showing the trajectories taken when the MAV is controlled by a human pilot versus the on board obstacle avoidance controller using the radar, it is evident that the FMCW radar can reliably detect obstacles and determine when a collision is imminent, thus allowing the MAV to safely avoid damage or injury to the MAV or environment. On occasion the radar will struggle to track the further obstacle due to the inherent noise of the sensor, however when brought close enough to the obstacle the MAV was still able to perform a successful avoidance manoeuvre. This can best be visualised in Figure 7, which on the left shows the ground truth trajectory and location of the MAV and obstacles, and on the right the output of the filtering and tracking algorithms, showing the relative paths taken by the obstacles (Doppler information is not displayed). First obstacle 1 comes into view (orange ground truth and red detections), which the MAV avoids by moving to the left (or the obstacle moving to the right relative to the

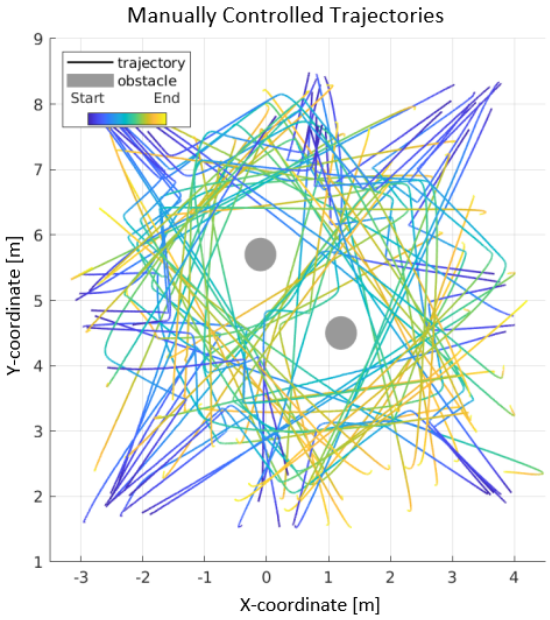


Fig. 5. Manually flown trajectories of the MAV of 78 samples from the obstacle avoidance dataset, avoiding 2 obstacles from different angles.

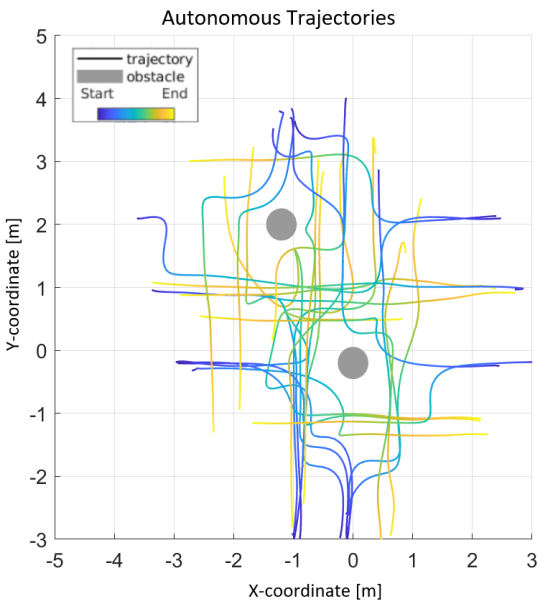


Fig. 6. 26 sample trajectories of the autonomously controlled MAV using the radar sensor.

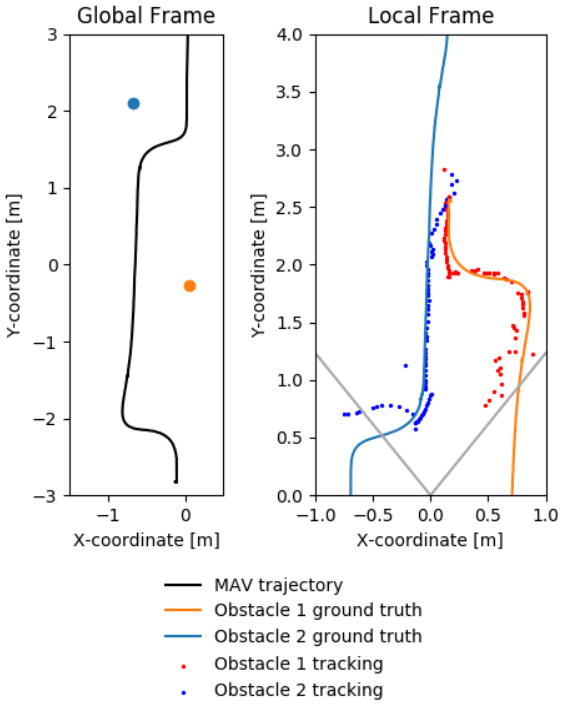


Fig. 7. Illustration of what the FMCW radar sensor detects after filtering and tracking (right) when following the sample trajectory on the left. The grey lines indicate the HPBW FOV of the radar (78°) and the ground truth position is shown in both figures (orange and light blue paths)

MAV), followed by the second obstacle coming into view approximately 0.7 seconds later (light blue ground truth and dark blue detections), which the MAV avoids to the right.

As can be seen in Figure 7 on the right, the error in tracking is most evident when the MAV changes trajectory, which in fact represents a non-linearity in the motion of the obstacle (or MAV) meaning it can take some steps before the ordinary KF is able to cope with this. However when the radar sensor would come to a complete halt, an increase in noise around the obstacle was also observed for approximately one second, further exacerbating the error. This however did not impact the MAVs ability to sense and avoid a collision. Additionally, the error in bearing and range increases as the obstacles move towards the edge of the HPBW FOV, which can be seen in Figure 8, showing an approximately linear trend.

V. Conclusion

This work demonstrates the pertinence of using a standalone FMCW radar sensor for the purpose of sense and avoid. A multi-target tracking and avoidance algorithm have been implemented on a MAV and tested on both one and two obstacles, showing that the MAV is successfully able to avoid them when solely relying on the radar sensor, demonstrating that reliance on this sensor can be effective when required, especially when other sensors fail due to the presence of fog, smoke or flames. This will ultimately help make MAVs for applications such as surveillance and search and rescue safer and more reliable.

To better detect obstacles in cramped spaces, 77GHz FMCW radars can be used, which feature improved bandwidth and resolution, allowing for more accurate detection of obstacles and perhaps classification of walls as well, however requiring a more robust and computationally expensive data association algorithm capable of clustering detections (e.g. DBSCAN [17]). Other FMCW radar sensors also incorporate more than two receiver antennas (allowing for more detec-

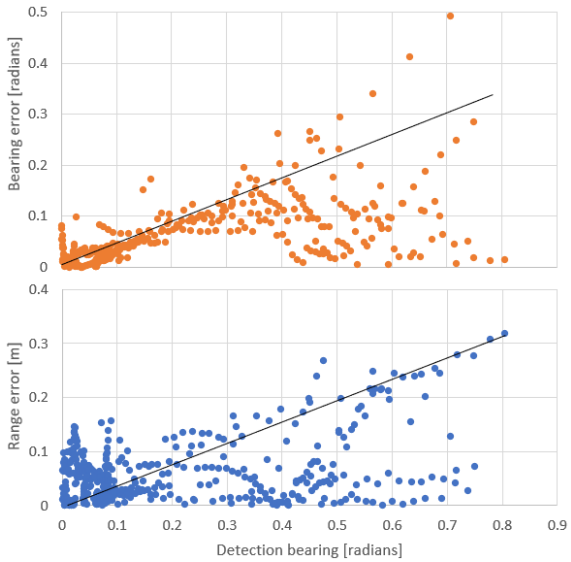


Fig. 8. Data from 4 trials showing the approximately linear trend of the error in bearing and range as the object moves further from the centre of the radar.

tions per scan) or beamforming (scanning a larger FOV). While this study has demonstrated that sense and avoid using a standalone radar sensor can be very useful, it is best used when fused with other sensors when circumstances and conditions allow (even with event-based cameras which can operate in low-light environments, as shown by Zhang et al. 2019 [18] who fuse the sensors in an EKF to compensate for the error bounds produced by both sensors.

Supplementary Materials

The ROS implementation our radar-based navigation system can be found here: `https://github.com/tudelft/radar_nav`, along with supporting videos. The Obstacle Detection and Avoidance dataset is available at: `https://github.com/tudelft/ODA_Dataset`.

Acknowledgements

This work is part of the Comp4Drones project and has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No. 826610. The JU receives support from the European Union's Horizon 2020 research and innovation program and Spain, Austria, Belgium, Czech Republic, France, Italy, Latvia, Netherlands.

References

[1] J. Borenstein and Y. Koren, "Obstacle avoidance with ultrasonic sensors," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 213–218, 1988.

[2] V. Milovanovic, "On fundamental operating principles and range-doppler estimation in monolithic frequency-modulated continuous-wave radar sensors," *Facta universitatis - series: Electronics and Energetics*, vol. 31, pp. 547–570, 2018.

[3] N. Long, K. Wang, R. Cheng, W. Hu, and K. Yang, "Unifying obstacle detection, recognition, and fusion based on millimeter wave radar and rgb-depth sensors for the visually impaired," *Review of Scientific Instruments*, vol. 90, no. 4, p. 044102, 2019. [Online]. Available: https://doi.org/10.1063/1.5093279

[4] D. Y. Kim and M. Jeon, "Data fusion of radar and image measurements for multi-object tracking via kalman filtering," *Information Sciences*, vol. 278, pp. 641–652, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025514003715

[5] J. Ćesić, I. Marković, I. Cvišić, and I. Petrović, "Radar and stereo vision fusion for multitarget tracking on the special euclidean group," *Robotics and Autonomous Systems*, vol. 83, pp. 338–348, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889015303286

[6] C. M. Kreucher, K. D. Kastella, and A. O. Hero III, "Tracking multiple targets using a particle filter representation of the joint multitarget probability density," in *Signal and Data Processing of Small Targets 2003*, vol. 5204. International Society for Optics and Photonics, Conference Proceedings, pp. 258–269.

[7] A. M. Allistair, J. R. Matthew, K. Michail, and P. V. Kimon, "Uav-borne x-band radar for mav collision avoidance," in *Proc.SPIE*, vol. 8045, Conference Proceedings. [Online]. Available: https://doi.org/10.1117/12.884150

[8] Y. K. Kwag and C. H. Chung, "Uav based collision avoidance radar sensor," in *2007 IEEE International Geoscience and Remote Sensing Symposium*, Conference Proceedings, pp. 639–642.

[9] S. Kemkemian, M. Nouvel-Fiani, P. Cornic, P. L. Bihan, and P. Garrec, "Radar systems for "sense and avoid" on uav," in *2009 International Radar Conference "Surveillance for a Safer World" (RADAR 2009)*, Conference Proceedings, pp. 1–6.

[10] I. Eric, W. Jean-Philippe, M. Sébastien, O. Matern, and H. Albert, "Fmcw radar for the sense function of sense and avoid systems onboard uavs," in *Proc.SPIE*, vol. 8899, Conference Proceedings. [Online]. Available: https://doi.org/10.1117/12.2028518

[11] A. F. Scannapieco, A. Renga, and A. Moccia, "Compact millimeter wave fmcw insar for uas indoor navigation," in *2015 IEEE Metrology for Aerospace (MetroAeroSpace)*, Conference Proceedings, pp. 551–556.

[12] A. F. Scannapieco, A. Renga, G. Fasano, and A. Moccia, "Ultra-light radar sensor for autonomous operations by micro-uas," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, Conference Proceedings, pp. 727–735.

[13] H. Yu, F. Zhang, P. Huang, C. Wang, and L. Yuanhao, "Autonomous obstacle avoidance for uav based on fusion of radar and monocular camera."

[14] I. J. Cox, "A review of statistical data association techniques for motion correspondence," *International Journal of Computer Vision*, vol. 10, no. 1, pp. 53–66, 1993.

[15] B. Sahbani and W. Adiprawita, "Kalman filter and iterative-hungarian algorithm implementation for low complexity point tracking as part of fast multiple object tracking system," in *2016 6th International Conference on System Engineering and Technology (ICSET)*. IEEE, 2016, pp. 109–115.

[16] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998. [Online]. Available: https://doi.org/10.1177/027836499801700706

[17] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Kdd*, vol. 96, Conference Proceedings, pp. 226–231.

[18] R. Zhang and S. Cao, "Extending reliability of mmwave radar tracking and detection via fusion with camera," *IEEE Access*, vol. 7, pp. 137 065–137 079, 2019.

II

Preliminary Report

1 Introduction

The task of safe indoor navigation of micro air vehicles (MAV) remains a challenging one, requiring the MAV to reach a destination while also sensing and avoiding (S&A or SAA) obstacles. Apart from dealing with cluttered GPS denied environmental conditions and closed tight spaces, MAVs are also constrained by computational, power and weight limits. For this reason, the use of cheap, lightweight, and passive vision systems are among the most popular methods, being a rich source of information. Vision systems however require an adequate amount of computational power and will not function in poor visibility conditions. This is especially important when using MAVs to provide aid and assistance in a disaster zone scenario, where lighting might not be available and moving objects/persons could be present.

This is the premise of the IMAV 2021 competition, to locate bodies in an indoor dark environment, and navigate in the presence of moving obstacles and limited visibility, the latter of which the research aims to tackle. Radar and ultrasound sensors are the only ones which can operate in low visibility conditions. Ultrasound sensors however have more difficulty sensing soft or curved edges, and its accuracy is sensitive to variations in temperature. Additionally, turbulent airflow from the propellers can interfere with the readings and can get very noisy requiring signal processing. Radar has the advantage of also being able to detect the doppler signature of a moving object, from which the radial velocity can be extracted, and thus is more suited to solve the problem at hand.

The problem of S&A can be divided into two sub-problems, namely sensing (or detecting) and avoiding. The former deals with selecting the appropriate instrument and signal processing to convert the raw data into a meaningful representation, while the latter involves further processing and use of the autopilot for extracting information to determine the state of the environment and vehicle, and provide commands to avoid a collision with the use of a controller. There are three types of control [58]: reactive, behavioural and deliberative. Reactive control involves minimal processing where the sensors are directly mapped to the actuators or where the avoidance algorithm only considers the instantaneous, local sensor information in the vicinity of the robot. On the other hand, deliberative control creates an internal known representation/model of the environment (white box) which is separated from the avoidance controller, often requiring a lot more memory and processing power. This can apply both to sensing and avoiding: the standard processing pipeline otherwise known as detection and tracking of moving obstacles (DATMO) [35] is used to explicitly estimate the state of the environment, while global path planning algorithms serve as the controller. In between there is behavioural control (grey/black box), which provides additional and more complex processing/representation than reactive control and is also more robust to sensor noise than deliberative control, often involving neural networks (NN).

This literature study serves to explore the latest knowledge and applications of S&A on an autonomously navigating MAV in a dynamic, low visibility environment by searching the relevant and latest industry standards, technologies and literature on the matter, such as object tracking and applications of artificial neural networks (ANN) and spiking neural networks (SNN). This will involve researching what type of sensors are suitable for S&A in the mentioned environment, how these sensors operate, what type of data they produce, their limitations and the requirements they should fulfil. Similar questions have to be answered regarding the conventional processing methods (in this case deliberative control) for S&A regarding their limitations, the methods that are more established than others, how they can be integrated into the implementation and if they meet the restrains of the problem. Furthermore, other recent developments and non-conventional behavioural methods can be investigated, in what ways they perform differently and what situation/environment they are more suited for. Lastly, an analysis can be done to determine which methods can be combined to design a system specific for this implementation, and what qualitative or quantitative metrics should be used to evaluate the controller.

Research Questions

In order to get an overview of the requirements the project should fulfil, the following questions should be considered:

- What sensors are appropriate to perform S&A in the given environment?
  - What are the limitations of the sensor?
  - What requirements must the sensor fulfil?
  - What type of data is produced by the sensor?
- What existing methods can perform S&A with the given sensors?
  - What are the limitations of the method?
  - Which methods are more standard/established than others?
  - How can they be integrated to process the data and tackle the problem?
  - Do the methods fulfil the requirements on their own?
- What parameters determine the effectiveness of a behavioural controller?
  - To what complexity should the controller be trained/evolved?
    - ◇ What objectives (fitness functions) should be optimised?
    - ◇ What parameters should be optimised?
  - What should the fidelity level of the simulation be?
  - What type of controller should be used?
    - ◇ Which inputs should the controller accept?
    - ◇ What type of output should the controller give?
    - ◇ Which models should be used to construct the controller?
- What is the significance of the comparison between the behaviour controller and the standard method?
  - What quantitative/qualitative measures can be used to compare the methods?
  - What conclusions can be drawn about the proposed method compared with the standard methods?
  - What improvements and future developments can be recommended?

Research Objective

The objective of this research project is to contribute to the development of a dynamic obstacle avoidance system aboard a MAV by providing an assessment of a neural network behavioural controller trained using evolutionary strategies to utilise data from a FMCW radar.

This study, based on a literature study and theory on radar, tracking, evolutionary strategies and simulation will give insight into the requirements and constrains of the design, by which means the neural network behavioural controller will be trained and evaluated in contrast with traditional deterministic methods such as velocity obstacles. The confrontation of the results will contribute to the development and evaluation of the dynamic obstacle avoidance system aboard a MAV.

As for the IMAV competition, the focus is on dynamic obstacle avoidance in an indoor environment with a small lightweight MAV and poor visibility. The task is relatively straightforward (at least for humans), and cosidering the small and lightweight constraints the solution should not be overly complicated. Using EAs to train neural net controllers is an elegant solution, and although they are rarely implemented on MAVs, especially using radar in a dynamic environment, there is no reason it cannot be applied to this situation as well. By comparing it with a simple standard tracking and velocity obstacles method, its performance can be evaluated to discover the benefits and solutions the algorithm can evolve.

This report is structured as follows: Chapter 2 will cover the operation and characteristics of radar instruments in consideration, followed by methods used for the standard tracking procedure in Chapter 3. Chapter 4 will give an overview of the different types of controllers used that generate commands for the flight controller. Chapter 5 and Chapter 6 will follow by investigating the use of (spiking) neural nets and evolutionary algorithms, as they have some desirable properties. Chapter 7 will follow up with an analysis of the gathered information, and how it can be applied to the S&A problem, and lastly the conclusion will be given in Chapter 8.

2

Sensor Detection of Moving Obstacles

The majority of the applications concerning S&A focus on self-driving cars and autonomous ground-based robots. As such, weight and power (both computational and electrical) are not prioritised as much, often implementing expensive and heavy Lidar array scanners, which produce denser, more memory intensive 3D point clouds than radar, or use multiple stereo camera setups which require more intensive computer vision algorithms. Moreover, deliberative control is the standard in these industries as a more deterministic approach is preferred to guarantee the predicted control characteristics, although artificial intelligence (AI) can be used to extract meaningful features, especially regarding vision. This is usually combined with DATMO (detection and tracking of moving obstacles), which traditionally requires some form of localisation or positioning system to perform odometry and estimate the ego-state of the vehicle, which is usually accomplished using GNSS (e.g. GPS or DGPS) and accurate/expensive IMUs.

However, as is the case with indoor navigation of a MAV, the only direct information about ego-motion comes from MEMS gyroscopes and accelerometers, which can be used for some rough state estimation. As an alternative, the external sensors onboard the vehicle can be used to estimate the state of the vehicle, for instance when employing simultaneous localisation and mapping (SLAM), an inexpensive way to localise [17]. Most studies however assume that the environment is mostly static. When dealing with a dynamic environment, the problem of SLAM and DATMO are combined, also known as SLAMMOT [43, 59]. Employing this method with radar based sensors is difficult however due to the sparse and noisy nature of the data.

2.1. Sensing using radar

Radar has been used since the 2nd world war to detect enemy craft. It is a time of flight sensor, meaning it determines the distance to another object by the time it takes for the radar energy to travel to the object and reflect back to the sensor. Unique to radar, it can also detect the change in frequency of the returned energy, and thus can separate the target from the background clutter easier.

It can illuminate the object in two different ways, either using a pulse or a continuous wave (CW) radar, although the principles and processing methods are quite similar. A pulse radar transmits a pulse train (pulse width modulated - PWM) radar carrier wave and listens for the returns, whereas a CW radar transmits a continuous carrier wave. In the context of the IMAV competition which focuses on indoor operation of a drone in a low-visibility, dynamic environment, the requirements are to detect and avoid dynamic obstacles/posts, which are $0.5m$ in diameter and moving at a velocity

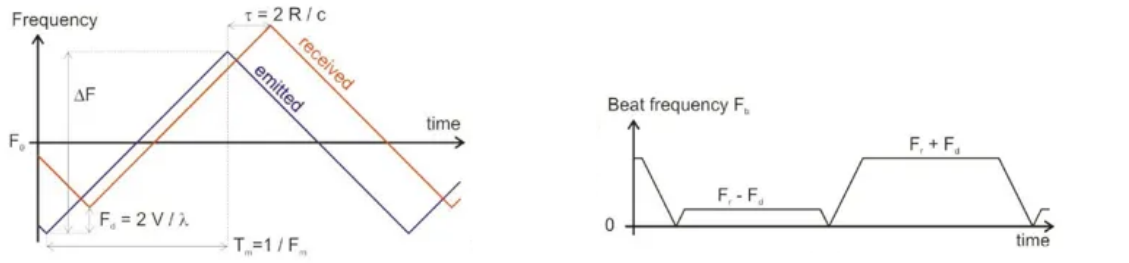
up to $2ms^{-1}$ in an enclosed space of around 4x4m.

The frequency band of a radar instrument determines its properties such as range, regulations, interference, and through-wall capabilities. The higher the frequency, the more bandwidth that is available and hence the better the range resolution. Therefore, for S&A, we speak of millimetre wave (mmW) radar. 24GHz (K band radar) is a popular radar frequency used in industrial and autonomous driving automotive applications and is also suited for through-wall sensing. It suffers from high atmospheric attenuation due to the resonance peak of water vapour, which limits it usage to short range applications, such as indoor sensing. It is also more susceptible to erroneous detections due to interfering radar transmitters and subject to regulations, which limits the available bandwidth.

For these reasons, some industries are switching to 77GHz radar, which offers improved range resolution and accuracy (up to a factor of 20 due to higher bandwidth) and also a higher velocity resolution (up to a factor of 3 due to the higher frequency) [46]. These bands commonly use a frequency modulated CW radar (FMCW). Generally speaking, higher frequencies will reflect better and radar waves are scattered better by materials with a higher dielectric constant, especially regarding conductive materials such as metals or carbon fibre. Water, concrete, gyps and wood are also good reflectors of radar waves, while plastics and glass are poor reflectors[34]. It should also be noted that with this increased resolution, there will be multiple detections per object, and a more robust data association algorithms would be needed (see Chapter 3).

2.1.1. FMCW radar

A frequency modulated continuous wave (FMCW) radar implementation can be used to determine the radial distance, velocity, and angle of multiple objects. Traditional FMCW radar operates by transmitting a saw-tooth FM carrier wave, and then listens for the returns (one cycle is referred to as a chirp). The received signal in Figure 2.1a is shifted to the right of the emitted signal (delay) with increasing distance and is shifted vertically due to the Doppler effect (an approaching object will shift the signal upwards). As can be seen in Figure 2.1b, the difference of the transmitted and received signal produces an IF signal (one for each object). This signal is then passed through a low pass filter, followed by an ADC and a DSP for processing and extracting the range frequency F_r and Doppler frequency F_d.



(a) Emitted (transmitted) and received signals of one chirp.

(b) Corresponding IF/beat frequency by taking absolute difference

Figure 2.1: Radio signals of FMCW radar. Adapted from [59]

An extension of this implementation is known as a 'fast-chirp FMCW' radar which offers an improvement in range resolution[38][57]. While in classical FMCW radar a chirp typically lasts $1-10ms$, the modulation time in a fast-chirp FMCW radar is limited to under $100\mu s$, and the system only considers the up-ramp of the modulation that is generated by the synthesizer (Figure 2.3), giving reduced range but better resolution. This is especially useful in short range applications. Fig-

ure 2.3 does not show the effect of the Doppler frequency shift as with fast-chirp FMCW radar, it becomes significantly smaller than the range frequency shift and can be neglected. Also note that the delay of the reflected signal τ is typically a small fraction of the total chirp time. As with classical FMCW radar, the transmitted and received signals are mixed to produce the IF signal and is transformed into a range FFT. To be able to distinguish between two separate objects at different distance, they also need to be distinguishable in the range FFT, which is achieved by increasing the bandwidth, which effectively increases the observation time if the slope S remains constant (if the slope were to change instead, this would put more demand on the sampling frequency of the ADC). The range resolution Δd and maximum range d_{max} are given by Equation (2.1) and Equation (2.2).

$$\Delta d = \frac{c}{2B} \tag{2.1}$$

$$d_{\max} = \frac{F_S c}{2S} \tag{2.2}$$

Where c is the speed of light, B is the bandwidth, F_s is the sampling frequency and S is the slope of the frequency modulated ramp. Also note that larger observation time requires a lower F_s of the ADC. The range can then be calculated using Equation (2.3), where T_c is the up-chirp time and f_b is the beat frequency (IF).

$$R = \frac{cT_c f_b}{2B} = \frac{cf_b}{2S} \tag{2.3}$$

$$\Delta\omega = \frac{4\pi V T_c}{\lambda} \tag{2.4}$$

The velocity of the detected object is determined by the change of phase over 2 chirps, since the phase is very sensitive to small changes in distance (essentially, a phase change is a Doppler frequency change). This change in phase $\Delta\omega$ is given by Equation (2.4), where V is the radial velocity of the target and λ is the wavelength. By taking N number of IF samples instead of 2, a velocity estimate can be computed for each individual object by taking a Doppler FFT of $\Delta\omega$. Again, the longer the observation time (the more samples N), the better the velocity resolution. Combining the range and Doppler FFT creates a 2D FFT (Figure 2.2), which identifies and resolves objects by their range and radial velocity. Following this, the horizontal bearing of the object to the sensor can be estimated by adding a second receiver antenna and analysing the phase difference between the two. Similar to the velocity estimates, adding more samples from more antennas increases the angular resolution by creating an angle FFT, that is, the angular resolution of two objects in the same range-velocity bin in the 2D FFT[2, 62]. This also means that better range and velocity resolution eases the requirements on the angular resolution. The equations for velocity resolution and range are given by Equation (2.5) Equation (2.6) and Equation (2.7).

$$\Delta v = \frac{\lambda}{2NT_c} \tag{2.5}$$

$$v_{\max} = \frac{\lambda}{4T_c} \tag{2.6}$$

Where λ is the wavelength, N is the number of IF samples and T_c is the observation time of one ramp. Similarly for the angular resolution and bearing calculation:

$$\Delta\theta = \frac{\lambda}{Nd\cos(\theta)} = \frac{2}{N} \tag{2.7}$$

$$\theta = \arcsin\left(\frac{\lambda\Delta\omega}{2\pi d}\right) = \arcsin\left(\frac{\Delta\omega}{\pi}\right) \tag{2.8}$$

For $\theta = 0$ and $d = 0.5\lambda$ (which gives the best largest field of view of 90°). Note that $\Delta\omega$ in Equation (2.8) refers to the phase difference between the two antennas, and not the phase difference between chirps as in Equation (2.4). The radar unit currently in consideration for this research is the Infineon XENSIV 24GHz Position2Go development kit, a small 10g fast-chirp FMCW radar that features human target detection at a range of 1-12m and a horizontal field of view of 76deg at half power beamwidth. For every frame, it outputs the range, bearing and velocity of every target detected. According to the data sheet and specs of the radar, with a maximum bandwidth of $200MHz$ the range resolution is around $0.75m$, and the velocity resolution can be as low as $0.04ms^{-1}$ with a chirp time $T_c = 0.3ms$ and 64 chirps per frame. From testing, the range accuracy is around $\pm 15cm$ and the angular accuracy is around $\pm 2°$ from $0-20°$, and up to $\pm 8°$ from $20-65°$ [1]. If these properties are insufficient, a $77GHz$ radar can be considered.

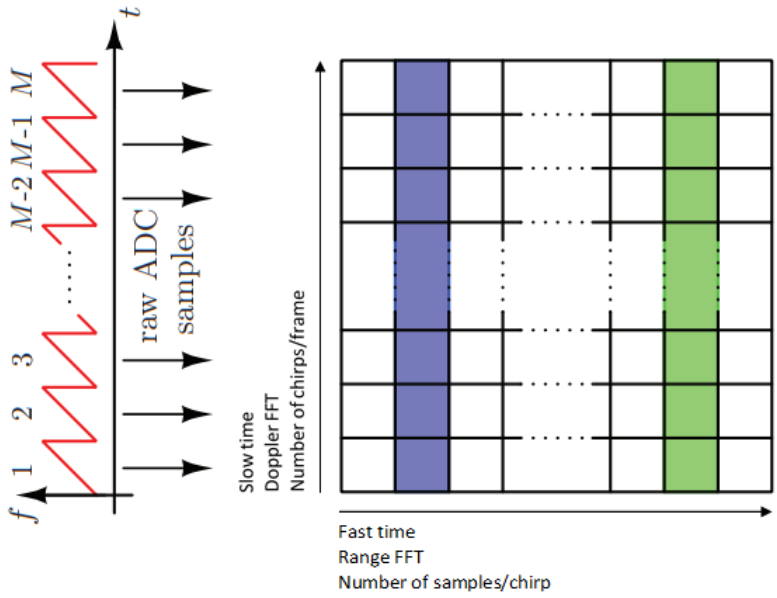


Figure 2.2: A 2D FFT, where the horizontal axis represents fast time (one chirp) and the vertical axis represents slow time (one frame). After performing FFTs on the fast time axis identifying any peaks (shaded regions), an FFT is performed along the slow time axis to determine the phase change (doppler shift) of same range bin can be compared to distinguish one or multiple peaks (objects with different velocities at same range). Adapted and modified from [38]

.

2.2. Data representation

There are several ways that can be used to represent data to facilitate further processing. The most fundamental is a point cloud, which is natively produced by the sensor (2D plane point cloud). Usually when dealing with point clouds, they come from scanning Lidar instruments, which produce a much denser point cloud than radar sensors, and also return a fixed amount of points every frame instead of a variable amount of detection like radar. The data received from consecutive frames can be aligned into an global frame of reference using a class of methods known as point cloud registration (PCR), such as Perfect Match (PM), Iterative Closest Point (ICP) and Normal Distribution Transform (NDT) [53], which is one of the primary steps involved in SLAM. However, for sparse and noisy

[1] https://www.infineon.com/dgdl/Infineon-P2G_Software_User_Manual-ApplicationNotes-v01_01-EN.pdf?fileId=5546d4626b2d8e69016b89493bf842af

radar data this becomes inaccurate as consecutive frames might differ significantly. Feature based representation extracts meaningful features from the data and converts or labels it (e.g. by colour, shapes, geometries) by assuming some knowledge of the environment model, which reduces the memory and processing requirements. Again, since radar data only produces one detection per object due to the low resolution, this cannot be applied.

The last representation type is grid based, and there are several types. 2.5D (height maps) discretise the space into a 2D grid on the ground, where each grid contains the maximum height of an object occupying that grid, while 3D voxelised grids contain the full 3D representation of the environment. However since the FMCW radar sensor only produces 2D data (it can only measure angles horizontally), we speak of occupancy grids in which the observable space is discretised into pixels that states whether the space is occupied by an object or not (often probabilistic/Gaussian). This is a means to ensure that each frame contains a fixed number of data points, and also allows for image processing techniques to be applied, such as optical flow or using convolutional neural nets.

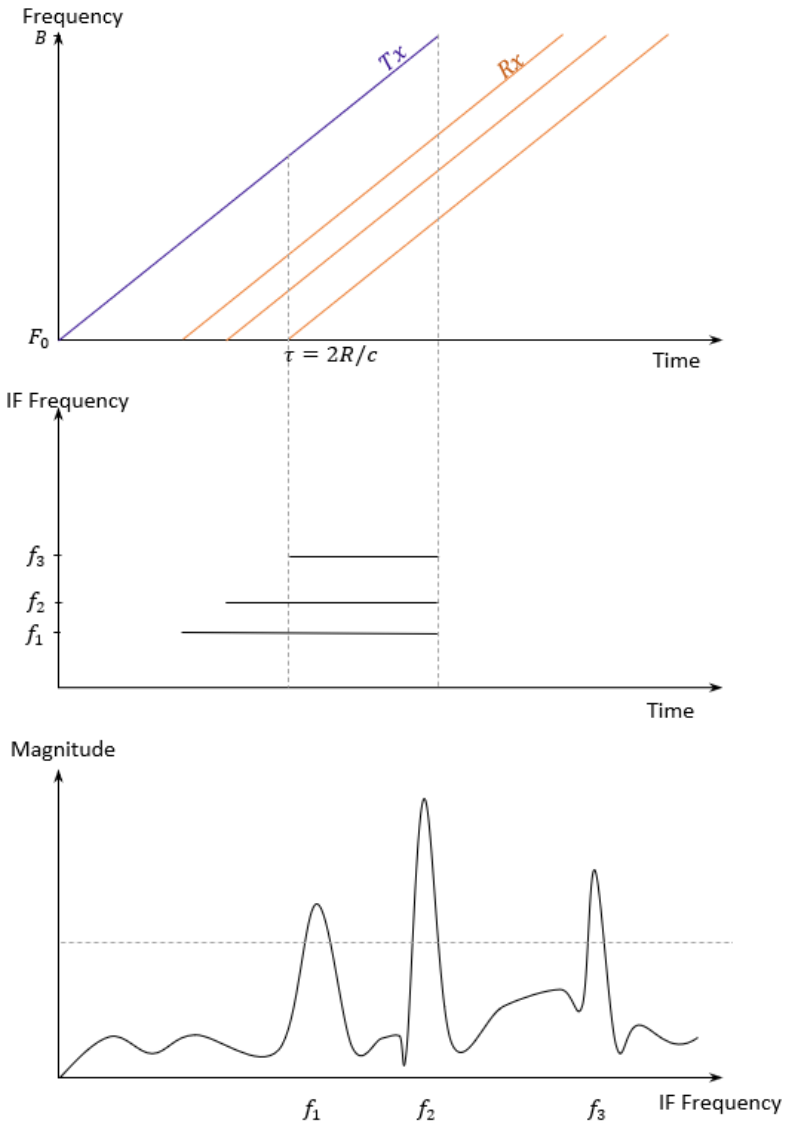


Figure 2.3: Frequency analysis of fast-chirp FMCW radar, showing the Transmitted (Tx) and received (Rx) signals from multiple objects, and mixed to produce the IF signal, and FFT transform of IF signal to produce range FFT, delineating the 3 objects

3

Multi-Target Tracking

While avoidance algorithms can directly use this data to avoid stationary objects, it is important to track any dynamic moving objects to assess the risk of collision, also known as multi-target tracking (MTT) [1]. This involves two steps which work in conjunction with each other: associating each new data point with an existing target, and determining the state (position, velocity, acceleration) of the object. Once this is known, the autopilot or controller can adjust the state of the vehicle to avoid collisions.

3.1. Data Association

Data association is the process which assigns data points to the various targets in the environment [14]. In some literature, this process is split up into two: gating and association, where gating discards unlikely measurement candidates, and the association step converts a likelihood matrix into an assignment matrix according to one of several algorithms. This also involves creating new tentative tracks for measurements that do not conform with existing tracks. Once these tentative tracks meet certain criteria, the track is confirmed. Tracks which no longer have any associated measurements are deleted. The process of data association becomes especially difficult when the objects are in close proximity to each other or in the midst of clutter, where it becomes more difficult to distinguish between the two. All data association filters use the predicted state of the target based on the tracking algorithm that is carried out after this step. Note that it is assumed that only one measurement (radar detection) will be recorded per obstacle.

When considering MTT, we speak of the various Gaussian densities (probability density functions) given below, namely the posterior density (probability distribution of the state estimate given measurements from time $t = 0$ to τ), the transition density, and the measurement likelihood, where Q and R are the covariance matrices of the process and measurement noise, respectively, as will also be covered in the Kalman Filter process in Section 3.2. With the KF, the state estimate is taken as the mean of the posterior density. This is a recursive process, where the next time step prediction of the posterior density $p(x_k|z_{1:k-1})$ is made based on the model and current density, after which the estimate $p(x_k|z_{1:\tau})$ is calculated.

$p(x_k\|z_{1:\tau}) = \mathcal{N}(x_k; \hat{x}_{k\|\tau}, P_{k\|\tau})$	posterior density
$p(x_k\|x_{k-1}) = \mathcal{N}(x_k; f_{k-1}(x_{k-1}), Q_{k-1})$	transition density
$p(z_k\|x_k) = \mathcal{N}(z_k; h_k(x_k), R_k)$	measurement likelihood

It is also important to understand the clutter and measurement models. The clutter (false alarms) can be modelled using the Poisson point process, where the number of clutter detections $m_k^c \sim Po(\lambda V)$, where λ is the expected number of clutter detections per unit volume and V is the observable volume (FoV), over which they are distributed according to the intensity function $\lambda_c(c)$ (with corresponding pdf $c \sim f_c(c)$ within V, and equal to zero outside V). Often, this is uniformly distributed $f_c(c) = 1/V$. The vector C_k then represents all the clutter detections. A measurement model can then be defined $Z_k = \{O_k, C_k\}$, where the measurements consist of clutter and object detections O_k, which occur with a probability of $P^D(x_k)$, where the individual detections have densities $o_k \sim g_k$ (or measurement likelihood $p(z_k|x_k)$ above). These models will also become important for simulation of the radar sensor.

The problem is to then assign the measurements to the objects with an association θ_k (Equation (3.1)) at time $t = k$, and this final posterior density $p(x_k|Z_{1:\tau})$ can be found by performing a weighted sum of all of the individual densities corresponding to the different hypotheses (association possibilities) as given in Equation (3.2), where the second factor of the sum (Pr) is the (unnormalised) weight, the probability that the hypothesis is valid and is denoted by w_θ, which is given by Equation (3.3).

$$\theta_k^i = \begin{cases} j & \text{if measurement i is associated with object j} \\ 0 & \text{if object j has been misdetected} \end{cases} \tag{3.1}$$

$$p(x_k|z_{1:\tau}) = \sum p(x_k|Z_{1:k-1}, \theta_{1:k}) \cdot Pr[\theta_{1:k}|Z_{1:k}] \tag{3.2}$$

$$w_\theta = Pr[\theta_{1:k}|Z_{1:k}] = \begin{cases} 1 - P^D & \text{if } \theta_k^i = 0 \\ \frac{P^D \mathcal{N}(z^\theta; \bar{z}, S)}{\lambda_c(z^\theta)} & \text{if } \theta_k^i = 1...m \end{cases} \tag{3.3}$$

Where $\bar{z} = Hx$ and $S = HPH^T + R$ (H is the observation matrix and S is calculated in the KF process). However, in reality it is intractable to keep track of all possible hypotheses which grows exponentially over time, and thus assignment algorithms can either prune the hypotheses (which removes the ones with small weights) or merge the hypotheses (replace the sum of all Gaussian densities with a single Gaussian density). Generally when using merging, some form of pruning is done beforehand such as gating.

When using the simplest data association method, the global nearest neighbourhood (GNN) [43] approach, pruning is employed such that only the hypothesis with the largest weight is considered. This is a very greedy and computationally efficient approach that works well for simple scenarios. However the algorithm struggles to make correct associations in high clutter, close proximity environments. Another method is the joint probabilistic data association filter (JPDAF) [1, 14], which instead uses merging to reduce the number of hypotheses into one for every track. JPDAF however cannot account for object births and deaths. Instead of considering a single hypothesis, the multiple hypothesis tracking (MHT) [33] algorithms takes multiple hypotheses into account over different time steps, and thus is more likely to converge towards the true posterior density. GNN is a MHT algorithm where only one hypothesis is considered.

For an unknown number of multiple objects, the data association becomes an assignment optimisation problem, where an assignment matrix A should be generated to minimise the cost: $A^* = argmin(tr(A^T L))$, where L is the cost matrix. This is equivalent to minimising the sum of the negative log weights w_θ. Both A and L are $(n \times m + n)$ matrices, where n is the number of detections and m is the number of tracks. Equation (3.4) shows the structure of the cost matrix, where the left section encodes for the assignment of the measurements to the tracks, and the right section encodes

for misdetections. The problem is bound by the constraints in Equation (3.7).

$$L = \left[\begin{array}{cccc|cccc} -\ell^{1,1} & -\ell^{1,2} & \dots & -\ell^{1,m} & -\ell^{1,0} & \infty & \dots & \infty \\ -\ell^{2,1} & -\ell^{2,2} & \dots & -\ell^{2,m} & \infty & -\ell^{2,0} & \dots & \infty \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -\ell^{n,1} & -\ell^{n,2} & \dots & -\ell^{n,m} & \infty & \infty & \dots & -\ell^{n,0} \end{array}\right] \tag{3.4}$$

$$\ell^{i,0,h} = \log\left(1 - P^D\right) \tag{3.5}$$

$$\ell^{i,j,h} = \log\left(\frac{P^D V}{\bar{\lambda}_c}\right) - \frac{1}{2}\log\left(\det\left(2\pi S^{i,h}\right)\right) - \frac{1}{2}\left(z^i - \hat{z}^{i,h}\right)^\top \left(S^{i,h}\right)^{-1}\left(z^j - \hat{z}^{i,h}\right) \tag{3.6}$$

$$\begin{array}{lll} \text{subject to} & A^{i,j} \in \{0,1\}, & \text{i, } j \in \{1,\dots,n\} \times \{1,\dots,n+m\} \\ & \sum_{j=1}^{n+m} A^{i,j} = 1, & i \in \{1,\dots,n\} \\ & \sum_{i=1}^{n} A^{i,j} \in \{0,1\}, & j \in \{1,\dots,n+m\} \end{array} \tag{3.7}$$

In reality a lot of different methods simplify the elements of the cost matrix by only considering the Euclidean or Mahalanobis distance (which is the last term of Equation (3.6), given in Equation (3.8)). Furthermore, Gating is used to discard very unlikely measurements and reduce computational load by creating a bounding region (gate) outside of which any measurements are immediately discarded. This region is defined by Equation (3.8) which forms an ellipse. If this distance is greater than a certain threshold $r^2 > G$, it is discarded (i.e. $\ell^{i,j,h} = \infty$). $r^2 \sim \chi^2(n_z)$, and the threshold G can be selected such that $Pr[r^2 > G] = P_G$, where P_G is a value such as 0.995. It turns out however that this ellipse is computationally expensive to deal with, and a bounding rectangle can be used instead which encompasses the ellipsoid, again based on the Mahalanobis distance.

$$[z_k - \hat{z}_k]^T S^{-1} [z_k - \hat{z}_k] = r^2 \tag{3.8}$$

A number of solvers can be used to find the optimal assignment matrix, the simplest being the Hungarian algorithm. Furthermore, object birth and death also has to be dealt with. Object death occurs when the tracked object is misdetected for a certain number of frames. Object birth happens when clutter detections are associated with new tentative tracks. If the posterior density increases beyond a certain threshold, the track is confirmed and becomes a new track.

Other approaches to data association include the use of simple artificial neural networks (ANN). Chung et al. 2007 [11] employs a Hopfield neural net, a simple type of binary, recurrent NN. The network is trained at every time step to map new detections with the target positions in the previous time frame. ANNs can also be used in conjunction with a Kalman filter (KF), as Silven 1992 [52] demonstrated, where the ANN is able to convert a likelihood assignment matrix into a binary assignment matrix, which essentially solves a network of differential equations.

3.2. Obstacle state estimation

Kalman Filters

After new detections are associated with new or existing targets, tracking can be done independently to determine the state (position, velocity or acceleration) of the objects. The most common method is the Kalman filter (KF), which is a weighted average between the measured and predicted state.

$$\begin{aligned} \underline{x}_{k+1} &= \Phi_{k+1,k}\underline{x}_k + \Psi_{k+1,k}\underline{u}_k + \Gamma_{k+1,k}\underline{w}_{d,h} \\ \underline{z}_{k+1} &= H_{k+1}\underline{x}_{k+1} + D_{k+1}\underline{u}_{k+1} + \underline{v}_{k+1} \end{aligned} \tag{3.9}$$

Given a discretised system with process noise $\underline{w}_{d,h}$ and sensor noise $\underline{v}_{k+1}$ as given in Equation (3.9), where $\Phi_{k+1,k}$ is the system transition matrix, $\Psi_{k+1,k}$ is the input distribution matrix, $\Gamma_{k+1,k}$ is the noise input matrix, H_{k+1} is the observation matrix and D_{k+1} is the feedforward matrix. Using the 5 steps in Figure 3.1, the state can be estimated using the Kalman gain K, which depends on the uncertainties in the system by optimising a quadratic weighted least square cost function which tries to minimise state prediction error and the measurement error. In the case of tracking, the objects position, velocity and acceleration can be estimated.

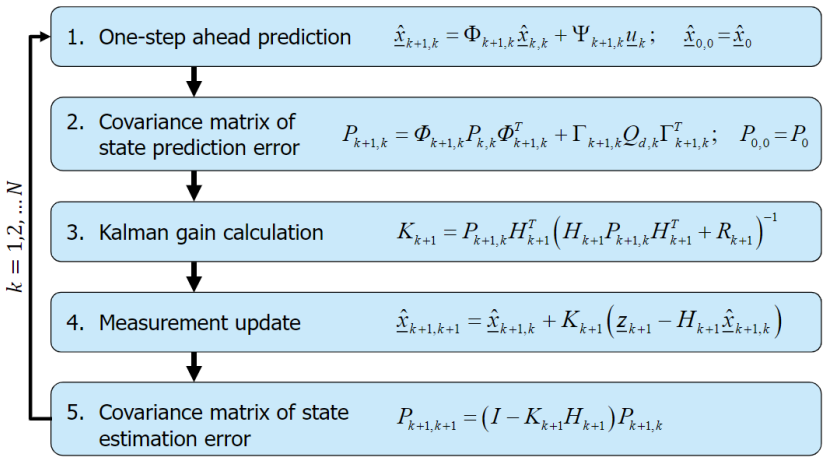


Figure 3.1: 5 steps of the Kalman filtering process. Adapted from AE4320, System Identification of Aerospace Vehicles, Delft University of Technology, Dr.ir. Daan Pool

Where $\underline{\hat{x}}_{k+1,k}$ is the one-step-ahead state prediction, $\underline{\hat{x}}_k$ is the current optimal estimated system state, $\underline{\hat{x}}_{k+1,k+1}$ is the one-step-ahead optimal state estimation, and $P_{k,k}$, $Q_{d,k}$ and R_{k+1} are the covariance matrices of the current estimation error $\underline{\hat{\epsilon}}_{k,k}$, known process noise $\underline{w}_{d,h}$, and known sensor noise $\underline{v}_{k+1}$, respectively.

The Kalman filter however, only applies to linear systems, assumes the noise covariance matrices are known and requires a fully observable system to converge. When the system behaves differently in different environments, sometimes the interacting multiple model (IMM) algorithm is used which uses two or more KFs with different models and selects the best one. Alternatively, the extended Kalman filter (EKF) can be used for non-linear systems, which linearises the system about nominal state and input values. Because of this, the system is now defined by the perturbations δx, δu and δz from the nominal state in Equation (3.10), requiring the calculation of the Jacobian $F_x(\bullet)$ and $H_x(\bullet)$ of the transition and observation matrices, respectively (the $\bullet$ represents the nominal condition around which the system is linearised). After this is discretised, steps 2-5 in Figure 3.1 are followed as with the standard KF. The iterative EKF takes this one step further and iterates on the Jacobian, Kalman gain and measurement update steps, to allow for better convergence and less sensitivity to initial conditions.

$$\begin{aligned} \delta\underline{\dot{x}}(t) &= F_x(\bullet)\,\delta\underline{x}(t) + G(\bullet)\,\underline{w}(t) \\ \delta\underline{z}(t) &= H_x(\bullet)\,\delta\underline{x}(t) + \underline{v}(t) \end{aligned} \tag{3.10}$$

Particle Filters

Another estimation method is a particle filter (PF), which is an iterative estimation method that evaluates the probability of hypotheses (particles) (it can also be closely compared to evolutionary

algorithms). It starts with a set of N particles with randomly distributed positions and velocities and equal normalised weights $1/N$:

1. One step ahead prediction $\hat{z}_{i,k} = h(\hat{x}_{i,k}) = h(f(\hat{x}_{i,k-1}))$
2. Calculate probability based on error $\epsilon_{i,k} = (z_k - \hat{z}_{i,k})$
3. Multiply particle weights by probabilities and re-normalise weights
4. Calculate estimate based on weighted average
5. Create new set of particles with weights $1/N$, distributed based on current particle weights
6. Calculate sample covariance $P_k = \frac{1}{n-1}\sum_i^n \left(\hat{x}_{i,k} - \bar{x}\right)\left(\hat{x}_{i,k} - \bar{x}\right)^T$
7. Add perturbations to particles by sampling from P_k and multiplying by tuning parameter

Particle filters can coarsely estimate a state as well as the likelihood of a measurement error for many different types of problems with different models. However, due to the large number of particles required to accurately estimate the state, it can become computationally expensive. Moreover, since the uncertainty is represented as a set of particles and weights, the state estimation is usually coarse.

Neural Networks

There are several types of ANN that can complement or replace the standard Kalman filter (KF) to perform target tracking, especially when dealing with manoeuvring targets. Often, general regression neural networks (GRNN) are used, where the output of the network is a weighted sum of Gaussian radial basis functions of the inputs (the centres of these functions represent the input distribution/features). Training can then be done using standard back-propagation or also simple linear regression. GRNNs can replace KF for target state estimation, especially when considering manoeuvring targets [30, 32, 55]. Fun-Bin and Chin-Teng 2004 [22] uses a (hand crafted) neural fuzzy inference network to detect when manoeuvres occur and then adapts the covariance matrix in the KF process to allow for greater flexibility in the estimation process.

Recurrent neural networks (RNN) are also often used, which have recurrent connections in the hidden layer that simulate an internal state (or memory). RNNs have been shown to produce better results regarding sequential data and variable length inputs. However, standard RNNs have trouble deleting obsolete information at appropriate times when new input is presented, or do not place enough significance with new information.

To combat this, gated RNNs are used, such as long short-term memory (LSTM) networks which are among the most popular. These networks produce an output and state at every cycle. In Figure 3.2, the forget gate f_t processes the input and decides whether or not to delete information in the state $c(t-1)$. The input gate i_t decides whether or not to allow information to pass through the main gate $\bar{C}_t$ and add to the state, and the output gate O_t decides what information to output $h(t)$ from the state $c(t)$. In this way, it can hold the state of the system, while also being adaptive to new changes or unseen data. RNNs however require more memory for back-propagation to store all the inputs. Iter, Kuck et al. 2016 [28] which uses 2 LSTM networks to both predict the future position of a detected object and the distribution of likelihood (variance of prediction). Again, this is especially effective with manoeuvrable targets.

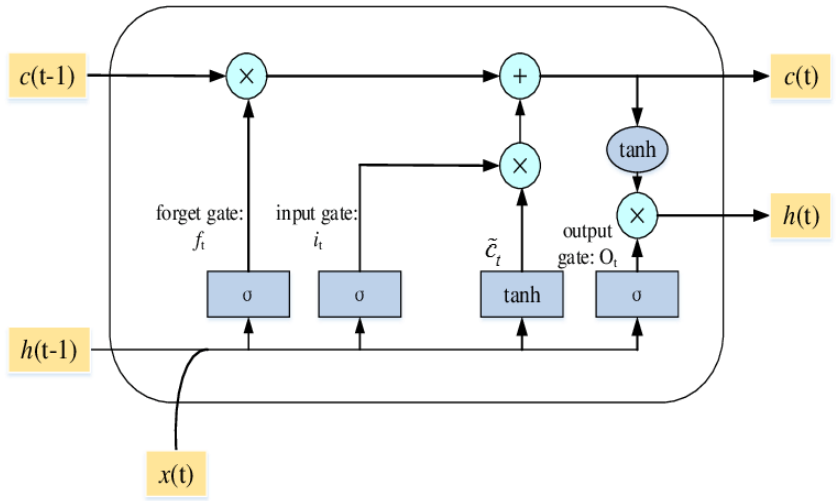


Figure 3.2: Structure of a recurrent LSTM network. Adapted from [63]

4

Avoidance of Moving Obstacles

Once the state of the environment and the vehicle is known, it can be determined whether or not a collision will occur and steer clear. Many methods however do not take into account the velocity of the obstacles, and merely update the response when the environment has changed, showing more erratic behaviour [13]. This can be done in a deliberative way where all of the information from the environment is used to guarantee convergence, such as global path planning methods. Deliberative methods however are quite memory and processing intensive, while also being less robust to noise. Alternatively, reactive (or local path planning) methods can be employed, which are less processing intensive but do not guarantee global convergence and may get stuck in local minima. Sometimes these methods can be combined Thrun et al. 1998 [15] that utilise a path planner to provide intermediate goals.

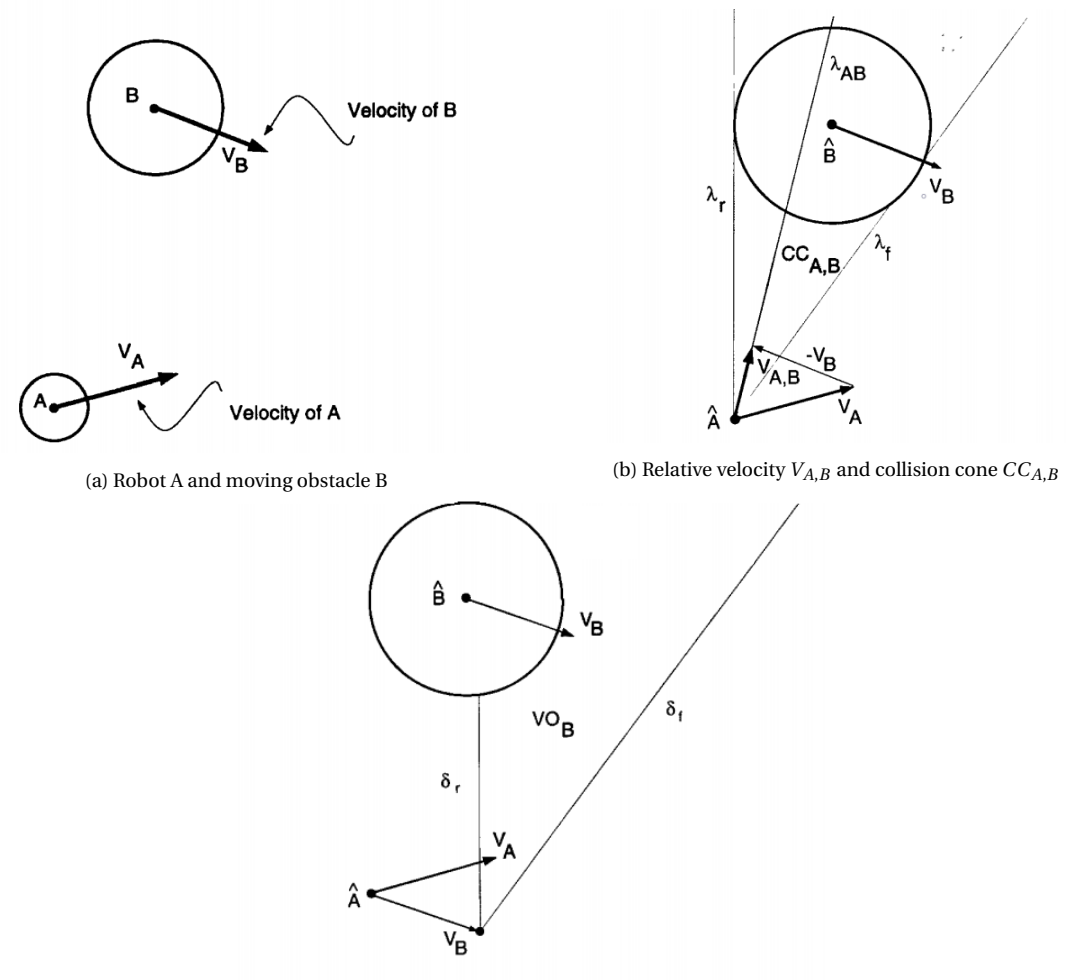
4.1. Reactive Controllers

One of the simplest reactive methods is known as velocity obstacles (VO), which determines the set of velocity vectors of the vehicle which will result in a collision with another moving object.

This explanation has been obtained from Fiorini et al. 1998 [18]. Consider a robot A and an obstacle B with velocities V_A and V_B and radii r_A and r_B, as given in Figure 4.1a. Mapping B onto the configuration space of A means enlarging object B by the radius of A to form object $\hat{B}$, and reducing A to a point $\hat{A}$ and computing the relative velocity of robot A with respect to object B, $V_{A,B} = V_A - V_B$, as shown in Figure 4.1b. The collision cone $CC_{A,B}$ can then be formed, in which any *relative* velocity $V_{A,B}$ will result in a collision with object B. Then the absolute velocities of robot A which result in a collision can be found by adding the velocity of object B, V_B, to the cone with respect to A to form the velocity obstacles $VO_B = CC_{A,B} + V_B$ (using the Minkowski vector sum operator) as seen in Figure 4.1c.

This can subsequently be done for every detectable obstacle, and the union of these will form the complete velocity obstacles for the whole environment: $VO = \cup_{i=1}^{m} VO_{B_i}$. Additionally, prioritising imminent collisions which will occur within a time frame T_h will free up some space for manoeuvring in case of clustered environments. This involves subtracting the set of velocities VO_h that will result in a collision after $t = T_h$: $|V_{A,B}| < d_m / T_h$, where d_m is the shortest relative distance between the obstacle and robot. Uncertainties and delays (or acceleration limits) with the robot dynamics also have to be taken into account, as well as the set of reachable velocities. The advantage of this method is that all velocities can also be processed relative to the drone's frame of reference, rather than an inertial one. To avoid obstacles, the algorithm should select the appropriate avoidance ve-

locities while also directing the robot towards a goal.



(a) Robot A and moving obstacle B

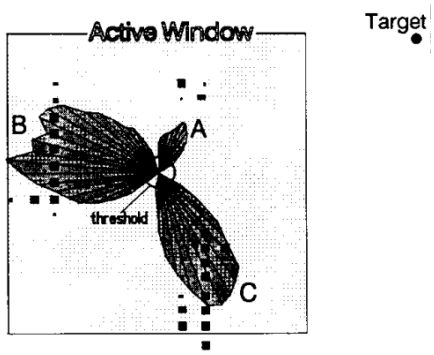
(b) Relative velocity $V_{A,B}$ and collision cone $CC_{A,B}$

(c) Velocity obstacle VO_B

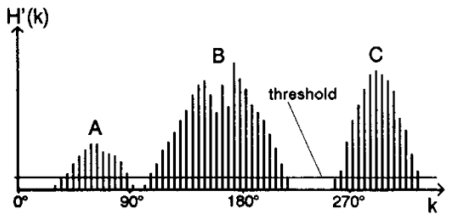
Figure 4.1: Velocity Obstacle procedure. Adapted from [18]

Another such reactive method is known as vector field histogram (VFH) [7]. The algorithms starts out by creating an (Cartesian) occupancy grid centred around the robot where the certainty $c^*_{i,j}$ that the grid (i, j) is occupied increases with more detections, thereby also taking into account previous detections. Each cell is then assigned a magnitude which is proportional to the square of the certainty and inversely proportional to the distance $m_{i,j} = (c^*_{i,j})^2(a - bd_{i,j})$, essentially signifying the danger to the robot. This is then converted to a one-dimensional polar histogram, where each sector of the histogram represents a small field of view, and the density of the histogram is the sum of the magnitudes of the cells in that sector $h_k = \Sigma_{i,j} m_{i,j}$ (see Figure 4.2). After applying a smoothing filter, a threshold is applied. Any regions below the thresholds are grouped together into valleys, and in the VFH algorithm, the valley closest to the goal is selected. An extension of this is the VFH+ algorithm, which also takes the robot's dimensions and dynamic limits into account by excluding any valleys that extend beyond these limits. Additionally, it provides a cost function to the valleys, and more optimally decides which valley to choose based on the angular distance of the valley (in the histogram) to the goal, the angular distance from the current robot orientation and angular distance from the previous direction selected. Building on this the VFH* algorithm simulates candidate directions one time step in advance and explores the subsequent branches, computing the cost after

N number of steps, thus allowing for a more global approach. Although VFH algorithms do not take into account the velocity of the obstacles, it is still shown to work for simple, slow moving robots. It should also be noted that data association and tracking are not necessary for this approach.



(a) Cartesian occupancy grid showing the robot facing right in the centre and detections of objects A, B and C, the size of the detection squares represents the magnitude $m_{i,j}$. The lobes are the histogram densities $h_{i,j}$



(b) Conversion into the histogram

Figure 4.2: Vector field histogram. Adapted from [7]

Other methods include using potential fields (from which VFH was derived) to generate a net force from the obstacles and goal that drives the robot, and the dynamic window approach that incorporates the dynamic and kinematic constrains of the robot in the optimisation problem.

4.2. Behavioural Controllers

Although the controllers in the above section could also be classified as behavioural in some ways, in the context of this report, behavioural controllers refer to the use of neural networks (NN) to map the sensory input to the output actuators.

As mentioned, behavioural control lies in between deliberative and reactive control, and neural networks are ideally suited for this type of control due to their smooth optimisation landscape, learning capabilities, parallel processing capabilities, adaptive solutions, and toleration to noisy input data. While the result of the tracking procedure from Chapter 3 could be used as input to the NNs, typically the raw sensors are directly used as input with no pre-processing, although all of the studies examined here are of a slightly different nature, and deal with stationary obstacles. When dealing with moving obstacles, especially for collision avoidance, most methods require some kind of knowledge about the state of the environment. Since this study is specific to radar implementation, the data produced from the radar sensor is also of a different nature than lidar/ultrasound/IR single point sensors: firstly the Doppler measurements already give an indication of the state of the obstacles by measuring the radial velocity, and secondly the radar produces data of variable size, returning the position and radial velocity of all detections, instead of a fixed, discretised set of range measurements.

It should also be noted that for a controller to be classified as behavioural, there should strictly be some sort of memory involved to keep track of targets or the previous state of the controller and exhibit more behavioural rather than reactive control. Alnajjar et al. 2009 [5] argue that adaptive (learning) and contextual (memory) properties are necessary for higher fidelity system and dealing with a changing environment types when constructing their modular controller, allowing it to build up of knowledge of the environment to use to its advantage. However, memory is also important to

store the state of the environment, which is necessary for all tracking algorithms. Moreover, modularity allows a complex task to be broken down into simpler sub-tasks, which each module can then solve independently. In fact when looking at existing studies that use behavioural controllers, even simple tasks that are trivial for humans are often too complex for a single controller to handle, and as such, the controller is made modular as some studies do in Chapter 5 and Chapter 6. Moreover, controllers often also use evolutionary algorithms (EA) as the optimisation algorithm as it can find solutions with a better interaction with the environment (fitness). This is especially useful when applied to spiking neural nets, as is explained in the following section.

Other methods besides neural networks can also be applied to create a behavioural controller. Tubb and Roberts 1998 [58] implement a neuro-fuzzy network, which processes commands and sensor input through set of weighted rules, followed by defuzzification, from which the robot can move forwards, turn a certain angle, or maintaining its heading. Another method to go about constructing an obstacle-avoid algorithm is to use genetic programming (not to be confused with genetic algorithm). Genetic programming evolves a program, a function of different operators such as addition, multiplication or non-linear function, represented as a tree structure, in order to become more proficient at performing a task. Oh and Barlow 2004 [41] applies it to a navigational controller for locating targets, although it lacks the advantages of NN, mainly robustness to noise and better adaptability/flexibility.

5

Spiking Neural Networks

Spiking neural networks (SNN), also referred to as 3rd generation neural networks, try to more accurately model biological neural networks and are especially well suited for processing spatio-temporal data and event based data, as well as being more robust to noise and more computationally efficient for their size [21, 44, 56]. For this reason, this section will discuss the basic operation and concepts behind SNNs.

Traditional artificial neural networks (ANN) have made tremendous progress over the years with the advent of training and inference in deep learning. However, ANNs fundamentally lack similarity to biological neural networks, which have evolved to optimise for energy efficiency, analogue computation, fast inference and parallelism. This is what SNNs try to mimic, and unlike with ANNs which consist of a network of neurons which output a value based on the non-linear but continuous activation function at every propagation cycle, SNNs consist of a network of spiking neurons connected with synapses, which either have the property of being weighted or having a transmission delay.

A post-synaptic spiking neuron generates an action potential (pulse or spike) when its membrane potential exceeds a threshold. This membrane potential is modulated by the spikes coming from pre-synaptic neurons or input according to the neuron model used. Additionally, information is encoded from digital/analogue input into a spike train and vice versa [60] according to one of several spike encoding schemes. Moreover, due to the event-based and timing-sensitive nature of SNNs, they are especially well suited for asynchronous neuromorphic hardware and sensors (such as dynamic vision sensors), making them computationally and energy efficient due to the scarce and information-dense spikes. Despite recent progress, SNNs have not yet achieved the same accuracy as the top performing ANNs on standard benchmarks such as MNIST or ImageNet [44], however these datasets were designed for conventional synchronous ANNs, which have been studied far more extensively.

5.1. Neuron models

In biological neurons, incoming spikes affect the voltage (membrane potential) of the soma, and when this reaches a threshold, it releases an action potential, which is characterised by a sharp increase in voltage followed by a long negative spike after potential (SAP), also referred to as the refractory period. The spikes travel down the axon and reaches the synapse, where through a complex set of reactions between the terminals, neurotransmitters and dendrites, the (excitatory or inhibitory) post synaptic potential (PSP) passes to the next neuron [23].

SRM model

In SNNs, the neuron models are drastically simplified, and dictate how the PSP affects the membrane potential of the neuron. Since action potentials are all very similar, they can be modelled instead by the timing of the spikes. According to the spike response model (SRM), the membrane potential $u(t)$ of neuron i can be modelled by Equation (5.1), where $\eta(t-\hat{t}_i)$ describes the effect of the action potential at time $\hat{t}_i$ and SAP on the membrane potential, and $\kappa(t-\hat{t}_i,s)$ being the response to the input current $I(t-s)$. When dealing with input from presynaptic spike arrival, the convolution of κ with the time response of the input spike current yields the PSP response ε as in Equation (5.2), where it is also assumed that κ is independent of the last spike time $\hat{t}_i$. That is, the membrane potential is the sum of the action potential effect η plus the sum of the PSP ε over every spike f and every synapse j with corresponding weight w_j. This weight is positive or negative depending on whether the synapse is excitatory or inhibitory. Furthermore, Equation (5.1) assumes the resting potential is zero (otherwise a constant is added).

$$u_i(t) = \eta(t-\hat{t}_i) + \int_0^\infty \kappa(t-\hat{t}_i,s)I(t-s)ds \tag{5.1}$$

$$u_i(t) = \eta(t-\hat{t}_i) + \sum_j \sum_f w_{ij}\varepsilon_{ij}\left(t-t_j^f\right) \tag{5.2}$$

The model (also referred to as simplified SRM) only takes into account the latest spike with regards to η. The neuron fires when u reaches a threshold ϑ:

$$u_i(t) = \vartheta \text{ and } \frac{d}{dt}u_i(t) > 0 \quad \Longrightarrow \quad t = t_i^{(f)} \tag{5.3}$$

The PSP kernel ε and action potential kernel η will be of the form in Equation (5.4) and Equation (5.5), which can be visualised in Figure 5.1.

$$\varepsilon_{ij}(s) = \left[\exp\left(-\frac{s-\Delta^{ij}}{\tau_m}\right) - \exp\left(-\frac{s-\Delta^{ij}}{\tau_s}\right)\right] H\left(s-\Delta^{ij}\right) \tag{5.4}$$

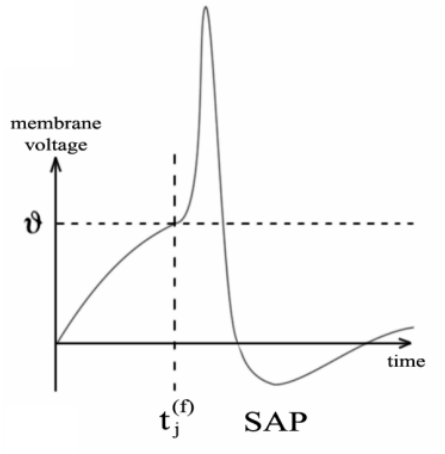
$$\eta(s) = -\eta_0 \exp\left(-\frac{S-\delta^{abs}}{t}\right) H\left(s-\delta^{abs}\right) - KH(s)H(\delta^{abs}-s) \tag{5.5}$$

$$\text{Where } H(s) = \begin{cases} 1 \text{ if } s > 0 \\ 0 \text{ if } s \leq 0 \end{cases}$$

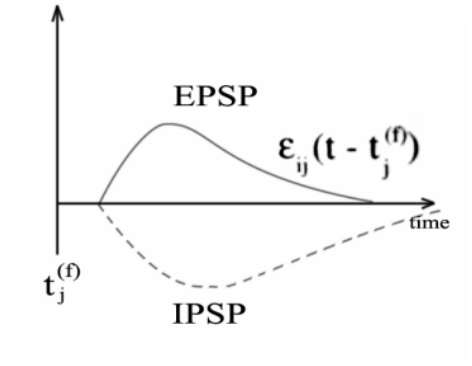
And Δ^{ij} denotes the transmission delay, $0 < \tau_s < \tau_m$ are constants that define the duration of the effects of the PSP and δ^{abs} is the duration of the absolute refractoriness (the positive spike scaled by K).

LIF model

The more popular neuron model for SNNs is the leaky integrate and fire (LIF) [23] model for its simplicity, which is a modification of the SRM. This model is governed by Equation (5.6), which describes the state of a circuit, where a capacitor C that is in parallel with a resistor R is driven by an input current I ($\tau_m = RC$ is a time constant with which voltage decays over time). Unlike with the SRM model, the action potential is not explicitly modelled, but rather defined by the firing time $t^{(f)}$, again when $u(t^{(f)}) = \vartheta$, where the membrane potential simply set to the resting potential u_r instead of a SAP: essentially, the model 'leakily' integrates all the incoming PSPs, releases a spike when crossing the threshold and resets the membrane potential to the resting potential. Integration of Equation (5.6) gives a similar form to Equation (5.1).



(a) Example of the effect of the action potential on the membrane potential, η



(b) Example of an excitatory and inhibitory PSP, ε

Figure 5.1: Membrane response to action potential and PSP. Adapted from [23]

$$\tau_m \frac{du}{dt} = -u(t) + RI(t) \tag{5.6}$$

Aside from these two threshold dynamics models, Hodgkin-Huxley's and Izhikevich neuron models also exist, however the former is computationally expensive and the latter is a less common method, which is governed by two ordinary differential equations as given in Equation (5.7), where v is the membrane potential, and a, b, c and d are tuning parameters to create different spike patterns. u represents a recovery variable [29].

$$\begin{aligned} \dot{v} &= 0.04v^2 + 5v + 140 - u + I \\ \dot{u} &= a(bv - u) \end{aligned} \tag{5.7}$$

Where

$$\text{if } v \geq v_t, \text{ then } \begin{cases} v = c \\ u = u + d \end{cases}$$

5.2. Encoding Schemes

Furthermore, there are various methods to encode and decode information, and normally the same method is used for both. Specifically, the input to the neural network must be converted from their analogue/digital value into spike trains, and vice versa to produce a meaningful output. In the past, it was thought that all information was encoded using **rate coding**, where the frequency of spikes indicates the intensity of the signal. For decoding the spike trains, the spikes can be averaged over a sufficiently large observation window instead of specific spike sequences due to the variable nature of a SNN, which is typically between $100 - 500ms$ [3, 23], depending on the neuron models as well. Alternatively they can be averaged over multiple runs or populations, although that further limits the speed of the network.

For encoding information into spike trains, the signal can be directly converted into firing rates with additive noise, or probabilistic firing rates can be used, where the probability that a neuron fires is dependent on the strength of the input signal (such as using a Poisson distribution). Although rate coding is more resistant to noise, it is much slower due to this observation window, and hence is also not considered to by biologically realistic (although the peripheral nervous system does exhibit

rate coding behaviour). Rate coding can also be used to convert conventional ANNs into SNNs [44], as increasing firing rates is analogous to a higher output value of a neuron.

The other type of encoding is **temporal coding** (also known as latency coding or time-to-first-spike coding) [31], with the idea that the precise timing of the spikes also carries information, and is now thought to play a major role in cognitive processing, due to the fact that functions of the brain are faster and more precise that rate coding would allow. Information is carried much faster and efficiently, and inputs are processed in much shorter time scales.

Rank-order encoding is one such example. Here each neuron fires at most once, and information intensity of the signal is represented by the delay of the spikes. This means that where the most significant information is held in earlier arriving spikes. This is especially useful when considering the output of the network where the first spike to arrive already gives a very good indication of the output, meaning the propagation cycle can be halted prematurely, resulting in faster computation speeds. This advantage is even more apparent when using neuromorphic hardware handling asynchronous data, where precise timing of the spikes is of the essence.

Another temporal encoding scheme is threshold-based encoding (or temporal contrast) where a spike is released only when the change of the input occurs beyond a certain threshold. Hough Spiker Algorith (HSA) and Ben's spiker algorithm (BSA) are encoding schemes where the stimulus is estimated by applying a linear filter to the spike train (finite impulse response). In Equation (5.8), which gives the estimation of the signal, h is the filter applied (through convolution) to the spike train x, where spikes occur at $t = t_k$. With HSA, to encode the information, the process is reversed and tries to do a reverse convolution of the signal by comparing if a hypothetical spike would be higher or lower than the signal at each time step. BSA on the other hand uses two error metrics to decide if a spike should be generated [31, 47], showing a better SNR than HSA. BSA turns out to be more suited for high frequency signals and can only account for excitatory spikes, while temporal contrast is more suited for steady signals. Step forward encoding and moving window spike encodings further extend the algorithm to better construct the original signal and be more robust to noise, respectively [31].

$$s_{est} = (h * x)(t) = \int_{-\infty}^{+\infty} x(t-\tau)h(\tau)d\tau = \sum_{k=1}^{N} h(t-t_k) \tag{5.8}$$

5.3. Training SNNs

One of the major shortcomings of SNNs is the training. Due to their non-differentiable activation functions, standard back propagation is difficult. While there are handcrafted solutions for implementing a SNN [27, 42], learning is the preferred choice. Supervised back propagation can be accomplished using proxy functions in place of the derivative and error assignment [49] or SpikeProp (uses backpropagation on the errors based only on the membrane potentials), or alternatively conventional ANNs can be trained and then converted to SNNs, however usage of these methods is still limited.

Hebbian learning on the other hand is extensively used for unsupervised learning. Based on biological mechanisms, Hebbian learning strengthens synapses which fire in a correlated manner. Specifically, spike-timing dependent plasticity (STDP) is a type of Hebbian learning algorithm that alters the weight of a synapse when the pre- and post-synaptic neurons fire in a correlated manner through the use of long-term potentiation (LTP) that strengthens the connection when the firing sequence happens in the expected causal temporal order and long term depression (LTD) which

weakens the connection when the post-synaptic neuron fires before the pre-synaptic one [12, 31] based on the exact timing. STDP can be used to make an SNN react quicker to data it has seen before, and is sometimes used in combination with other learning methods.

Wang, Hou et al. 2008 [61] use a manually constructed rate-coded SNN with LIF neurons and unsupervised Hebbian learning to allow a robot equipped with ultrasonic sensors to avoid objects while roaming by modifying the weights of the synapses, and showed improved, faster, less erratic, and more consistent behaviour. Mahadevuni and Li 2017 [36] used STDP learning to train a SNN to decide what direction the robot should turn when confronted with a wall to get to its target as fast as possible. These controllers however are very simple input-output NN, and should actually be classified as reactive controllers, since they do not have a hidden layer. Cao, Cheng et al. 2015 [10] build upon their previous study ([61]) to create a 3-layer modular SNN controller (using the SRM and rate coding) to reach a goal, follow walls and avoid obstacles. While the obstacle avoid module is pre-trained (using Hebbian learning), the other connections are further adapted using Hebbian learning as well. Arena, P et al. 2010 [6] also use STDP conditioning learning on a SNN with Izhikevich neurons for navigation and obstacle avoidance aboard a two-wheeled robot, demonstrating the efficiency of using SNNs for learning the association between visual features and basic behaviours.

As discussed, there are solutions for supervised training of SNNs, however, due to the nature of the problem at hand, perhaps a more self-supervised approach will be more favourable, as it could find patterns and solutions in the data beyond human perception. For this, bio-inspired evolutionary algorithms (EA) can be used to train both conventional and spiking neural nets that try to mimic nature. EAs are popular in the field of robotics, often referred to as evolutionary robotics, as they can find elegant solutions and behaviours to the problem. Chapter 6 will give a more extensive review of this optimisation algorithm.

6

Evolutionary Optimisation Algorithms

Evolutionary algorithms (EA) are used to optimise a wide variety of problems, especially for SNNs as they are not gradient descent based. With SNNs, they are sometimes combined with STDP learning, and EAs are also commonly used to optimise conventional ANNs, especially when dealing with robotics (often referred to as evolutionary robotics).

An EA is a bio-inspired algorithm for optimising problems based on natural selection. Pertaining to NNs (often coined evolving connectionist systems), the algorithm encodes the parameters of an initial population of (random or pre-trained) networks into genomes (mostly a string of binary values). It then evaluates each NN and computes its effectiveness based on a specified fitness function, which determines the probability that a NN is selected for a number of breeding operations. These breeding operations determine whether (characteristics of) a particular NN genome are passed on to the next generation or not. These operations are also probabilistic and include elitism, which guarantees that the best performing NN is always copied into the next generation, replication, where a genome is directly copied into the next generation based on their probability of selection, crossover, which exchanging genes between selected genomes, and mutation, which randomly changes a gene in a selected genome. This ensures that the optimisation landscape is searched in a unique way to EAs.

6.1. Encoding and Issues of EAs

The genes (parameters of network) can be encoded in different ways[21]. Direct representation means that there is a one-to-one mapping of the values into genes (for example if the EA is to optimise the synapse weights, then every weight is encoded into the genome). This means that each genome needs to encode a parameter accurately enough, which can make the genome rather long. Adaptive encodings try and solve this by encoding the most significant part of the NN e.g. by first encoding higher level parameters until it converges to a satisfactory solution, at which point it starts to encode more low level parameters.

Additionally, direct encoding schemes have been shown to produce excellent results in small networks, however they do not always scale well with increased network size, causing competing conventions and premature convergence. Competing conventions occurs when different genotypes code for NN with similar structures or behaviours, reducing diversity and complexity of the solutions. Premature convergence occurs when the algorithm converges to a local minimum which can also cause a rapid decrease in population diversity. Some methods have been developed to combat these problems such as the SANE (symbiotic adaptive neuro-evolution, which evolves individual

neurons to cooperate) and neuro-evolution of augmenting topologies (NEAT)[54].

The latter one, NEAT, is quite a popular algorithm that can evolve the structure and weights of NN simultaneously by using a genetic representation that allows crossover between different topologies in a meaningful manner using historical marking. NEAT also protects new topological innovations that might not perform as well initially until the weights are further evolved by only comparing the fitness of networks with similar structure. The genome contains 2 sets: the node genes, which contain the node number and type of node (input, hidden, output), and the connection genes, which contain the node in and out connection, the weight, whether it is enabled or disabled and the innovation number. A mutation can create either a connection or a node that is placed in the middle of an existing connection. When the latter occurs, the old connection is disabled and 2 new connections and a new node are created in the genotype.

Conversely, crossover compares the 2 parent genes, determines which genes are shared and combines them into the offspring gene as shown in Figure 6.1 [54]. To protect innovation, a method known as speciation is adapted, where the population is divided into species that share similar topologies. This is determined if compatibility distance δ is below a certain threshold δ_t, which is a function of the number of excess genes E, the disjoint genes D (as displayed in Figure 6.1), and the average weight differences between the two genomes W. N is the number of genes in the genome to normalise, as given in Equation (6.1). Species cannot overlap and they are represented by a random member in the species. Furthermore, to prevent a particular species from dominating the entire population, fitness sharing is used, where the fitness is penalised for increased species size, as given in Equation (6.2), where the sharing function sh is equal to 0 if $\delta > \delta_t$ and vice versa [54].

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \bar{W} \tag{6.1}$$

$$f_i' = \frac{f_i}{\sum_{j=1}^{n} \mathrm{sh}(\delta(i,j))} \tag{6.2}$$

Using NEAT, it is less likely to encounter competing conventions and premature convergence as NEAT attempts to explore the optimisation landscape more (thus less likely to fall in local minima). The objective of the algorithm is to make a simple-as-possible network by starting out with the smallest network (direct input to output mapping) and building on that by adding new connections and neurons. Other than direct representations, there are many other methods to encode a genome, such as ones that encode a process that constructs a NN (such as HyperNEAT, an extension of NEAT) and implicit representation[21, 26]. An EA can optimise different parameters, including synapse weights, fitness function, connectivity between layers, general architecture, activation function thresholds, and learning rules (such as STDP).

There are several general issues concerning EAs when applied to robotics [16, 40, 50]. The first is that EAs require large populations and generations to converge well, and this can take a long time to train as this needs to be carried out in real life or in simulation (which can be done faster). Concerning MAVs, it is undesirable to train the networks in real life as the controller would cause a crash often. With simulations however, there is always a gap when performing the same tasks in real life due to noise and inaccuracies in the sensors and model dynamics. Many solutions exist that try to minimise this gap.

One such is to simulate many generations until convergence, and then use those results as the initialisation to train the networks in real life for several more generations, which closes the gap [50].

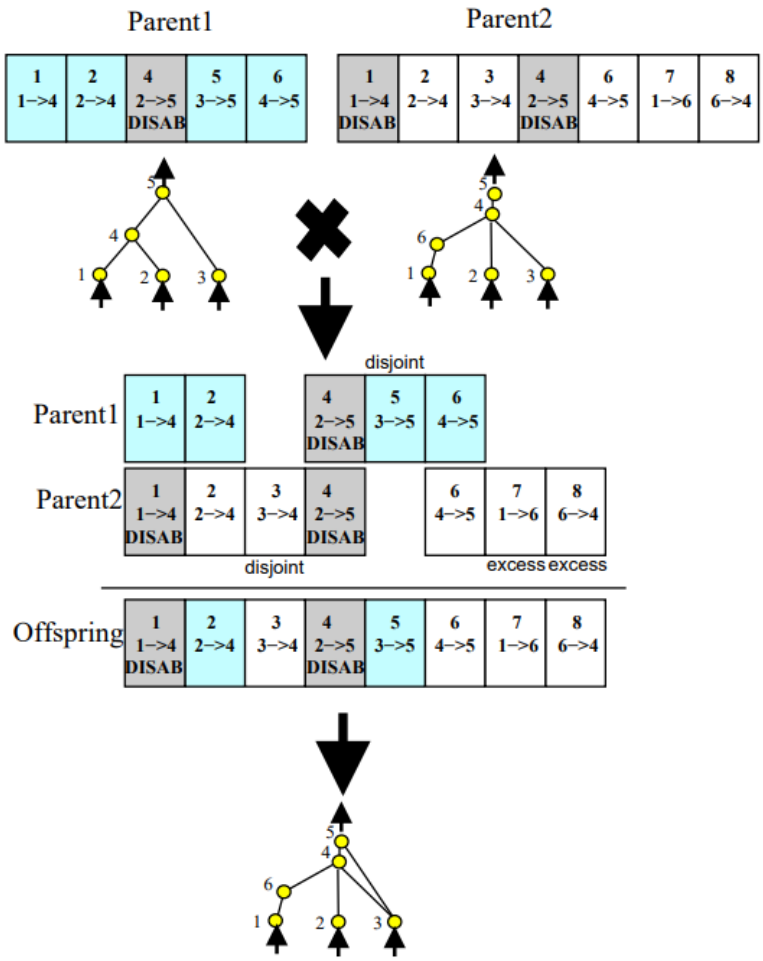


Figure 6.1: The crossover procedure in NEAT. The top number of the genes is the innovation number. Adapted from [54]

Alternatively, the gap can be minimised by periodically training the network in real life instead of just in simulation. Furthermore, it has been found that using real sensor data in simulation significantly improves both the performance and convergence when testing in real life, as was done in Silva, Correia et al. 2017 [51] using NEAT. It was also shown that using simulated noise in sensors may not always do better and might introduce an unwanted bias.

As mentioned before, EAs can also suffer from poor fitness functions, where they might not be entirely suitable such that initial evolution fails to produce desirable results (bootstrap problem) or that it fails to build a gradient towards the global optimum and instead converges towards a local optimum (deception). There are many solutions to these problems [50], examples of which include breaking down the task into simpler sub-tasks (or fitness functions), using human intervention to provide intermediate goals or continuously pushing for behaviour innovation rather than fitness [16].

6.2. EA applications in robotics

Conventional artificial neural networks

Floreano and Mondada 1994 [19] implement a 2-layer ANN controller using 8 ultrasonic sensors (8 inputs) to allow a miniature ground-based robot to avoid obstacles using the 2 wheels (2 output neurons). The EA evolved the weights of the ANN in real life rather than simulation by processing data on a host computer and learned to avoid obstacles in 100 generations. Miglino, Nafasi et al. 1994 [37] use optical sensors (optical flow) as input to a 3-layer RNN with a memory neuron to navigate towards the centre of an enclosure and avoid collision with walls. In this case the EA optimised the weights of the network in simulation and managed to avoid the walls in real life even if the paths were different. Nolfi, Miglino et al. 1994 [40] use an EA that evolves both the weights and thresholds of the connection and neurons in a 3-layer ANN to allow a robot equipped with IR sensors to navigate in a closed environment without colliding with walls. For the simulation, real recorded data was used as the sensory input to the network and was trained for 300 generations. When transferred to real life, it was allowed to train for an extra 30 generations, which closed the reality gap almost completely.

Duarte, Oliveira et al. 2015 [16] present a more recent study that discusses some of the general issues in evolutionary robotics such as the reality gap, the bootstrap problem and deception, and demonstrate a modular RNN which is based on the principle that if a task is too complex, it and the fitness function are broken down into simpler sub-tasks for which a separate 3-layer fully-connected RNN is evolved. The modular controller showed significant improvement. Silva, Correia et al. 2017 [51] use an online NEAT algorithm to quickly create a working controller in real life. It was also shown that pre-training the network in simulation using real sensor data performed the best, rather than randomly initialising or adding artificially generated noise in the simulation (which added a bias).

Spiking neural networks

SNNs can also make good use of EAs due to the limited learning capabilities. A notable example is Qiu et al. 2018 [45], which use the NEAT algorithm to train a SNN to control the pole-balancing problem, which is a behavioural, non-linear problem. For this, they compare a conventional RNN with a recurrent spiking network, with rate-coded probabilistic firing rates and Izhikevich neuron models, and show that SNNs actually converge faster than the conventional RNN. The recurrent connections allow for approximations of derivatives, and to avoid sparse spiking propagation through the network, a constant background current is applied.

More specific to controllers, Howard et al. [25] utilises a self-adaptive evolutionary algorithm to gen-

erate a robust low-level SNNc(LIF neurons) quadrotor neurocontroller for waypoint holding, while allowing for plasticity for in-trial training, and demonstrated more accurate performance and better robustness to different environments as apposed to PID control. Relating to more higher level control, Hagras, Pounds-Cornish et al. 2004 [24] presents a controller to perform obstacle avoidance of stationary objects using ultrasound sensors. The weights of a 2-layer SNN are trained using an adaptive EA with small populations, which changes the mutation and crossover probabilities to prevent premature convergence. This is also done as in this study learning is done online, which requires fast convergence in order to quickly produce a stable controller. Mitchell, Bruer et al. 2017 [39] also uses an EA to train a SNN controller on neuromorphic hardware for an autonomous, roaming, obstacle-avoiding ground robot using neuromorphic hardware and lidar single point sensors. The EA evolves the neuron thresholds, weights, and structure of the network using a method from Schuman, Disney et al. 2016 [48], and afterwards plasticity is allowed (STDP) to further optimise the network. This showed that the robot was able to avoid the walls of the enclosure while exploring the area. Alnajjar et al. 2005 [4] compare the use of EAs and self-organisation algorithms to train SNN for obstacle avoidance and navigation, demonstrating that while the latter is faster in training, the EA is able to elicit correct behaviours to perform the tasks.

Floreano, Zufferey et al. 2003 [20] explores the possibility of using an EA to train a 3-layer SNN (with rate coding and a SRM model) to avoid collision with walls using optical sensors (it essentially learns to equalise the optical flow on both sides of the vehicle). The EA evolves both the connections weights and signs between the inter-connected neurons and between the neurons and input (the structure is fixed, but the connectivity is evolved). The output of the network controls the speed of the 2 wheels. The study further tries to extend the controller for a blimp craft (which showed success) and a flapping-wing MAV, the latter of which was not implemented yet, mostly because an appropriate simulator was not available. The authors also speculated that transferring the SNN from simulation to real life would present a gap, and a possible solution would be to use some plasticity learning like STDP.

Liquid state machine

Another way to train SNNs is using Liquid State Machines (LSM), which are a class of recurrent SNNs where the input feeds into the hidden layers, which form a reservoir of neurons with recurrent connections. These connections are randomly initialised and remain fixed. The only connections that are trained are the output synapses, which greatly reduces the learning problem to a super fast supervised linear regression method. Although the system is sensitive to the initialisation of the randomised weights and might not always produce desired results, many different LSMs can be trained fast. The reservoir holds information about the state of a system, and the output samples from this state. Burgsteiner 2020 [8] uses a LSM to train a robot to mimic the behaviour of pre-recorded data from another NN controller with success. Burgsteiner, Kröll et al. 2007 [9] trains a LSM to predict the location of a moving object on a 6x8 grid screen some time steps ahead.

7

Analysis

After reviewing relevant literature, the aim is to develop methods that utilise a 24GHz FMCW fast-chirp radar to perform dynamic obstacle avoidance in a poor visibility environment. Furthermore, it might prove interesting to demonstrate that a simple neural network controller can be evolved to display more complex behaviour, and whether it can outperform the traditional method of tracking avoidance (such as velocity obstacles) in terms of reliability, speed and computational efficiency. Although few existing studies investigate the use of controllers on MAVs but rather simple ground-based robots, the controllers can be transferred from the ground to air as the degrees of motion are similar. This has also been demonstrated by [20].

Furthermore, most studies assume a static environment, thus rarely involving the use of radar instruments and preferring single point sensors such as Lidar or ultrasound, which produce simpler data. Avoiding moving obstacles and incorporating the velocity of the moving obstacles in the controller also presents a new level of complexity to the problem. This is the reason that evolutionary algorithms are explored, with the hope that it can search the optimisation landscape better and find a more optimum solution that could not be found manually. It is also the intent to reduce the computational load such that the methods can be applied to platforms that have inferior computational resources available, such as lightweight miniature drones or flapping wings.

In order to evaluate the proposed NN-based behavioural controller, a standard more established method should be constructed for comparison. This will involve selecting the appropriate controller, the primary choice being velocity obstacles as it also takes into account the velocity of the obstacles (which is also the intention for the behavioural controller). When using velocity obstacles, multi-target tracking will have to be done. A data association method has to be selected based on the motion of the obstacles, clutter present in the environment and the characteristics of the radar, such as the range, velocity and angular resolutions. Moreover, the radar parameters should also be tuned, such as bandwidth, chirp time and power to get the best characteristics and minimal noise or clutter. Since the objects remain relatively separated in the obstacle course, a simpler method such as GNN can be tested. Depending on the settings of the radar, and whether the clutter from the walls interfere with the procedure, a more complex method can be explored.

Furthermore, a tracking procedure also has to be selected for velocity obstacles. Again since the motion of the obstacles is relatively constant and linear, the ordinary KF can be applied to estimate the position, velocity and acceleration of the obstacles. Since there is also limited computational power to perform S&A in real time, both the data association and state estimation procedures should not be too complex. Regarding velocity obstacles, the avoidance manoeuvres and parameters should

also be tuned to account for the MAV dynamics and reaction speed, which will mostly depend on the update speed of the algorithms based on the computation time.

Concerning the NN-based behavioural controller, the appropriate type and structure has to be selected, for instance, what the input and output of the network will be and their dimensions, and whether or not spiking neurons will be used. The most direct approach is to feed the NN with the position and radial velocity of every detection directly from the radar with no pre-processing. Because of the variable possible detections at every time instant, the network could process every obstacle individually and then sum the outputs to generate meaningful avoidance commands. Alternatively, some filters can be applied, either to simply discard some clutter or apply the full MTT association and tracking procedure, with as input to the network the exact velocity of the obstacles (rather than the radial velocity) and estimated positions. Another option is to discretise the observation space into a 2D (polar) occupancy grid where the radial velocity could also be incorporated into the multi-dimensional grid, which can then be fed into the NN as a fixed dimensional input pixel by pixel, or even by using convolutions. A similar approach to the VFH algorithm in Section 4.1 can also be considered, where the observation space is discretised into a polar histogram, with this as input to the network to generate meaningful commands. This last one is perhaps more in line with existing studies, which use multiple single point sensors (which already represents a polar discretised data form). These options would however increase the number of nodes in the network and thus also increase computation time and training. The output of the network also has to be selected, which could simply be a stopping command, or alternatively turn rate/angle and velocity commands to the autopilot.

When this has been established, a suitable learning algorithm should be selected that optimises certain network parameters. When dealing with SNNs, the most attractive method ((in this case) is using evolutionary algorithms. For instance, NEAT optimises both the structure and weights of the network. If employed, an appropriate fitness function will have to be selected (i.e. what the network has to learn) in order to allow for better convergence and training speed. One option is to train the network to mimic human response to the moving obstacles by training it with data gathered from flying the MAV manually through the obstacle course. The fitness function would then indicate how similar the paths taken are between the human operator and controller. Alternatively, some other objectives that could constitute a fitness function include the time taken to traverse the obstacle course, (lateral) distance travelled, proximity to the obstacles (or collisions with obstacles) or the number of avoidance manoeuvres required. The other possibility is to use a LSM network, which can be trained much faster through linear regression. This solution is perhaps less controlled since most of the network is randomly selected.

The most important aspect of using evolutionary algorithms is ensuring that the simulation is of high enough fidelity to simulate the dynamics of the MAV and characteristics of the radar sensor, such as noise and clutter. This will involve collecting data of the radar in the environment and either analysing and simulating it, or using it as direct input when simulating. This has to be done in order to reduce the gap when transferring the controller to reality, while also being efficient such that training is fast enough.

Lastly, some metrics have to be defined to evaluate the controller against the standard method. This will include the computational power required (or cpu load), consistency of results with different starting configurations, the objectives defined in the fitness function and a qualitative analysis of the behaviour. Based on this confrontation, possible issues can be addressed and recommendations can be given.

8

Conclusion

This study has explored the latest knowledge and applications of sense and avoid (S&A) concerning autonomous navigation of a MAV in a dynamic environment with poor visibility, delving into the existing literature and technologies of object tracking and controllers.

Firstly it has been shown how radars function and sense the environment. Specifically, how fast-chirp FMCW radars are best suited for short range S&A applications, the processing involved and the data that the sensor produces. Following, the standard methods used to further process this data into useful information have been explored, including the standard data association algorithms, such as global nearest neighbour, joint probabilistic data association and multiple hypothesis tracking filters, and tracking algorithms such as the Kalman filter. This information can then be better used by the avoidance algorithms.

This study has explored the more established and deterministic methods of S&A controllers, such as velocity obstacles and vector field histograms, as well as looking into what behavioural controllers are and how they differ from the standard avoidance algorithms. This has shown that neural network based controllers can exhibit more complex behaviour while also being more robust to noise and possibly more computationally efficient, with the added advantage of their learning capabilities. Subsequently, it was explored how neural network based behavioural controllers can be constructed using spiking neural nets by investigating how spiking nets operate, and how they can be trained using Hebbian learning and evolutionary strategies, the latter of which is commonly applied to robotics.

Lastly, an analysis was done to determine which methods can be combined to design a system specific for this implementation using the knowledge gathered in this study. The main takeaway is to investigate whether a neural network based controller can outperform the standard pipeline and be more computationally efficient. Most studies that implement behavioural controllers however deal with ground based robots in a static environment. For this reason, evolutionary algorithms are considered to better search the optimisation landscape to find an optimum.

The biggest challenge to evolutionary algorithms is constructing a simulation for training, as this decreases the time needed. The simulation needs to approximate the drone dynamics and sensor characteristics to a high enough degree such that when the controller is transferred to real life, the difference in behaviour is minimal. Furthermore, it is important to consider what format the input and output should be of, which will influence the structure of the neural net and the complexity of the behaviour.

Next to this, an implementation using the standard methods of association, tracking and avoidance should be constructed as a means to compare and evaluate the proposed behavioural controller, which should not be overly complex given the nature of the problem and the computational constraints, as well as providing an equivalent control test. This involves constructing a standard pipeline consisting of FFT processing, GNN data association, Kalman Filtering and Velocity Obstacles. The intention is to use a simple controller that can exhibit more complex behaviour. Lastly, the evaluation will take place according to the appropriate metrics and criteria, which will be a combination of quantitative metrics, such as speed, distance and computational power, and qualitative analysis, as behaviour is more difficult to quantify.

Bibliography

[1] *Multitarget Tracking*, pages 1–15. doi: 10.1002/047134608x.W8275. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/047134608X.W8275.

[2] Introduction to mmwave sensing: Fmcw radars. URL https://training.ti.com/sites/default/files/docs/mmwaveSensing-FMCW-offlineviewing_4.pdf.

[3] Dr Ammar Almomani, Mohammad Alauthman, Mohammed Alweshah, Osama Dorgham, and Firas Albalas. A comparative study on spiking neural network encoding schema: implemented with cloud computing. *Cluster Computing*, 22, 2019. doi: 10.1007/s10586-018-02891-0.

[4] Fady Alnajjar and K. Murase. Self-organization of spiking neural network generating autonomous behavior in a miniature mobile robot. In Kazuyuki Murase, Kosuke Sekiyama, Tomohide Naniwa, Naoyuki Kubota, and Joaquin Sitte, editors, *Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE 2005)*, pages 255–260. Springer Berlin Heidelberg. ISBN 978-3-540-29344-6.

[5] Fady Alnajjar, Indra Bin Mohd Zin, and Kazuyuki Murase. A hierarchical autonomous robot controller for learning and memory: Adaptation in a dynamic environment. *Adaptive Behavior*, 17(3):179–196, 2009. ISSN 1059-7123. doi: 10.1177/1059712309105814. URL https://doi.org/10.1177/1059712309105814.

[6] P. Arena, S. De Fiore, L. Patané, M. Pollino, and C. Ventura. Insect inspired unsupervised learning for tactic and phobic behavior enhancement in a hybrid robot. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. ISBN 2161-4407. doi: 10.1109/IJCNN.2010.5596542.

[7] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991. ISSN 2374-958X. doi: 10.1109/70.88137.

[8] Harald Burgsteiner. Training networks of biological realistic spiking neurons for real-time robot control. 2020.

[9] Harald Burgsteiner, Mark Kröll, Alexander Leopold, and Gerald Steinbauer. Movement prediction from real-world images using a liquid state machine. *Applied Intelligence*, 26(2):99–109, 2007. ISSN 1573-7497. doi: 10.1007/s10489-006-0007-1. URL https://doi.org/10.1007/s10489-006-0007-1.

[10] Zhiqiang Cao, Long Cheng, Chao Zhou, Nong Gu, Xu Wang, and Min Tan. Spiking neural network-based target tracking control for autonomous mobile robots. *Neural Computing and Applications*, 26(8):1839–1847, 2015. ISSN 1433-3058. doi: 10.1007/s00521-015-1848-5. URL https://doi.org/10.1007/s00521-015-1848-5.

[11] Y. Chung, P. Chou, M. Yang, and H. Chen. Multiple-target tracking with competitive hopfield neural network based data association. *IEEE Transactions on Aerospace and Electronic Systems*, 43(3):1180–1188, 2007. ISSN 1557-9603. doi: 10.1109/TAES.2007.4383609.

[12] André Cyr and Mounir Boukadoum. Classical conditioning in different temporal constraints: an stdp learning rule for robots controlled by spiking neural networks. *Adaptive Behavior*, 20 (4):257–272, 2012. ISSN 1059-7123. doi: 10.1177/1059712312442231. URL https://doi.org/10.1177/1059712312442231.

[13] B. Damas and J. Santos-Victor. Avoiding moving obstacles: the forbidden velocity map. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4393–4398. ISBN 2153-0866. doi: 10.1109/IROS.2009.5354210.

[14] Hubertus Wilfridus de Waard. *A new approach to distributed data fusion*. Universiteit van Amsterdam [Host], 2008. ISBN 9090235027.

[15] Sebastian Thruny ArnoBücken WolframBurgard DieterFox and ThorstenFröhlinghaus Daniel-Hennig ThomasHofmann MichaelKrell TimoSchmidt. Map learning and high-speed navigation in rhino. 1998.

[16] Miguel Duarte, Sancho Moura Oliveira, and Anders Lyhne Christensen. Evolution of hybrid robotic controllers for complex tasks. *Journal of Intelligent & Robotic Systems*, 78(3-4):463–484, 2015. ISSN 0921-0296.

[17] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006. ISSN 1558-223X. doi: 10.1109/MRA.2006.1638022.

[18] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998. ISSN 0278-3649. doi: 10.1177/027836499801700706. URL https://doi.org/10.1177/027836499801700706.

[19] Dario Floreano and Francesco Mondada. Automatic creation of an autonomous agent. genetic evolution of a neural network driven robot, 1994. URL http://hdl.handle.net/20.500.11850/82611.

[20] Dario Floreano, J. C. Zufferey, and J. D. Nicoud. From wheels to wings with evolutionary spiking circuits. In Fernando Moura Pires and Salvador Abreu, editors, *Progress in Artificial Intelligence*, pages 3–3. Springer Berlin Heidelberg. ISBN 978-3-540-24580-3.

[21] Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008. ISSN 1864-5917. doi: 10.1007/s12065-007-0002-4. URL https://doi.org/10.1007/s12065-007-0002-4.

[22] Duh Fun-Bin and Lin Chin-Teng. Tracking a maneuvering target using neural fuzzy network. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):16–33, 2004. ISSN 1941-0492. doi: 10.1109/TSMCB.2003.810953.

[23] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002. ISBN 0521890799.

[24] H. Hagras, A. Pounds-Cornish, M. Colley, V. Callaghan, and G. Clarke. Evolving spiking neural network controllers for autonomous robots. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 5, pages 4620–4626 Vol.5. ISBN 1050-4729. doi: 10.1109/ROBOT.2004.1302446.

[25] David Howard and Alberto Elfes. Evolving spiking networks for turbulence-tolerant quadrotor control. In *Artificial Life Conference Proceedings 14*, pages 431–438. MIT Press. ISBN 0262326213.

[26] Benjamin Inden and Jürgen Jost. Evolving neural networks to follow trajectories of arbitrary complexity. *Neural Networks*, 116:224–236, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.04.013. URL http://www.sciencedirect.com/science/article/pii/S089360801930111X.

[27] Giacomo Indiveri and Paul Verschure. Autonomous vehicle guidance using analog vlsi neuromorphic sensors. In *International Conference on Artificial Neural Networks*, pages 811–816. Springer.

[28] Dan Iter, Jonathan Kuck, Philip Zhuang, and CM Learning. Target tracking with kalman filtering, knn and lstms, 2016.

[29] E. M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14 (6):1569–1572, 2003. ISSN 1941-0093. doi: 10.1109/TNN.2003.820440.

[30] W. Juszkiewicz. Target tracking with recurrent artificial neural network. In *2006 International Radar Symposium*, pages 1–4. ISBN 2155-5753. doi: 10.1109/IRS.2006.4338062.

[31] Nikola K Kasabov. *Time-space, spiking neural networks and brain-inspired artificial intelligence*. Springer, 2019. ISBN 3662577135.

[32] W. Kazimierski. Determining of marine radar target movement models for the needs of multiple model neural tracking filter. In *2011 12th International Radar Symposium (IRS)*, pages 611–616. ISBN 2155-5753.

[33] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4696–4704.

[34] Tarmo Koppel, Andrei Shishkin, Heldur Haldre, Nikolajs Toropovs, Inese Vilcane, and Piia Tint. Reflection and transmission properties of common construction materials at 2.4 ghz frequency. *Energy Procedia*, 113:158–165, 2017. ISSN 1876-6102. doi: https://doi.org/10.1016/j.egypro.2017.04.045. URL http://www.sciencedirect.com/science/article/pii/S1876610217321689.

[35] Ángel Llamazares, Eduardo J. Molinos, and Manuel Ocaña. Detection and tracking of moving obstacles (datmo): A review. *Robotica*, 38(5):761–774, 2020. ISSN 0263-5747. doi: 10.1017/S0263574719001024. URL https://www.cambridge.org/core/article/detection-and-tracking-of-moving-obstacles-datmo-a-review/BB94A95B06491BE09227A8BE7EDB7777.

[36] A. Mahadevuni and P. Li. Navigating mobile robots to target in near shortest time using reinforcement learning with spiking neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2243–2250. ISBN 2161-4407. doi: 10.1109/IJCNN.2017.7966127.

[37] Orazio Miglino, Kourosh Nafasi, and Charles E. Taylor. Selection for wandering behavior in a small robot. *Artificial Life*, 2(1):101–116, 1994. doi: 10.1162/artl.1994.2.1.101. URL https://www.mitpressjournals.org/doi/abs/10.1162/artl.1994.2.1.101.

[38] Vladimir Milovanovic. On fundamental operating principles and range-doppler estimation in monolithic frequency-modulated continuous-wave radar sensors. *Facta universitatis - series: Electronics and Energetics*, 31:547–570, 2018. doi: 10.2298/FUEE1804547M.

[39] J. P. Mitchell, G. Bruer, M. E. Dean, J. S. Plank, G. S. Rose, and C. D. Schuman. Neon: Neuromorphic control for autonomous robotic navigation. In *2017 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, pages 136–142. doi: 10.1109/IRIS.2017.8250111.

[40] Stefano Nolfi, Dario Floreano, Orazio Miglino, Francesco Mondada, Rodney Brooks, and Pattie Maes. How to evolve autonomous robots: Different approaches in evolutionary robotics. 1994.

[41] C. K. Oh and G. J. Barlow. Autonomous controller design for unmanned aerial vehicles using multi-objective genetic programming. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 2, pages 1538–1545 Vol.2. doi: 10.1109/CEC.2004.1331079.

[42] G. Orchard, R. Benosman, R. Etienne-Cummings, and N. V. Thakor. A spiking neural network architecture for visual motion estimation. In *2013 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 298–301. ISBN 2163-4025. doi: 10.1109/BioCAS.2013.6679698.

[43] A Pancham, N Tlale, and G Bright. Literature review of slam and datmo, 2011. URL `http://hdl.handle.net/10204/5457`.

[44] Michael Pfeiffer and Thomas Pfeil. Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, 12(774), 2018. ISSN 1662-453X. doi: 10.3389/fnins.2018.00774. URL `https://www.frontiersin.org/article/10.3389/fnins.2018.00774`.

[45] H. Qiu, M. Garratt, D. Howard, and S. Anavatti. Evolving spiking neural networks for nonlinear control problems. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1367–1373. doi: 10.1109/SSCI.2018.8628848.

[46] Karthik Ramasubramanian, Kishore Ramaiah, and Artem Aginskiy, 2017. URL `http://www.ti.com/lit/wp/spry312/spry312.pdf?ts=1588318620318`.

[47] B. Schrauwen and J. Van Campenhout. Bsa, a fast and accurate spike train encoding scheme. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 4, pages 2825–2830 vol.4. ISBN 1098-7576. doi: 10.1109/IJCNN.2003.1224019.

[48] C. D. Schuman, A. Disney, S. P. Singh, G. Bruer, J. P. Mitchell, A. Klibisz, and J. S. Plank. Parallel evolutionary optimization for neuromorphic network training. In *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*, pages 36–46. doi: 10.1109/MLHPC.2016.008.

[49] Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1412–1421. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/7415-slayer-spike-layer-error-reassignment-in-time.pdf`.

[50] Fernando Silva, Miguel Duarte, Luís Correia, Sancho Moura Oliveira, and Anders Lyhne Christensen. Open issues in evolutionary robotics. *Evolutionary Computation*, 24(2):205–236, 2016. doi: 10.1162/EVCO_a_00172%M26581015. URL `https://www.mitpressjournals.org/doi/abs/10.1162/EVCO_a_00172`.

[51] Fernando Silva, Luís Correia, and Anders Lyhne Christensen. Evolutionary online behaviour learning and adaptation in real robots. *Royal Society Open Science*, 4(7):160938, 2017. doi: doi:10.1098/rsos.160938. URL `https://royalsocietypublishing.org/doi/abs/10.1098/rsos.160938`.

[52] S. Silven. A neural approach to the assignment algorithm for multiple-target tracking. *IEEE Journal of Oceanic Engineering*, 17(4):326–332, 1992. ISSN 1558-1691. doi: 10.1109/48.180301.

[53] Héber Sobreira, Carlos M. Costa, Ivo Sousa, Luis Rocha, José Lima, P. C. M. A. Farias, Paulo Costa, and A. Paulo Moreira. Map-matching algorithms for robot self-localization: A comparison between perfect match, iterative closest point and normal distributions transform. *Journal of Intelligent & Robotic Systems*, 93(3):533–546, 2019. ISSN 1573-0409. doi: 10.1007/s10846-017-0765-5. URL https://doi.org/10.1007/s10846-017-0765-5.

[54] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002. doi: 10.1162/106365602320169811. URL https://www.mitpressjournals.org/doi/abs/10.1162/106365602320169811.

[55] A. Stateczny and W. Kazimierski. A comparison of the target tracking in marine navigational radars by means of grnn filter and numerical filter. In *2008 IEEE Radar Conference*, pages 1–4. ISBN 2375-5318. doi: 10.1109/RADAR.2008.4721044.

[56] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural Networks*, 111:47–63, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2018.12.002. URL http://www.sciencedirect.com/science/article/pii/S0893608018303332.

[57] Z. Tong, R. Renter, and M. Fujimoto. Fast chirp fmcw radar in automotive applications. In *IET International Radar Conference 2015*, pages 1–4. doi: 10.1049/cp.2015.1362.

[58] C. A. J. Tubb and G. N. Roberts. Development of a fuzzy behavioural controller for an autonomous vehicle. In *UKACC International Conference on Control '98 (Conf. Publ. No. 455)*, pages 1717–1722 vol.2. ISBN 0537-9989. doi: 10.1049/cp:19980488.

[59] Damien Vivet, Paul Checchin, Roland Chapuis, Patrice Faure, Raphaël Rouveure, and Marie-Odile Monod. A mobile ground-based radar sensor for detection and tracking of moving objects. *EURASIP Journal on Advances in Signal Processing*, 2012(1):45, 2012. ISSN 1687-6180. doi: 10.1186/1687-6180-2012-45. URL https://doi.org/10.1186/1687-6180-2012-45.

[60] Jilles Vreeken. Spiking neural networks, an introduction, 2003.

[61] Xiuqing Wang, Zeng-Guang Hou, Anmin Zou, Min Tan, and Long Cheng. A behavior controller based on spiking neural networks for mobile robots. *Neurocomputing*, 71(4):655–666, 2008. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2007.08.025. URL http://www.sciencedirect.com/science/article/pii/S0925231207003025.

[62] Andrzej Wojtkiewicz, Jacek Misiurewicz, Marek Nałęcz, Konrad Jedrzejewski, and Krzysztof Kulpa. *Two-dimensional Signal Processing in FMCW Radars*. 1997.

[63] Xiaofeng Yuan, Lin Li, and Yalin Wang. Nonlinear dynamic soft sensor modeling with supervised long short-term memory network. *IEEE Transactions on Industrial Informatics*, PP:1–1, 2019. doi: 10.1109/TII.2019.2902129.

A

Neural Network Implementation

This section will outline the work done with neural networks during this thesis that was not covered in the scientific paper or preliminary report, although not implemented in the final working result.

Neural Network implementation

While the standard processing pipeline was able to reliably process the raw data of the FMCW radar, determine the state of the environment and avoid the obstacles, an alternate method was explored using neural networks (NN) to process the data.

To better get an understanding of the part of the processing pipeline that needs to be learned by NNs, Figure A.1 gives this architecture, from the raw data of the radar, to the raw unfiltered object detections in the field of view, and the filtered detections after data association and Kalman filtering. Note that the calculation of the radial velocity (Doppler component) is not included in the figure and was also not considered in this implementation for simplicity.

It was decided to focus development of the NNs on the raw data processing and filtering with the hope of improving the quality of the filtering procedure, as this study is more an evaluation of the sensor itself rather than the avoidance capabilities, on which there is already quite some literature [19, 36, 37, 40, 61]. This will also allow for the future development of an end-to-end (spiking) NN pipeline.

Implementation

The intention was to construct an ordinary NN and train it using standard back-propagation and stochastic gradient descent (SGD) in a self-supervised manner. Thus training data was needed for

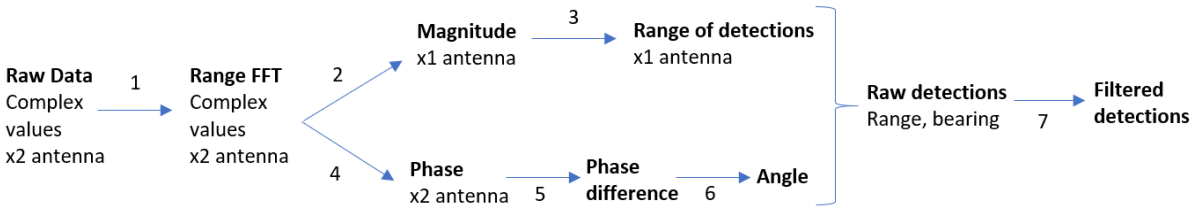


Figure A.1: Processing pipeline of the FMCW radar, beginning with performing the range FFT, determining the magnitude and ranges by thresholding (2 and 3), and comparing the phase difference between the two antennas to compute the angle (4,5 and 6). After this, data association and Kalman filtering remove the noise and estimate the states of the objects.

this purpose, for which the obstacle detection and avoidance dataset for drones was used (`https://github.com/tudelft/ODA_Dataset`). This dataset contains approximately 1000 usable flight runs of a MAV equipped with a FMCW radar. For each run, the pilot takes off and flies towards the centre of the flying arena, where one or two stationary poles are placed. Depending on the position and path of the MAV with respect to the pole, the pilot either chooses to avoid left, right, or fly straight forward. Each trial contains the ground truth position and orientation of the MAV and the pole, as well as the raw data recorded from the FMCW radar.

Implementation of the NN was done using Pytorch and its associated dataset class, wherein the raw data was loaded, processed and stored as the different steps in Figure A.1, as well as converting the global frame of reference ground truth positions to the MAVs local body frame of reference. The input consisted of 128 samples of **complex** values from both antennas (therefore 4x128 values, representing the raw data from one chirp. Other chirps were not considered as Doppler calculations were not taken into account), and for the output, initially two representations were considered. The first is a histogram representation, where each bin (or neuron) represents 1 degree of the FOV, and the magnitude of that bin will be equal to the inverse of the distance to a detection, such that closer objects represent a higher significance, and that the magnitude of 1-degree-bins with no objects present is equal to 0. The second representation chosen was to only represent the range and bearing (2 neurons) of object that was closest to the MAV.

Results

Initially, an attempt was made to train the network to learn the ground truth positions (filtered detections) from the raw data end-to-end using a standard multi-layer perceptron (MLP - a number of fully connected layers with ReLU activation functions). Different hyper parameters were varied to try and converge the network. This included changing the number of layers up to 10, the number of neurons up to 2048 neurons per layer, trying different common activation functions at the output such as ReLU, Softmax, Sigmoid and tanh, different optimisers such as SGD, adaptive SGD, ADAM and RMSprop algorithms, and general parameters such as the learning rate. However after quickly realising that the network would only converge to random values in the output, the problem was broken down to instead first determine the raw detections with one NN, and the filtered detections with another NN. The two NN would then later be combined and retrained.

Moreover, the first calculation step in the processing pipeline (to determine the range FFT) was done separately for two reasons: firstly the fast Fourier transform is in itself already a really efficient algorithm that scales computationally in the order of $O(N \cdot log(N))$, as opposed to a neural network that scales computationally with $O(N^2)$. Secondly, as the FFT is a linear function, a NN approximating this function will have linear activation functions, which defeats the purpose of the NN's implementation. When performing an FFT, the information is simply transformed into a different domain (frequency instead of time). Furthermore, after the FFT is performed (after step 1), only the first 20 values of the FFT are used as input to the NN to reduce dimensionality, which represents a maximum range of 7.5m as this study concerns mostly close proximity obstacle avoidance (lower frequency represents closer range, where each frequency bin is equal to $0.375m$. See Chapter 2).

The same procedure was applied for the first NN to determine the raw detections (range and bearing as calculated with the standard pipeline - not ground truth positions). By varying the hyper parameters, a NN with ReLU activation functions, 3 hidden layers, 512 neurons per layer converged to determine the range of any detections in the FOV (steps 2 and 3). Note that for this step, the input are the first 20 FFT data points (40 neurons total, for both real and imaginary values of one antenna), and the output are 20 neurons that each represent a range bin of $0.375m$ (for a maximum range of

7.5m), and the network would always predict the correct bins in which detections were present, else the neuron output would be zero for no detections. Concerning the bearing of detections (steps 4,5 and 6), initially the network was not able to converge. Instead, a NN was created to separately compute all the 20 angle values from the FFT data points individually instead of simultaneously all at one. This network has as input 4 neurons (the real and imaginary FFT values of both antennas, representing only one frequency value of the FFT), 4 hidden layers (ReLU) with 64 neurons each, and 1 output neuron representing the angle of that point. This network would then be run 20 times in parallel to determine all the angle values of the corresponding 20 range bins of the previous NN, with an accuracy of ± 0.023 radians.

Once this was established, a NN to learn the ground truth pole locations (of optitrack) from this output could be created (step 7). For this, the two output representations were used (the histogram representation and nearest object representation). Initially a standard MLP was implemented, however after varying the hyper parameters and failing to converge the network, this was discarded. Furthermore, the nature of the data should also be considered: each frame coming from the preprocessing of the raw data is temporally correlated. An object is expected to appear near its location in the previous frame. Therefore LSTM cells were added to the network (Figure 3.2), and the data was fed in sequentially. For this, again the various hyper parameters were varied, also including the number of hidden neurons in the LSTM cells and the number of cells stacked (up to 3 cells were tested). However after extensive testing with various configurations the network would fail to learn, either converging to random values or all zero output.

One reason for this failed convergence could be due to the representation. By nature, the histogram representation is very sparse, only showing a non zero value in the neuron when objects are in that particular one-degree angle bin, encouraging the NN to simply converge to low values and not providing sufficient gradient to converge towards the global minimum. The other representation (where the output is the range and bearing of the nearest object) lacks integrity as it discards any other objects and potential hazards. Instead, a more probabilistic based representation can be tested in which each bin contains the probability that it contains an object. Another reason could be that the radar data is very noisy by nature, and that the neural network struggles to distinguish the object detections from clutter.

Another solution could be to utilise evolutionary algorithms to better search the optimisation landscape and converge to the global minimum. Although more complex and lengthy to train, it might provide additional insight and better convergence. Alternatively, general regressional neural networks (GRNNs) have already been proven to be able to approximate estimators such as Kalman filters [30, 32, 55], perhaps providing a better architecture for learning. This could aid in creating a full end-to-end neural network pipeline to integrate with the avoidance algorithm and potentially convert into a spiking neural network for faster and more efficient computation.