NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF SIGNALS, CONTROL, AND ROBOTICS

Contribution to the Design and Implementation of an Autonomous Driving System for a Formula Student Driverless Car

DIPLOMA THESIS

of

Aris I. Koumplis

Supervisor : Costas Tzafestas
Associate Professor NTUA

Athens, February 2023

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF SIGNALS, CONTROL, AND ROBOTICS

Contribution to the Design and Implementation of an Autonomous Driving System for a Formula Student Driverless Car

DIPLOMA THESIS
of
Aris I. Koumplis

Supervisor : Costas Tzafestas
Associate Professor NTUA

Approved by the examination committee on 5th December, 2023.

....	....	....
Costas Tzafestas	Petros Maragos	Athanasios Rontogiannis
Associate Professor NTUA	Professor NTUA	Associate Professor NTUA

Athens, February 2023

....
Aris I. Koumplis
Graduate of Electrical and Computer Engineering NTUA

Περίληψη

Η παρούσα διπλωματική εργασία αποσκοπεί να συμβάλλει στην ανάπτυξη ενός συστήματος αυτόνομης οδήγησης αγωνιστικού μονοθέσιου οχήματος στο πλαίσιο του φοιτητικού διαγωνισμού Formula Student. Η κατηγορία αυτόνομης οδήγησης (Driverless) προστέθηκε στους φοιτητικούς διαγωνισμούς σχεδίασης αγωνιστικών μονοθεσίων Formula Student το 2017. Ακολούθως, το 2022, ο κυριότερος διαγωνισμός Formula Student, αυτός της Γερμανίας, εισήγαγε την απαίτηση από όλες τις συμμετέχουσες ομάδες να συμπεριλάβουν δυνατότητα αυτόνομης οδήγησης στα αγωνιστικά μονοθέσιά τους.

Η παρούσα εργασία περιγράφει τη σχεδίαση της συνολικής αρχιτεκτονικής, ενός πλήρους pipeline, για το πρώτο αυτόνομο αγωνιστικό μονοθέσιο της φοιτητικής ομάδας PROM Racing του ΕΜΠ. Επιπλέον, διερευνεί σε βάθος θέματα υλοποίησης και αξιολόγησης των μονάδων "Αντίληψης" (Perception) και "Ταυτόχρονης Χωροθέτησης-Χαρτογράφησης" (SLAM) και παρουσιάζει μία νέα προσέγγιση υπολογισμού απόστασης εμπόδια που δεν έχει χρησιμοποιηθεί από άλλη ομάδα Formula Student. Επιπροσθέτως, παρουσιάζει αρχικές δοκιμές εναλλακτικών μεθόδων "Σχεδιασμού Κίνησης" (Motion Planning) που μπορεί να αποτελέσουν τη βάση για μελλοντική έρευνα καθώς και για περαιτέρω ανάπτυξη του οχήματος.

Η ικανότητα του οχήματος να εκτιμήσει την απόστασή του από τα όρια της πίστας (όπως αυτά οριοθετούνται στην προκειμένη περίπτωση από συγκεκριμένους κώνους) εξετάζεται σε ένα σύνολο 486 φωτογραφιών με 2745 κώνους σε γνωστές αποστάσεις. Η ικανότητα της μονάδας SLAM να παράξει το χάρτη της πίστας αξιολογείται πάνω στο όχημα καθώς κινείται σε πίστες τύπου Formula Student. Τέλος, οι εναλλακτικοί Motion Planners που υλοποιήθηκαν δοκιμάζονται στον προσομοιωτή του οχήματος. Η νέα μέθοδος εκτίμησης αποστάσεων πετυχαίνει μείωση μέσου τετραγωνικού σφάλματος, επιτρέποντας την αξιόπιστη ενσωμάτωση και πληροφορίας σχετιζόμενης με κώνους ευρισκόμενους σε μεγαλύτερες αποστάσεις. Με τη χρήση αυτής της μεθόδου, η μονάδα SLAM σημειώνει recall και precision της τάξης του 96.5% και 100%, αντίστοιχα. Τέλος, παρουσιάζονται και αποτελέσματα προσομοιώσεων για τη μελέτη της επίδοσης των εναλλακτικών μεθόδων Motion Planning που δοκιμάσθηκαν, δείχνοντας τη δυνατότητα υιοθέτησης αγωνιστικής γραμμής από το μονοθέσιο με ταυτόχρονη αποφυγή εξόδου από την πίστα.

Keywords

Αυτόνομη οδήγηση Formula Student, Αγωνιστικό Μονοθέσιο, μονοφθαλμική εκτίμηση βάθους, YOLO, Φίλτρο Kalman, graph SLAM, Pure-Pursuit, MPC

Abstract

This diploma thesis aims at contributing towards the development of an autonomous race car driving system in the frames of Formula Student competitions. Formula Student constitutes a set of race car design competitions addressing university teams. The new Driverless class in Formula Student competitions was introduced in 2017. Subsequently, in 2022, the most prominent Formula Student competition, that of Germany, introduced the requirement for all competing race cars to have autonomous driving capabilities, with non-compliance resulting in significant point reductions.

This diploma thesis lays out the comprehensive conceptual design of the autonomous system pipeline for the first Driverless race car of the NTUA PROM Racing Formula Student team. Furthermore, it explores in-depth issues related to the implementation and testing of the "Perception" and "Simultaneous Localization & Mapping" (SLAM) modules and proposes a new approach for detecting distance from obstacles which, to the best of the author's knowledge, has not been previously used by other teams in the Formula Student Driverless community. Finally, it describes the implementation and initial testing of different Motion Planners, which can constitute the basis for future research and further developments on the Driverless race car.

The goal of the Perception module is to detect the limits of the race track, which are in this case outlined by means of specific cones. The goal of the Perception module is, thus, to detect those cones, recognize their color and calculate the vehicle's distance to them. For testing the performance of this module, a dataset of 486 images containing 2745 cones at known distances is used. Furthermore, the performance of the SLAM module is evaluated based on its capacity to map a circuit and is tested on the actual vehicle while being driven on Formula Student style tracks. Finally, different motion planning approaches are implemented and tested on a simulator. The new perception approach achieves a reduction of Root Mean Squared Error (RMSE), enabling the reliable integration in the system of information associated with a larger set of cones situated at greater distances from the vehicle. Using this method, the SLAM module achieves a recall rate of 96.5% and a precision rate of 100% on the test tracks. Lastly, this thesis presents results from the implementation of different motion planning approaches tested in simulation, depicting the capacity of the vehicle to adopt a racing line while always remaining within the track limits.

Keywords

Driverless, Formula Student, Race car, monocular depth estimation, YOLO, Kalman filters, graph SLAM, Pure-Pursuit, MPC, knowledge distillation

Ευχαριστίες

Η ολοκλήρωση αυτής της διπλωματικής εργασίας σηματοδοτεί το πέρας της φοίτησής μου στην Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Ε.Μ.Π. Θα ήθελα να ευχαριστήσω τους ανθρώπους που συνέβαλαν σε αυτήν την πορεία.

Αρχικά, τον επιβλέποντα καθηγητή της εργασίας μου, τον κ. Κώστα Τζαφέστα. Σας ευχαριστώ για την εμπιστοσύνη που μου δείξατε και την ελευθερία που μου δώσατε να διαλέξω την κατεύθυνση της έρευνας.

Θα ήθελα επίσης να ευχαριστήσω την οικογένειά μου. Τους γονείς μου, Ιωάννη και Ειρήνη, που στηρίζουν κάθε βήμα μου και τον αδερφό μου, Αλέξανδρο, που βρίσκεται πάντα στο πλευρό μου.

Τέλος, θα ήθελα να ευχαριστήσω όλα τα μέλη της φοιτητικής ομάδας PROM Racing με τα οποία είχα την τύχη να συνεργαστώ στα δύο μου χρόνια στην ομάδα και ιδιαιτέρως το Δημήτρη, το Νίκο, το Μάνο, το Βασίλη και τον Πάρη.

Contents

List of Figures

Chapter 1

Εκτενής περίληψη

1.1 Εισαγωγή

Η παρούσα εργασία περιγράφει τη σχεδίαση ενός πλήρους pipeline για το πρώτο αυτόνομο αγωνιστικό μονοθέσιο της φοιτητικής ομάδας PROM Racing. Επιπλέον, διερευνεί σε βάθος θέματα υλοποίησης και αξιολόγησης των τμημάτων Perception και SLAM και παρουσιάζει μία νέα προσέγγιση υπολογισμού απόστασης από εμπόδια που δεν έχει χρησιμοποιηθεί από άλλη ομάδα Formula Student. Επιπροσθέτως, έχουν συμπεριληφθεί δοκιμές εναλλακτικών Motion Planners που μπορεί να αποτελέσουν τη βάση για μελλοντική έρευνα.

Το 2022, ο κυριότερος διαγωνισμός Formula Student, αυτός της Γερμανίας, ξεκίνησε να απαιτεί από όλες τις συμμετέχουσες ομάδες να συμπεριλάβουν δυνατότητα αυτόνομης οδήγησης στα αγωνιστικά μονοθέσιά τους. Η εγκατάσταση ενός αυτόνομου συστήματος έγινε αναγκαιότητα και αυτό αποτέλεσε το έναυσμα για την παρούσα εργασία.

Ο στόχος ήταν να σχεδιαστεί και να υλοποιηθεί ένα αυτόνομο σύστημα που θα μεγιστοποιούσε την πιθανότητα του οχήματος να ολοκληρώσει όλα τα δυναμικά αγωνίσματα των διαγωνισμών Formula Student Driverless. Για αυτό τον σκοπό, έχει δοθεί έμφαση στην απλότητα και στην αξιοπιστία, με την επίδοση να έρχεται σε δεύτερη μοίρα. Επιπλέον, οι οικονομικές δυνατότητες της PROM Racing είναι περιορισμένες σε σύγκριση με τις περισσότερες ομάδες των διαγωνισμών, εφιστώντας απαραίτητη μία οικονομική προσέγγιση στις επιλογές αισθητήρων και υπολογιστικού συστήματος.

Μετά το πέρας της σεζόν του 2023 κατέστη εμφανές πως ο κύριος περιοριστικός παράγοντας του αυτόνομου συστήματος ήταν το Perception και συγκεκριμένα η δυσκολία που αντιμετώπιζε στο να εκτιμήσει την ακριβή απόστασή του από εμπόδια. Αυτό γέννησε την ανάγκη ανάπτυξης ενός νέου μοντέλου Perception με βελτιωμένες δυνατότητες εκτίμησης βάθους.

Ήδη από την παρθενική σεζόν των διαγωνισμών Formula Student Driverless, ομάδες δημοσίευσαν το ερευνητικό τους έργο. Το πιο καθοριστικό έργο είναι το [19], στο οποίο η ομάδα του ETH παρουσιάζει το πλήρες αυτόνομο σύστημα του αγωνιστικού τους μονοθεσίου. Η ίδια ομάδα έχει επίσης δημοσιεύσει άρθρα που εξετάζουν λεπτομερώς και εισάγουν περαιτέρω βελτιώσεις στα διάφορα στοιχεία που απαρτίζουν το προαναφερθέν σύστημα. Αυτές οι εργασίες περιλαμβάνουν έναν Velocity Estimator που χρησιμοποιεί στοιχεία μηχανικής μάθησης[35], ένα υβριδικό EKF-Graph SLAM για χαρτογράφηση και εντοπισμό θέσης του μονοθεσίου στην πίστα[3], έναν ιεραρχικό Motion Planner[40], και έναν MPC που χρησιμοποιεί στοιχεία μηχανικής μάθησης[18]. Στο [36] παρουσιάζεται μία προσέγγιση του προβλήματος του Perception με μία κάμερα που αναπτύχθηκε από μια ομάδα αποτελούμενη από φοιτητές του MIT και του Delft. Τέλος, στο [25] γίνεται σύγκριση μεταξύ EKF SLAM και GraphSLAM στο αγωνιστικό μονοθέσιο του KIT.

1.2 Γενικές πληροφορίες

Οι διαγωνισμοί Formula Student αποτελούν διαγωνισμούς σχεδιασμού για φοιτητές που οργανώνονται από το Διεθνή Σύνδεσμο Μηχανολόγων Μηχανικών (SAE) από το 1980. Ομάδες φοιτητών καλούνται να σχεδιάσουν και να κατασκευάσουν αγωνιστικά μονοθέσια με τα οποία εν τέλει διαγωνίζονται στο τέλος κάθε σεζόν. Οι διαγωνισμοί χωρίζονται σε τρεις βασικές κατηγορίες: Οχήματα Εσωτερικής Καύσης (CV) (συμπεριλαμβανομένων Υβριδικών Οχημάτων (HY)), Ηλεκτρικά Οχήματα (EV) και Αυτόνομα Οχήματα (DV). Στην κατηγορία DV τα οχήματα αγωνίζονται τόσο σε γνωστές όσο και σε άγνωστες πίστες.

Η PROM Racing NTUA είναι μια ομάδα Formula Student που εκπροσωπεί το Εθνικό Μετσόβιο Πολυτεχνείο. Ιδρύθηκε το 2008 και αγωνίστηκε για πρώτη φορά στην κατηγορία CV το 2011. Το 2021 πραγματοποίησε τη μετάβαση στην EV κατηγορία, όπου έγινε η πρώτη ελληνική ομάδα που κερδίζει επίσημο διαγωνισμό Formula Student. Έπειτα από αυτή την επιτυχία, ξεκίνησε ένα διετές πλάνο για να αγωνιστεί το 2023 στην κατηγορία DV.

Το P23 είναι το αγωνιστικό μονοθέσιο της PROM Racing για τη σεζόν του 2023. Διαθέτει ηλεκτρικό κινητήρα 120 kW στον πίσω άξονα, ένα μετατροπέα για τον έλεγχο του κινητήρα και μια μπαταρία 9.1 kWh για την τροφοδοσία του συστήματος. Η ρύθμιση της μετάδοσης της κίνησης στον πίσω άξονα επιτυγχάνεται μέσω ενός διαφορικού περιορισμένης ολίσθησης. Το αεροδυναμικό πακέτο πετυχαίνει συντελεστή lift -7.3 με τους ανεμιστήρες ground effect και -5.5 χωρίς αυτούς, καθώς και λόγο lift προς drag -3.48 και 3.05, αντίστοιχα. Τέλος, χρησιμοποιούνται ελαστικά Hoosier R20 16.0x7.5 και στους δύο άξονες.

Το αυτόνομο σύστημα του P23 αποτελείται από τους μηχανισμούς του τιμονιού και φρένου, τους αισθητήρες κίνησης και αντίληψης, καθώς και το υπολογιστικό σύστημα. Οι αισθητήρες κίνησης του P23 περιλαμβάνουν έναν IMU, ένα GPS/INS και τέσσερις αισθητήρες ταχύτητας των τροχών. Το P23 διαθέτει δύο κάμερες που λειτουργούν ανεξάρτητα η μία από την άλλη, σημειώνοντας ένα συνολικό πεδίο θέασης της τάξης των $105^{\underline{o}}$. Τέλος, το υπολογιστικό σύστημα απαρτίζεται από έναν επεξεργαστή Intel i7-12700, έναν επιταχυντή TPU, μία RAM των 32GB και μία SSD του 1TB.

Το αυτόνομο σύστημα του P23 αποτελείται από τέσσερα συστατικά στοιχεία: την "Αντίληψη" (Perception), τον "Εκτιμητή Ταχύτητας" (Velocity Estimation), την "Ταυτόχρονη Χωροθέτηση-Χαρτογράφηση" (SLAM) και το "Σχεδιασμό Κίνησης" (Motion Planning). Το Perception είναι υπεύθυνο για την αναγνώριση των κώνων που λειτουργούν ως όρια της πίστας και την εκτίμηση του χρώματος και της απόστασής τους από το όχημα. Το Velocity Estimation είναι υπεύθυνο για την εκτίμηση των ταχυτήτων και των επιταχύνσεων του οχήματος. Σε γνωστές πίστες, το SLAM ασχολείται μόνο με τον εντοπισμό του σε αυτές, ενώ σε άγνωστες πίστες, επιφορτίζεται επιπλέον και με τη δημιουργία του χάρτη τους. Τέλος, το Motion Planning υπολογίζει την τροχιά που πρέπει να ακολουθήσει το όχημα καθώς και τις αντίστοιχες εντολές που πρέπει να σταλούν στους μηχανισμούς του τιμονιού, του φρένου και του γκαζιού. Η ροή των πληροφοριών στο σύστημα παρουσιάζεται στο σχήμα 1.1.

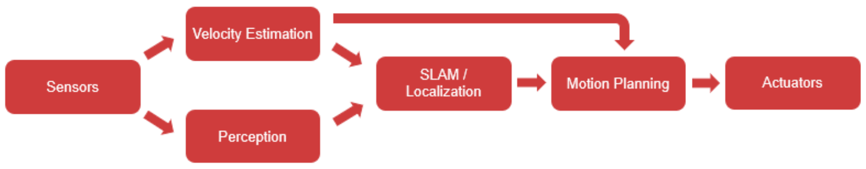


Figure 1.1: Το πλήρες αυτόνομο σύστημα του P23.

1.3 Εκτίμητης Ταχύτητας

Ο στόχος του Velocity Estimation είναι να συγχωνεύσει πληροφορίες από όλους τους αισθητήρες κίνησης προκειμένου να παρέχει συνεχώς μια εκτίμηση της ταχύτητας του οχήματος (u_x, u_y, ω). Ένα βασικό εργαλείο για την επίτευξη αυτού του σκοπού είναι το φίλτρο Kalman. Για τις περιπτώσεις, ωστόσο, που το σύστημα δεν είναι γραμμικό χρησιμοποιείται το Extended Kalman φίλτρο. Για τη συγκεκριμένη εφαρμογή οι εξισώσεις που περιγράφουν την εξέλιξη του συστήματος είναι:

$$\dot{u}_x = a_x + \omega * u_y + w_{u_x}$$

$$\dot{u}_y = a_y - \omega * u_x + w_{u_y}$$

$$\dot{\omega}_z = \alpha_z + w_{\omega_z}$$

$$\dot{a}_x = w_{a_x}$$

$$\dot{a}_y = w_{a_y}$$

$$\dot{\alpha}_z = w_{\alpha_z}$$

Αντίστοιχα, οι εξισώσεις που περιγράφουν τις προβλεπόμενες τιμές από τους αισθητήρες είναι:

$$z_{\omega_{front-wheel}} = \frac{u_x * cos(\delta) + (u_y + \omega_\theta * r_f) * sin(\delta)}{R_{wheel}} + v_{\omega_{front-wheel}}$$

$$z_{\omega_{rear-wheel}} = \frac{u_x}{R_{wheel}} + v_{\omega_{rear-wheel}}$$

$$z_a = Rot(a + \omega \times (\omega \times r) + \alpha \times r) - g + v_a$$

$$z_\omega = Rot(\omega) + v_\omega$$

$$z_u = Rot(u + \omega \times r) + v_u$$

Ο πίνακας συνδιακύμανσης για τις μετρήσεις της IMU προήλθε από το εγχειρίδιο του αισθητήρα. Το GPS/INS τρέχει εσωτερικά ένα INS Kalman φίλτρο και παρέχει το συνοδευόμενο πίνακα συνδιακύμανσης. Τέλος, για τους αισθητήρες μέτρησης ταχύτητας των τροχών και για το μοντέλο του συστήματος καθορίστηκαν βάση δοκιμών.

1.4 Αντίληψη

Ο στόχος του Perception είναι η ανίχνευση των κώνων, η αναγνώριση του χρώματός τους και ο υπολογισμός της απόστασης του οχήματος από αυτούς. Στο πλαίσιο αυτής της εργασίας διαμορφώθηκαν τρεις εναλλακτικές προσεγγίσεις.

Η πρώτη προσέγγιση βασίζεται στο γεγονός ότι οι ακριβείς διαστάσεις των κώνων είναι γνωστές εκ των προτέρων. Συγκεκριμένα, πραγματοποιείται η υπόθεση πως οι διαστάσεις του κώνου στη φωτογραφία αρκούν για να εκτιμηθεί η απόσταση της κάμερας από αυτόν. Για τον εντοπισμό του κώνου στην φωτογραφία χρησιμοποιείται το νευρωνικό δίκτυο YOLOv5n, το οποίο σημειώνει ορθογώνια παραλληλόγραμμα που περιβάλλουν κάθε κώνο στη φωτογραφία. Με χρήση του μοντέλου οπής για την κάμερα, σχηματίζονται δύο, κατά προσέγγιση, όμοιες πυραμίδες όπως φαίνεται και στο σχήμα 1.2. Η απόσταση του κώνου από την κάμερα είναι ίση με το μήκος του ευθυγράμμου τμήματος που ενώνει την κορυφή της πυραμίδας με το κέντρο της βάσης του κώνου, το οποίο υπολογίζεται γεωμετρικά.

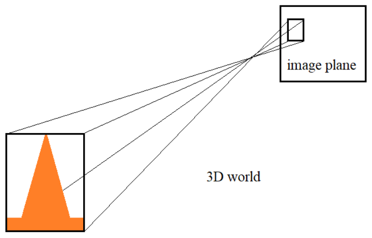


Figure 1.2: Οι δύο, κατά προσέγγιση, όμοιες πυραμίδες που σχηματίζονται.

Η δεύτερη προσέγγιση περιλαμβάνει τον εντοπισμό επτά χαρακτηριστικών σημείων πάνω στον κώνο. Βρίσκοντας τη θέση τους πάνω στην φωτογραφία, γνωρίζοντας τη σχετική τους θέση στον τρισδιάστατο χώρο από το κέντρο της βάσης του κώνου και έχοντας τις παραμέτρους της κάμερας και του φακού καλύπτονται όλες οι προϋποθέσεις για τη διατύπωση ενός προβλήματος Perspective-n-Point. Συγκεκριμένα, το κέντρο της βάσης του κώνου ορίζεται ως το κέντρο του κόσμου με σκοπό η επίλυση του προβλήματος Perspective-n-Point να δώσει την απόσταση μεταξύ κάμερας και κώνου. Για τον εντοπισμό των επτά χαρακτηριστικών σημείων χρησιμοποιείται ένα νευρωνικό δίκτυο βασισμένο στην αρχιτεκτονική ResNet.

Η τρίτη προσέγγιση χτίζει πάνω στην δεύτερη. Συγκεκριμένα χρησιμοποιεί το σφάλμα επαναπροβολής των επτά σημείων πάνω στην εικόνα για να αποφανθεί κατά πόσο έχει προκύψει σφάλμα κατά τον εντοπισμό τους. Αν το σφάλμα επαναπροβολής ξεπερνάει ένα κατώφλι τότε το πρόβλημα Perspective-n-Point λύνεται ξανά, αυτήν τη φορά για λιγότερα σημεία.

Στο διάγραμμα 1.3 αποτυπώνονται οι ρίζες των μέσων τετραγωνικών σφαλμάτων των τριών μεθόδων για κώνους που βρίσκονται σε διάφορες αποστάσεις από την κάμερα.

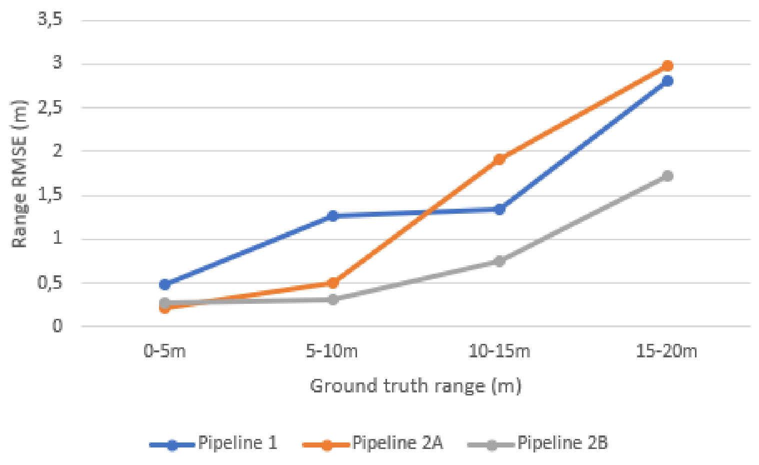


Figure 1.3: Οι ρίζες των μέσων τετραγωνικών σφαλμάτων των τριών μεθόδων. Η προσέγγιση των επτά σημείων πετυχαίνει καλύτερα αποτελέσματα από την προσέγγιση των όμοιων πυραμίδων μόνο για κοντινότερους κώνους. Η προσέγγιση του σφάλματος επαναπροβολής σημειώνει τα καλύτερα αποτελέσματα γενικά.

1.5 Ταυτόχρονη Χωροθέτηση-Χαρτογράφηση

Ο στόχος του SLAM είναι να κατασκευάσει το χάρτη της πίστας στην οποία τρέχει το όχημα, όταν αυτός δεν είναι διαθέσιμος, καθώς και να παρέχει συνεχείς εκτιμήσεις της θέσης του οχήματος σε αυτόν. Οι πίστες Formula Student είναι σχετικά επίπεδες και τα μόνα αντικείμενα που υπάρχουν σε αυτές είναι οι κώνοι που υποδεικνύουν τα όρια. Συνεπώς, χρησιμοποιείται ένας δισδιάστατος χάρτης, με τους κώνους απεικονιζόμενους ως σημεία στο επίπεδο. Για την επίλυση αυτού του είδους προβλημάτων, υπάρχουν δύο κύριες μέθοδοι: το filter και το graph SLAM[1].

Στην προκειμένη περίπτωση χρησιμοποιείται ο αλγόριθμος iSAM2[20], που ανήκει στην κατηγορία των αλγόριθμων graph SLAM. Ο αλγόριθμος iSAM2 χρησιμοποιεί τη δομή factor graph για να περιγράψει το χάρτη της πίστας. Τα factor graphs είναι διμερή γράφοι με δύο είδη κορυφών: τα variable nodes που απαρτίζονται από τις θέσεις των κώνων και του οχήματος και τα factor nodes που περιγράφουν τις σχέσεις μεταξύ των variable nodes, όπως αυτές προκύπτουν από το Perception και το Velocity Estimation.

Το SLAM λαμβάνει εκτιμήσεις ταχύτητας (u_x, u_y, ω) μαζί με τον αντίστοιχο πίνακα συνδιακύμανσης με συχνότητα 50Hz. Αντίστοιχα, λαμβάνει μια λίστα από παρατηρηθέντες κώνους (χρώμα, απόσταση, γωνία) με συχνότητα 10Hz. Ο πίνακας συνδιακύμανσης σε αυτή την περίπτωση έχει εκτιμηθεί βάσει πειραματικών δεδομένων και είναι συνάρτηση της απόστασης του κώνου από το όχημα.

Στην περίπτωση που η πίστα δεν είναι γνωστή εκ των προτέρων, το factor graph περιλαμβάνει δύο ειδών factor nodes: factor nodes οδομετρίας και factor nodes αντίληψης. Εδώ αξίζει να σημειωθεί πως το Velocity Estimation παρέχει εκτίμηση ταχύτητας και όχι μετατόπισης. Η μετάβαση από την ταχύτητα στη μετατόπιση πραγματοποιείται με τη μέθοδο Euler, η οποία είναι αρκετά ακριβής αφού το βήμα ολοκλήρωσης είναι ίσο με μόλις 1/50 του δευτερολέπτου. Δεδομένου ότι η υλοποίηση του iSAM2 στη βιβλιοθήκη gtsam απαιτεί από τη μετατόπιση να είναι εκφρασμένη στις συντεταγμένες του οχήματος, η προκύπτουσα μετατόπιση δίνεται από τον τύπο: $(u_x dt, u_y dt, \omega dt)$. Αυτό σημαίνει πως ο πίνακας συνδιακύμανσης που παρέχεται από το Velocity Estimation πρέπει να μετασχηματιστεί σε: $(dt * I)^T P_t (dt * I)$.

Στην περίπτωση που η πίστα είναι γνωστή εκ των προτέρων, οι θέσεις των κώνων στο χάρτη πρέπει να παραμένουν αμετάβλητες. Με άλλα λόγια, τα factor nodes αντίληψης πρέπει να είναι συναρτήσεις των θέσεων του οχήματος, αλλά όχι των θέσεων των κώνων. Για αυτό το σκοπό σχεδιάστηκε μία ειδική κλάση factor node με τα εργαλεία που παρέχει η βιβλιοθήκη gtsam.

Για τη συσχέτιση μεταξύ παρατηρηθέντων κώνων με κώνους που υπάρχουν ήδη στον χάρτη χρησιμοποιείται ο αλγόριθμος Nearest Neighbor (NN). Αν ο κοντινότερος κώνος του χάρτη είναι σε απόσταση μεγαλύτερη του 1.5 μέτρου από τον παρατηρηθέν κώνο τότε εισάγεται νέος κώνος στον χάρτη στη θέση του παρατηρηθέντος.

Σε μία ξεχωριστή δομή δεδομένων διατηρούνται στοιχεία για κάθε κώνο του χάρτη, όπως το χρώμα του και το πόσες φορές έχει εντοπιστεί. Για να αποφευχθεί η αποστολή χάρτη που να περιέχει αμφίβολους κώνους στο Motion Planning, ο χάρτης που αποστέλλεται περιέχει αποκλειστικά κώνους που έχουν εντοπιστεί τουλάχιστον πέντε φορές.

Το κύριο μειονέκτημα του συγκεκριμένου SLAM είναι ότι ο υπολογιστικός φόρτος της ενημέρωσης του factor graph αυξάνεται με το χρόνο και την προσθήκη νέων μετρήσεων[25]. Εάν ολόκληρη η διεργασία του SLAM έτρεχε σε ένα μόνο νήμα, μια καθυστερημένη ενημέρωση θα μπορούσε να αποτρέψει το σύστημα από το να χρησιμοποιήσει νέες μετρήσεις για να εκτιμήσει την τελευταία του θέση στην πίστα αυξάνοντας τον κίνδυνο σφάλματος. Για να αντιμετωπιστεί αυτό, χρησιμοποιείται ένα δεύτερο νήμα αποκλειστικά για τις ενημερώσεις του factor graph.

Στις άγνωστες πίστες που δοκιμάστηκε, το SLAM κατασκεύασε χάρτες με 94.7% recall και 97.3% precision. Ένας από αυτούς τους χάρτες απεικονίζεται στο διάγραμμα 1.4.

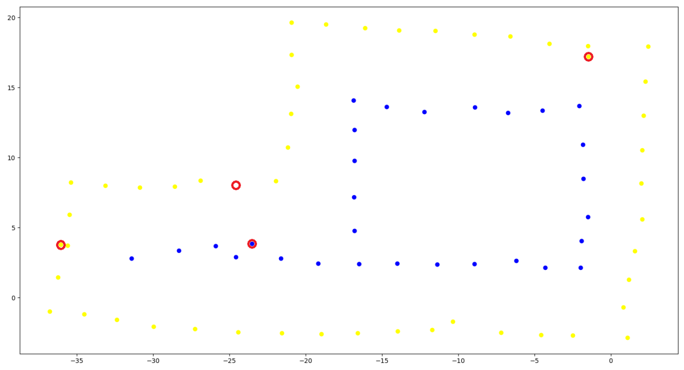


Figure 1.4: Ο χάρτης που προκύπτει από το SLAM. Λάθη στη χαρτογράφηση έχουν σημειωθεί με κόκκινους κύκλους.

1.6 Σχεδιασμός Κίνησης

Το Motion Planning είναι υπεύθυνο τόσο για τη χάραξη επιθυμητής τροχιάς όσο και για τον υπολογισμό των ενεργειών που θα κρατήσουν το όχημα πάνω σε αυτήν. Δεδομένου ότι οι πίστες Formula Student είναι στενές, η ακρίβεια των χαρτών που κατασκευάζει το SLAM χαμηλή, και η ποινή εξόδου από την πίστα υψηλή, το Motion Planning επικεντρώνεται στην εξαγωγή της κεντρικής γραμμής της πίστας έναντι μιας βελτιστοποιημένης τροχιάς. Έπειτα, υπολογίζεται βάση της δυναμικής του οχήματος το προφίλ ταχυτήτων για την κεντρική γραμμή. Τέλος, χρησιμοποιείται ένα ζεύγος ελεγκτών PID και Pure Pursuit που στοχεύουν στο να διατηρήσουν το όχημα πάνω στην επιθυμητή τροχιά.

Για την εξαγωγή της κεντρικής γραμμής, ο δισδιάστατος χώρος αρχικά διακριτοποιείται με τη χρήση της τριγωνοποίησης Delaunay, όπου οι θέσεις των κώνων λειτουργούν ως κορυφές των τριγώνων. Όλα τα μέσα των πλευρών των τριγώνων που προκύπτουν αντιμετωπίζονται ως πιθανά σημεία της κεντρικής γραμμής. Για να καθοριστεί ποια από αυτά πράγματι είναι, σχεδιάζεται ένας γράφος με κορυφές τα υποψήφια σημεία και τη θέση του οχήματος και ακμές που ενώνουν τις κορυφές που ανήκουν στο ίδιο τρίγωνο. Ένα δέντρο πιθανών μονοπατιών αναπτύσσεται κατά πλάτος σε αυτόν το γράφο με αφετηρία τη θέση του οχήματος. Ένα κλαδί του δέντρου παύει να αναπτύσσεται είτε όταν δεν υπάρχει υποψήφια κορυφή να προστεθεί σε αυτό, πράγμα που συμβαίνει στα όρια του κυρτού περιβλήματος της πίστας, είτε όταν έχει φτάσει σε ένα προκαθορισμένο βάθος. Τα μονοπάτια του δέντρου αξιολογούνται στο τέλος βάση της υπακοής τους στον χρωματικό κώδικα των κώνων και των μέγιστων κλίσεών τους για να προκύψει η κεντρική γραμμή. Τέλος, μία κυβική σπλίνα προσαρμόζεται πάνω σε αυτά τα σημεία.

Για την εξαγωγή του προφίλ ταχυτήτων η κεντρική γραμμή δειγματοληπτείται και στα σημεία που προκύπτουν εφαρμόζεται η μέθοδος που περιγράφεται στο [37], προσαρμοσμένη για τις δύο διαστάσεις. Αυτή η διαδικασία περιλαμβάνει τον καθορισμό της θεωρητικής μέγιστης ταχύτητας σε κάθε σημείο και έπειτα την προσαρμογή στις δυνατότητες επιτάχυνσης-επιβράδυνσης του οχήματος διανύοντας τη διαδρομή μία φορά προς την ορθή κατεύθυνση (επιτάχυνση) και μία προς την ανάποδη (επιβράδυνση). Υποτιμώντας τις δυνάμεις που μπορούν να προσδώσουν τα ελαστικά μπορούν να διαμορφωθούν προφίλ ταχυτήτων με αντίστοιχους συντελεστές ασφαλείας.

Αφότου επιλεγεί η επιθυμητή πορεία και η επιθυμητή ταχύτητα πάνω σε αυτήν, ο ελεγκτής PID καθορίζει την ποσότητα γκαζιού/φρένου που πρέπει να εφαρμοστεί και ο αλγόριθμος Pure Pursuit καθορίζει το πόσο πρέπει να στρίψει το τιμόνι. Ο PID θέτει σαν ταχύτητα-στόχο αυτήν που ορίζει το προφίλ ταχυτήτων για το κοντινότερο δείγμα της κεντρικής γραμμής στο όχημα. Ο Pure Pursuit επιλέγει ένα σημείο-στόχο από την επιθυμητή πορεία σε καθορισμένη απόσταση από το όχημα και στέλνει την κατάλληλη εντολή στο τιμόνι ώστε να περάσει από αυτό το σημείο-στόχο. Η απόσταση στην οποία αναζητείται το σημείο-στόχος είναι συνάρτηση της τρέχουσας ταχύτητας-στόχου, ώστε να είναι μεγάλη στις ευθείες αποφεύγοντας ταλαντώσεις και μικρή στις στροφές αποφεύγοντας τον κίνδυνο να κοπεί κάποια στροφή.

Στο πλαίσιο αυτής της εργασίας δοκιμάστηκαν και άλλες προσεγγίσεις στο πρόβλημα του Motion Planning. Η πρώτη προσέγγιση περιλαμβάνει την χρήση κινητικού-δυναμικού Probabilistic Roadmap (PRM). Η δεύτερη προσέγγιση περιλαμβάνει την χρήση Model Predictive Control (MPC). Σε αυτή την περίπτωση, ο καθορισμός τροχιάς και ο έλεγχος του οχήματος συγχωνεύονται σε μία ενιαία διαδικασία. Για τις δυναμικές εξισώσεις του οχήματος χρησιμοποιείται το μοντέλο ποδηλάτου. Οι περιορισμοί τίθενται με τέτοιο τρόπο ώστε να εξασφαλίζουν πως το όχημα μένει εντός πίστας, εντός ορίων πρόσφυσης των ελαστικών του και εντός των ορίων του μηχανισμού του τιμονιού. Τέλος, η συνάρτηση κόστους εκφράζει τον χρόνο που παίρνει στο όχημα να διασχίσει ένα καθορισμένο κομμάτι της πίστας. Έτσι προκύπτουν οι κατάλληλες εντολές προς το γκάζι, το φρένο και το τιμόνι ώστε να διασχίσει στον ελάχιστο δυνατό χρόνο το καθορισμένο κομμάτι της πίστας. Η τελευταία προσέγγιση περιλαμβάνει την αντικατάσταση του ζεύγους PID - Pure Pursuit από MPC, διατηρώντας την ίδια δομή με το υπάρχον Motion Planning μέχρι εκείνο το σημείο.

Η πρώτη προσέγγιση έχει πολύ μεγάλες απαιτήσεις μνήμης για να μπορέσει να εφαρμοστεί σε αυτόνομο μονοθέσιο. Από την άλλη, το πρόβλημα ελέγχου στο οποίο κατέληγαν οι προσεγγίσεις που χρησιμοποιούσαν MPC είτε δεν συνέκλινε σε κάποια λύση είτε καθυστερούσε να επιλυθεί.

1.7 Νέα προσέγγιση Αντίληψης

Μετά το πέρας της σεζόν του 2023 κατέστη εμφανές πως ο κύριος περιοριστικός παράγοντας του αυτόνομου συστήματος ήταν το Perception και συγκεκριμένα η δυσκολία που αντιμετώπιζε στο να εκτιμήσει την ακριβή απόστασή του από εμπόδια. Για αυτό το σκοπό αναπτύχθηκε μία νέα μέθοδος που βασίζεται στην αρχιτεκτονική του YOLOv8n. Συγκεκριμένα, τροποποιεί την κεφαλή του νευρωνικού δικτύου για να εκμεταλλευτεί τον μεγάλο όγκο από features που εξάγει ο κορμός του YOLOv8n και, έπειτα, τα αξιοποιεί για να υπολογίσει την απόσταση από τους εντοπισμένους κώνους.

Για την εκπαίδευση του τροποποιημένου νευρωνικού χρησιμοποιούνται οι προβλέψεις ενός ισχυρότερου μοντέλου πάνω σε 5000+ φωτογραφίες που έχουν συγκεντρωθεί κατά τις δοκιμές του P23. Το ισχυρό μοντέλο μοιράζεται την ίδια δομή με τα προαναφερθέντα μοντέλα Perception. Διαφέρουν ωστόσο ως προς εκδοχή του YOLO που χρησιμοποιούν (YOLOv5m6 αντί YOLOv5n) και ως προς μέγεθος του νευρωνικού που εκτιμά τη θέση των επτά σημείων στον κώνο.

Η κεφαλή του YOLOv8n αποτελείται από δύο συνελικτικά νευρωνικά δίκτυα τριών στρώσεων με κοινή είσοδο. Το ένα εκτιμά την κλάση του αντικειμένου και το άλλο το ορθογώνιο παραλληλόγραμμο που το περιβάλλει. Ακολουθώντας την ίδια λογική, προστίθεται ένα τρίτο συνελικτικό νευρωνικό δίκτυο τεσσάρων στρώσεων, στο οποίο ανατίθεται ο υπολογισμός της απόστασης από το αντικείμενο.

Το YOLOv8n αρχικά εκπαιδεύεται στα ίδια δεδομένα που είχε εκπαιδευτεί και το YOLOv5n. Αρχικοποιώντας με βάση αυτό τα βάρη του και αφότου γίνουν οι απαραίτητες αλλαγές στην κεφαλή, εκπαιδεύεται στις προβλέψεις του ισχυρότερου μοντέλου.

Η ακρίβεια της νέας μεθόδου στην εκτίμηση αποστάσεων είναι μεγαλύτερη από αυτή της παλιάς μεθόδου, ειδικά για κώνους σε απόσταση 15-20 μέτρα από την κάμερα. Η σύγκριση των δύο μεθόδων καθώς και του μοντέλου-δασκάλου αποτυπώνονται στο διάγραμμα 1.5. Εξίσου εντυπωσιακή είναι και η αύξηση του recall από 68.3% σε 94.4%.

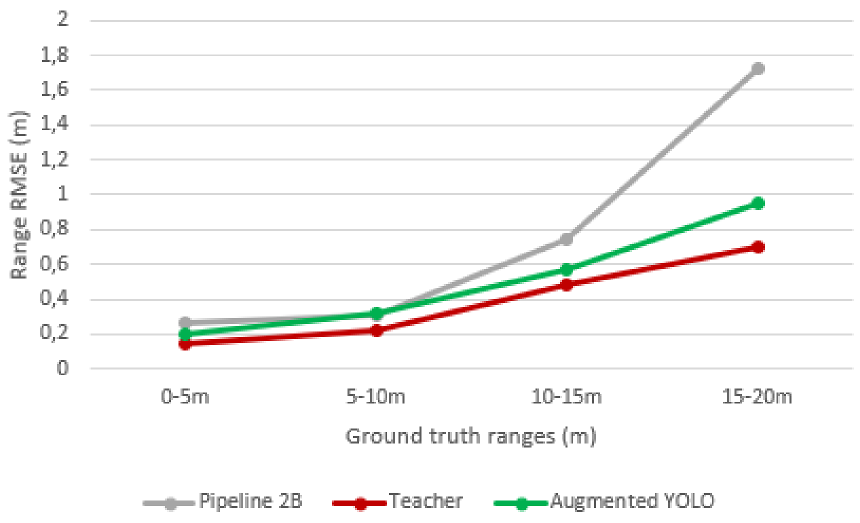


Figure 1.5: Οι ρίζες των μέσων τετραγωνικών σφαλμάτων του υπολογισμού απόστασης από κώνους για τη νέα μέθοδο, την παλιά και το μοντέλο-δάσκαλο.

Η βελτίωση του Perception μεταφράζεται και σε βελτίωση στη διαδικασία της χαρτογράφησης της πίστας όπως φαίνεται και στο διάγραμμα 1.6. Το recall ανεβαίνει από 94.7% σε 96.5% και το precision από 97.3% σε 100%.

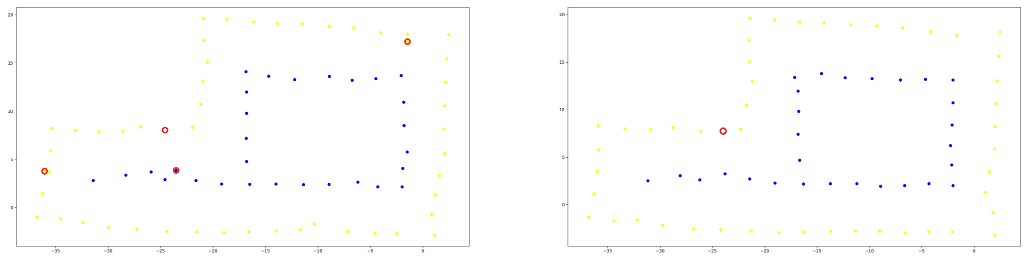


Figure 1.6: Ο χάρτης που προκύπτει με τη χρήση της παλιάς μεθόδου Perception (αριστερά) και της νέας μεθόδου (δεξιά). Λάθη στη χαρτογράφηση σημειώνονται με κόκκινους κύκλους.

Chapter 2

Introduction

This work lays out the comprehensive conceptual design of the autonomous system pipeline of PROM Racing's first Formula Student Driverless race car. Furthermore, it discusses the implementation and testing of the perception and SLAM modules. Finally, it presents a novel perception and motion planning approach which, to the best of the author's knowledge, have not been used by any other team in the Formula Student Driverless community.

2.1 Motivation and Goals

In 2022, the most prominent Formula Student competition, that of Germany, began requiring of all competing race cars to have autonomous driving capabilities, with non-compliance resulting in significant point reductions. Setting up an autonomous system became a necessity, which is what prompted this work.

The goal is to design and implement an autonomous system pipeline that would maximize the car's probability of completing all Formula Student Driverless dynamic disciplines. This meant that the focus is on simplicity and reliability rather than performance. Furthermore, PROM Racing's budget is several times smaller than that of the competing teams, providing an additional objective of staying economical with the choices in sensors and computer hardware.

With the 2023 season coming to a close, it has become obvious that the main bottleneck of the original autonomous system is the perception pipeline, and specifically, its inaccurate depth estimation. This creates the need for a revised perception module that would offer more accurate depth estimation than the original one.

2.2 Related work

Since the inaugural season of Formula Student Driverless competitions in 2017, several teams have published their work on the field.

The most prominent piece of work is [19], in which ETH's team presents the full autonomous system pipeline of their race car. The same team has also published papers delving into the details and discussing further improvements on the individual components comprising the aforementioned pipeline. These works include an end-to-end velocity estimator[35], an EKF-Graph SLAM hybrid for localization and mapping[3], an optimization-based hierarchical motion planner[40], and a learning-based Model Predictive Controller[18]. In [36], the monocular perception pipeline employed by a joint team from MIT and Delft is presented. Lastly, in [25], an EKF SLAM and a GraphSLAM approach are compared on the Formula Student race car of KIT.

Given that Formula Student Driverless competitions fall under the broad category of autonomous driving and the even broader one of robotics and automation, there is plenty more related work. However, the choice has been made for it to be mentioned in each chapter separately.

2.3 Contents

In chapter 3, background information regarding Formula Student competitions, PROM Racing NTUA, and the proposed pipeline are provided. In chapter 4, the velocity estimation module is presented. In chapter 5, the perception pipeline of P23 is discussed. In chapter 6, the SLAM and localization functionalities are described. In chapter 7, the motion planning process is outlined. Finally, in chapter 8, the novel perception approach is presented.

Chapter 3

General information

3.1 Formula Student competitions

The Formula Student competitions are student design competitions organized by the International Society of Automotive Engineers (SAE) since 1980. Teams of university students are challenged to conceive, design, fabricate, develop, and compete with small, formula style, race cars. The competitions are split into three main classes: Internal Combustion Engine Vehicle (CV) (including Combustion Hybrid Vehicle (HY)), Electric Vehicle (EV), and Driverless Vehicle (DV).

In the DV class, which is the main focus of this work, teams compete in four dynamic events: Acceleration, Skidpad, Autocross, and Trackdrive. In all dynamic disciplines, the boundaries of the circuit are denoted by a specified set of cones, with the left boundary denoted by blue cones with a white stripe and the right by yellow cones with a black stripe. The Acceleration event track is a 75m straight line, where the cones' positions are known in advance. Similarly, the map of the Skidpad track is known a priori and consists of two pairs of concentric circles in an 8-figure pattern. On the other hand, the Autocross and Trackdrive events share the same track, whose map is not provided beforehand. It is a closed loop circuit consisting of straights of up to 80m, constant turns with a diameter of up to 50m, hairpin turns with an outside diameter of at least of 9m, chicanes, multiple turns, decreasing radius turns etc. The length of one lap is approximately 200m to 500m. For the Autocross and Trackdrive events, one and ten laps of the aforementioned track have to be completed, respectively.

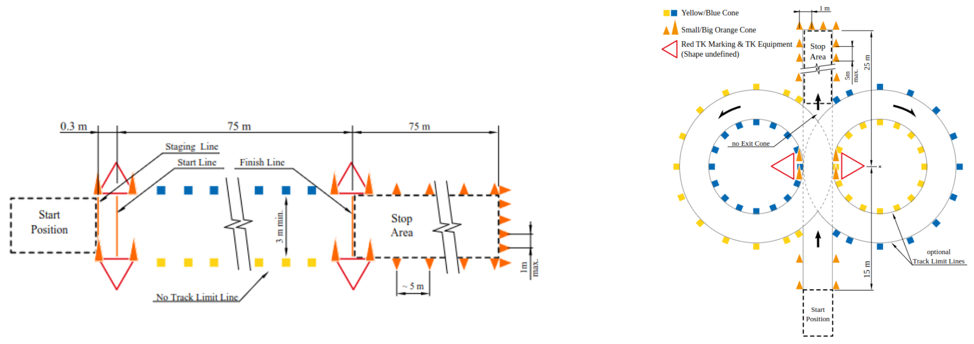


Figure 3.1: The Acceleration track layout (left), the Skidpad track layout (right).

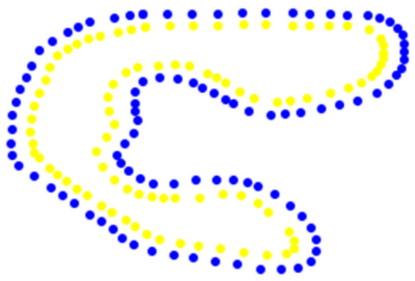


Figure 3.2: An example of an Autocross/Trackdrive track layout.

Points are awarded based on the time it took each team to complete the discipline. Regardless of the recorded time, every team that completed the discipline is awarded a small amount of points. Naturally, penalties are imposed for dropping cones or exiting the track.

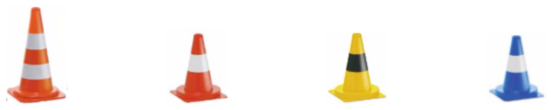


Figure 3.3: The cones used in Formula Student competitions.

3.2 PROM Racing NTUA

PROM Racing NTUA is a Formula Student team representing the National Technical University of Athens. It was founded in 2008 and competed in the CV class for the first time in 2011. It transitioned to the EV class and competed in 2021, becoming the first team in Greece to win an official Formula Student competition. Following that, a two year plan to compete in 2023 in the DV class commenced.

P23 is PROM Racing's race car for the 2023 season. It features an electric powertrain that includes a single 120kW motor on the rear axle, an inverter for controlling the motor, and an 9.1kWh accumulator to power the system. A limited slip differential handles the transmission on the rear axle. P23 is equipped with a heave–roll decoupled, push-rod actuated double wishbone suspension concept on the front axle, and a push-rod actuated spring-damper assembly featuring an anti-roll bar on the rear axle. Its aerodynamic package achieves a lift coefficient ($C_l A$) of -7.3 with the ground-effect fans on and -5.5 without them and a lift to drag ratio of -3.48 and 3.05, respectively. Finally, Hoosier R20 16.0x7.5 tires are used in both axles, data for which were obtained from the FSAE Tire Test Consortium, a volunteer-managed organization of member schools who pool their financial resources to obtain high quality tire force and moment data targeted for Formula SAE and Formula Student competitions.

Figure 3.4: P23, PROM Racing's race car for the 2023 season.

The autonomous system of P23 consists of the steering and braking actuators, the motion and perception sensors, and the computing system. The motion sensors used in P23 are an IMU, a GPS/INS, and four wheel speed sensors, one per wheel. Optical and RADAR ground speed sensors were considered, but their cost vastly exceeded the budget of the entire autonomous system. The perception sensors featured on P23 are two monocular cameras that work independently for a combined horizontal field-of-view of 105^{o}. RADAR sensors were considered, however, their inability to detect low dielectric constant objects, like the Formula Student cones, rendered them useless for the application. 3D LiDAR sensors were more enticing candidates but, again, their cost exceeded the team's autonomous system budget. Finally, commercial stereo cameras could have been used, but given the nature of the task, a more customizable approach is selected which has, so far, proven to be more accurate. For the computing system, an Intel i7-12700 CPU is used along with a TPU accelerator, a 32GB RAM, and a 1TB SSD.

Figure 3.5: The motherboard (left), the CPU (center), and the TPU accelerator (right) of P23's computing system.

Figure 3.6: The cameras and the lenses of P23.

3.3 Autonomous system software pipeline

The autonomous system software pipeline consists of four modules: perception, velocity estimation, SLAM, and motion planning. The perception module is responsible for identifying cones that act as track boundaries and estimate their color and their distance to the vehicle. The velocity estimation module is tasked with estimating the velocities and the accelerations of the vehicle. In known circuits, the SLAM module is responsible for locating the vehicle on the track map. In unknown circuits, it is also, simultaneously, building a map of the track. Finally, the motion planning module computes a trajectory to follow as well as the corresponding commands to be sent to the actuators. The flow of information in the pipeline is depicted in figure 3.7.

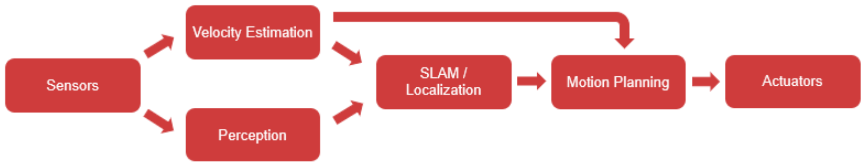


Figure 3.7: The autonomous system software pipeline.

Chapter 4

Velocity estimation

The goal of the velocity estimation pipeline is to fuse information from different motion sensors in order to continuously provide an estimate of the vehicle's velocity (u_x, u_y, ω). As mentioned in the introduction, P23 is equipped with four Hall-effect wheel speed sensors, attached to each of the four wheels, an IMU, and a separate GPS-Aided Inertial Navigation System (GPS/INS). The measurements of these sensors are fused with the help of an Extended Kalman Filter (EKF). In this chapter, the theory behind the EKF will be presented, followed by the discussion of its implementation on P23's software pipeline.

4.1 Theoretical background

4.1.1 Kalman filter

The Kalman filter[43] addresses the general problem of estimating the state of a discrete-time controlled process $x \in \mathbb{R}^n$ that is described by the following linear stochastic difference equation:

$$x_t = A_t x_{t-1} + B_t u_t + w_t$$

with measurements $z \in \mathbb{R}^m$:

$$z_t = H_t x_t + v_t$$

where w_t, v_t represent the process and measurement noise, respectively. Crucially, these are assumed to be independent random variables described by normal distributions of zero mean. A_t is a nxn matrix relating the previous to the next state, B_t is a nxm matrix relating the input to the state, and H_t is a mxm matrix relating the state to the measurements.

From here on, $x_{t|t-1}$is defined as the a priori state estimate at time step t, given knowledge of the process prior to time step t, and $x_{t|t}$ as the a posteriori state estimate at time step t, given measurement z_t. Furthermore, the a priori and a posteriori estimate errors and their respective covariances are defined as:

$$e_{t|t-1} = x_t - x_{t|t-1}$$

$$e_{t|t} = x_t - x_{t|t}$$

$$P_{t|t-1} = E(e_{t|t-1} e_{t|t-1}^T)$$

$P_{t|t} = E(e_{t|t}e_{t|t}^T)$

The Kalman filter can be broken down into two steps: predict and update. Specifically, in the predict step the new a priori estimate is produced based on the theoretic model $x_{t|t-1} = A_t x_{t-1|t-1} + B_t u_t$ and its covariance is calculated to be $P_{t|t-1} = A_t P_{t-1|t-1} A_t^T + Q_t$, with Q_t being the covariance matrix of the process noise, w_t. In the update step, the goal is to calculate the a posteriori estimate that corresponds to the minimum covariance by combining the a priori estimate with the sensor measurements. The first thing to be calculated is the innovation or pre-fit residual, which describes the difference between the measurement and its anticipated value from the a priori state estimate, along with its covariance: $y_t = z_t - H_t x_{t|t-1}$ and $S_t = H_t P_{t|t-1} H_t^T + R_t$, with R_t being the covariance matrix of the measurement noise, v_t. Based on the innovation, the a posteriori estimate is updated to $x_{t|t} = x_{t|t-1} + K_t y_t$ while the covariance is updated to $P_{t|t} = (I - K_t H_t) P_{t|t-1}$. The matrix K_t is called optimal gain and needs to be set to a value that leads to the minimization of the a posteriori estimate's covariance. Intuitively, that is equivalent to minimizing the uncertainty around the estimate of the state by utilizing information from both a theoretic model of the system and sensor measurements. It can be proven that the value that achieves that is $K_t = P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1}$.

In practice, both Q_t, R_t can often be determined prior to using the filter. That is particularly true for the measurement covariance as it is usually feasible to gather some offline sample measurements to estimate its value. When it comes to the parameter Q_t, the selection process is often less straightforward. Q_t is frequently employed to account for the uncertainty in the process model. On occasions, even when the process model is notably deficient, it can be compensated for by introducing a substantial amount of uncertainty through the choice of Q_t. Naturally, in such cases, it is imperative that the sensor measurements are dependable. Whether or not there is a rational basis for choosing the values of Q_t, R_t, often times superior filter performance can be obtained by tuning them off-line.

4.1.2 Extended Kalman filter

The Kalman filter works only if the measurement and the state transition model are both linear. In case one of these conditions does not hold true, an Extended Kalman filter (EKF)[43] may be used instead. The basic principle of the EKF is that it uses linearization around the current mean and covariance, but otherwise goes through the exact same steps as the linear Kalman filter.

This time the process is described by:

$x_t = f(x_{t-1}, u_t, w_t)$

with measurements:

$z_t = h(x_t, v_t)$

Once again, $x_{t|t-1}$is defined as the a priori state estimate at time step t, given knowledge of the process prior to time step t, and $x_{t|t}$ as the a posteriori state estimate at time step t, given measurement z_t. The a priori and a posteriori estimate errors and their respective covariances are defined as:

$e_{t|t-1} = x_t - x_{t|t-1}$

$e_{t|t} = x_t - x_{t|t}$

$P_{t|t-1} = E(e_{t|t-1}e_{t|t-1}^T)$

$P_{t|t} = E(e_{t|t}e_{t|t}^T)$

Just like the Kalman filter, the EKF can be broken down into two steps: predict and update. Specifically, in the predict step the new a priori estimate is produced based on the theoretic model $x_{t|t-1} = f(x_{t-1|t-1}, u_t)$, while its covariance is calculated to be $P_{t|t-1} = F_t P_{t-1|t-1} F_t^T + Q_t$, where F_t is the Jacobian of the process function, f, evaluated for $x = x_{t-1|t-1}$ and $u = u_t$. For the update step, the first thing to be calculated is the innovation or pre-fit residual, which describes the difference between the measurement and its anticipated value from the a priori state estimate, along with its covariance: $y_t = z_t - h(x_{t|t-1})$ and $S_t = H_t P_{t|t-1} H_t^T + R_t$, where H_t is the Jacobian of the process function, h, evaluated for $x = x_{t|t-1}$. Based on the innovation, the a posteriori estimate is updated to $x_{t|t} = x_{t|t-1} + K_t y_t$ while the covariance is updated to $P_{t|t} = (I - K_t H_t)P_{t|t-1}$. The matrix K_t is set to a value that leads to the minimization of the a posteriori estimate's covariance in the linearized space, making it potentially sub-optimal. It can be proven that the value that achieves that is $K_t = P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1}$.

4.2 Implementation

The state vector in the case of the velocity estimation EKF contains the longitudinal velocity u_x, the lateral velocity u_y, the angular velocity ω_z, the longitudinal acceleration a_x, the lateral acceleration a_y, and the angular acceleration α_z.

4.2.1 Process model

The model used to describe the process $x_t = f(x_{t-1}, u_t, w_t)$ is a kinematic bicycle model with constant accelerations and angular velocity. The equations are as follows:

$\dot{u}_x = a_x + \omega * u_y + w_{u_x}$

$\dot{u}_y = a_y - \omega * u_x + w_{u_y}$

$\dot{\omega}_z = \alpha_z + w_{\omega_z}$

$\dot{a}_x = w_{a_x}$

$\dot{a}_y = w_{a_y}$

$\dot{\alpha}_z = w_{\alpha_z}$

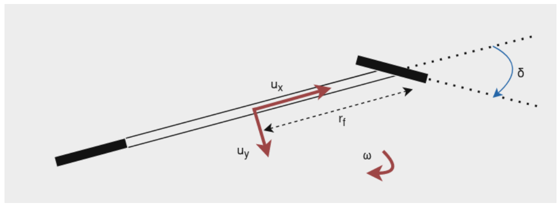


Figure 4.1: The bicycle model.

4.2.2 Sensor measurements

As previously mentioned there are 3 types of motion sensors on P23: wheel speed sensors, IMU, and GPS/INS.

Since a bicycle model is being used, the measurements of the wheel speed sensors on each axis must be concatenated into a single measurement, namely the mean of the two values. As a result, there are two measurements:

$$z_{\omega_{front-wheel}} = \frac{u_x * cos(\delta) + (u_y + \omega_\theta * r_f) * sin(\delta)}{R_{wheel}} + v_{\omega_{front-wheel}}$$

$$z_{\omega_{rear-wheel}} = \frac{u_x}{R_{wheel}} + v_{\omega_{rear-wheel}}$$

where δ is the steering angle, r_f is the distance between the point at which the velocity is estimated and the center of the front axis, and R_{wheel} is the radius of the wheels. It is worth noting that the slip ratio is assumed to be zero at all times, which is fairly accurate for the low accelerations expected in this case.

From the IMU, the lateral and longitudinal acceleration and the angular velocity measurements are being utilized:

$$z_a = Rot(a + \omega \times (\omega \times r) + \alpha \times r) - g + v_a$$

$$z_\omega = Rot(\omega) + v_\omega$$

where r represents the distance vector from the point at which the velocity is estimated to the position of the IMU, Rot represents the, assumed to be constant, rotation matrix from the vehicle frame to the IMU frame, and g represents the acceleration due to gravity. It is worth noting that the angular velocities ω_x, ω_y, the acceleration a_z, and the angular accelerations α_x, α_y are considered to be negligible. Although this is not always the case, this choice helps retain the size of the state vector small and the computational load of the process low.

From the GPS/INS, the longitudinal and lateral velocity estimates are being utilized:

$$z_u = Rot(u + \omega \times r) + v_u$$

where r represents the distance vector from the point at which the velocity is estimated to the position of the GPS/INS, and Rot represents the, assumed to be constant, rotation matrix from the vehicle frame to the GPS/INS frame.

4.2.3 Process and measurement covariances

The IMU measurement covariances are determined by the sensor's datasheet and validated by taking measurements on rest. The GPS/INS has a built-in microprocessor that runs a robust INS Kalman Filter and apart from the measurements, it additionally provides their covariance matrix. Things are not as straightforward with the wheel speed sensors and the process model, as both of them have to be fine-tuned off-line. In order to conduct any form of fine-tuning, ground truth data must be obtained. If there was a more accurate sensor that had measurements that could be used as the ground truth for velocity estimation, then there would be no need for a Kalman filter and said sensor would be used instead. Instead, the vehicle is driven on a mapped narrow circuit and the covariance matrices are fine-tuned so that the integral of the velocity estimates matches the shape of the circuit.

Chapter 5

Perception

The goal of the perception pipeline is to detect cones, recognize their color, and calculate the vehicle's distance to them. The most commonly used sensor in Formula Student for this type of task is LiDAR. It is capable of computing distances to obstacles by targeting them with laser beams and measuring the time it takes for the reflected light to return to the receiver. It offers outstanding accuracy in depth estimation (typically less than 5cm error), but no color information. As a result, it is often used in a sensor fusion setup alongside cameras, which contribute the, otherwise missing, cone color information. The main drawback, however, of the LiDAR sensor is its high cost, rendering it prohibitive for smaller Formula Student teams. That, along with recent developments in real-time object detection, tipped the scale towards a camera-only perception approach for the team's first iteration of an autonomous car. Specifically, two monocular cameras are used for a combined field-of-view of 105°. In this chapter, the theory behind object detection networks and the Perspective-n-Point problem will be presented, followed by the discussion of three alternative solutions to the aforementioned problem and their results.

5.1 Theoretical background

5.1.1 Object detection

Generic object detection aims at locating and classifying existing objects in any one image, labeling them with rectangular bounding boxes and providing confidence scores. The methodologies employed in generic object detection can generally be classified into two primary categories. One approach adheres to the conventional object detection workflow, commencing with the generation of region proposals before proceeding to classify each proposal into distinct object categories. In contrast, the second approach treats object detection as either a regression or classification problem, adopting an integrated framework to directly attain the final outcomes, which encompass object categories and their respective spatial locations.

Region proposal-based methods encompass a variety of techniques; however, since they are outside the scope of this work only the core method, R-CNN[16], will be discussed. R-CNN partly mimics the human brain's attention mechanism: it begins with a broad scan of the entire scene and then focuses on regions of interest (RoIs). Although R-CNN is agnostic to the particular region proposal method, selective search was used in the original paper, generating approximately 2000 region proposals for each image. Each region proposal is subsequently warped to a 227x227 resolution and is forward propagated through a CNN to get a 4096-dimensional feature map. Then, each extracted feature vector is scored using a SVM pretrained for each class. Finally, a greedy non-maximum suppression is applied, for each class independently, based on the intersection-over-union (IoU) overlap with higher scoring regions.

Region proposal-based frameworks consist of several interconnected phases, encompassing region proposal generation, CNN-based feature extraction, classification, and bounding box regression. Typically, these components are trained separately. Even in modern end-to-end models like Faster R-CNN[33], there is still a need for separate training to acquire shared convolutional parameters between the Region Proposal Network (RPN) and the detection network. Consequently, the time required to manage these distinct components becomes a limiting factor in real-time applications.

In contrast, one-step frameworks that employ global regression and classification techniques, directly mapping from image pixels to bounding box coordinates and class probabilities, can significantly reduce the computational time. Among the various object detection algorithms that employ the aforementioned techniques, the YOLO (You Only Look Once)[38] framework has stood out for its remarkable balance of speed and accuracy.

The first version of YOLO[30] unified the object detection steps by detecting all the bounding boxes simultaneously. To achieve this, it divides the input image into an S $\times$ S grid and predicts B bounding boxes, all of the same class, along with their confidence scores for C distinct classes within each grid cell. Each bounding box prediction comprises five values: Pc, bx, by, bh, and bw. Here, Pc signifies the confidence score, indicating the model's confidence in the box containing an object and its accuracy. The coordinates bx and by represent the box's center relative to the grid cell, while bh and bw represent the box's height and width relative to the entire image. The result of YOLOv1 is a tensor of dimensions S $\times$ S $\times$ (B $\times$ 5 + C), with the option of applying non-maximum suppression to eliminate redundant detections.

Its architecture comprises 24 convolutional layers followed by two fully-connected layers that predict the bounding box coordinates and probabilities. All layers use leaky rectified linear unit activation except for the last one that uses a linear activation function. The first 20 layers are pre-trained at a resolution of 224 $\times$ 224 using the ImageNet dataset, leaving the last four layers to be fine-tuned for any given dataset.

YOLOv1's straightforward architecture, coupled with its innovative one-shot regression for the entire image, significantly boosted its speed, enabling real-time performance—a feat unmatched by contemporary object detectors. Nevertheless, despite its impressive speed, it exhibited a larger localization error compared to state-of-the-art methods like Fast R-CNN[15]. This limitation stemmed from three primary factors. Firstly, it had a restriction in detecting a maximum of B (equal to 2 in the original paper) objects of the same class within a grid cell, limiting its ability to predict objects in proximity. Secondly, it encountered difficulties in predicting objects with aspect ratios not encountered during training. Thirdly, it relied on coarse object features due to the effects of down-sampling layers in its architecture.

Later versions tackled these problems by adding new features. YOLOv2's[31] inclusion of batch normalization improved convergence and reduced overfitting. Pre-training the model on a higher (448x448) resolution led to improvements in performance in higher resolution inputs. The shift to a fully convolutional architecture drastically reduced the memory size of the network. The introduction of anchor boxes, boxes with predefined shapes used to match prototypical shapes of objects, led to predicting bounding boxes more accurately.

In YOLOv3[32] residual connections were added to the network architecture which tackled the vanishing/exploding gradient problem. In addition to its expanded architecture, YOLOv3 incorporated multi-scale predictions, in other words predictions at various grid sizes. This innovation proved instrumental in achieving finer, more detailed bounding boxes and marked a significant enhancement in the accurate prediction of smaller objects—a notable weakness in the earlier iterations of YOLO. Later variants of YOLOv3 employed spatial pyramid pooling, a type of pooling that concatenates multiple max pooling outputs without subsampling (stride = 1), each with a different kernel size k $\times$ k (k = 1, 5, 9, 13), increasing the receptive field without affecting the inference speed.

YOLOv4[7] added cross-stage partial connections to the network architecture which reduced the computation load of the model while retaining the same accuracy. Most of the advancements in this version, however, were a result of new training methods. In addition to standard augmentations like random adjustments to brightness, contrast, scaling, cropping, flipping, and rotation, the authors introduced mosaic augmentation. This technique merges four images into a single composite image, enabling object detection even when objects are positioned beyond their typical context. This helps mitigate the necessity for a large mini-batch size when using batch normalization. Additionally, self-adversarial training and hyperparameter optimization with genetic algorithms were also employed.

YOLOv5[38] integrates an algorithm referred to as AutoAnchor. This pre-training tool examines and refines anchor boxes to ensure their suitability for the dataset and training parameters, including image size. Initially, it employs a k-means function on the dataset labels to establish initial conditions for a Genetic Evolution (GE) algorithm. The GE algorithm subsequently evolves these anchors through a default of 1000 generations, utilizing the CIoU loss and Best Possible Recall as its fitness function.

5.1.2 Perspective-n-Point

Perspective-n-Point is the problem of estimating the pose of a calibrated camera given a set of n 3D points in the world, their corresponding 2D projections in the image, as well as the intrinsic parameters of the camera. The camera pose consists of 6 degrees-of-freedom (DOF) which are made up of the rotation (roll, pitch, and yaw) and 3D translation of the camera with respect to the world.

The projection of 3D points in the world on to the image can be described by the following equation:

$$sp_c = K[R|T]p_w$$

where p_w is the homogeneous world point, p_c is the corresponding homogeneous image point, K is the matrix of intrinsic camera parameters, s is a scale factor for the image point, and R and T are the desired 3D rotation and 3D translation of the camera (extrinsic parameters) that are being calculated.

Consequently, for each world 3D point the following set of equations holds true:

$$s\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}f_x & \gamma & u_0\\0 & f_y & v_0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_1\\r_{21} & r_{22} & r_{23} & t_2\\r_{31} & r_{32} & r_{33} & t_3\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

where f_x and f_y are the scaled focal lengths, γ is the skew parameter which is sometimes assumed to be 0, and (u_0, v_0) is the principal point.

There are multiple methods of solving this equation for the extrinsic parameters of the camera. In this work, Infinitesimal Plane-Based Pose Estimation (IPPE)[10] is being used since the 3D world points lie on the same plane. This method capitalizes on the observation that when dealing with a noisy homography, it tends to provide a more accurate estimation of the transformation between the model plane and the image in certain regions of the plane compared to others. It, therefore, revolves around identifying a point where this transformation is most accurately estimated and utilizing only the local transformation at that point to constrain the pose. Because IPPE is analytic it is extremely fast.

5.2 Perception pipeline

Each version of the perception pipeline has to leverage the available information regarding the cone dimensions to deliver high precision and recall measurements to the SLAM module.

In this work, three versions will be discussed. The first version only uses the bounding box dimensions from YOLO to calculate the distance to the cone, the second one utilizes a keypoint detection neural network on each YOLO bounding box to turn the original problem into a geometric (PnP) one, and the third version adds a PnP recalculation module, based on reprojection error.

5.2.1 Bounding box only version

Due to the fact that the exact dimensions of the cones are known a priori, it would be reasonable to assume that the dimensions of the YOLO bounding boxes contain all the information required for an accurate distance estimation.

The first step of the pipeline is to forward propagate the image through the YOLO network. Specifically, the YOLOv5n variant is used as the TPU can only fit 8MB of model parameter data in its SRAM and, therefore, any larger YOLO variant would not fit, making the inference time critically higher. The YOLOv5n network is originally trained on the MS COCO dataset[27] and fine tuned on the FSOCO dataset[41]. As a result of the fine tuning, it is capable of detecting the four classes of cones featured in the Formula Student competitions with a mAP of 56.7%.

Using a pinhole camera model and the resulting bounding box for each cone, two approximately similar pyramids can be drawn (see figure 5.1). The first one features the bounding box on the image plane as its base and the pinhole as its apex, while the second one features the face of the cone in the 3D world as its base and shares the same apex. The distance between the midpoint of the cone base and the camera is equal to the median of the triangle formed by the lower end of the cone face and the apex. The length of the corresponding line segment on the image side, that connects the apex to the midpoint of the cone's depicted lower end, can easily be calculated using the Pythagorean theorem. The ratio between the two is equal to the square root of the ratio between the areas of the two pyramid bases. Using that, the distance between the midpoint of the cone base and the camera can finally be calculated.

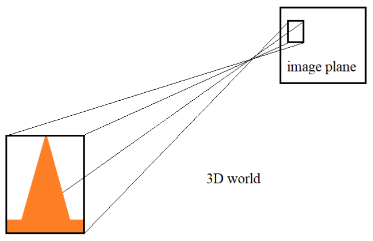


Figure 5.1: The two approximately similar pyramids formed by the face of the cone on the 3D world, the pinhole, and the projection of the bounding box on the image plane.

5.2.2 Keypoint detection version

The previous method is the result of certain costly simplifications. One important omission is that the base corresponding to the face of the cone in the 3D world has an area that is a function of the relative angle between the camera and the cone as seen in figure 5.2. However, because the content of the bounding box is not accounted for, there is no way of knowing the aforementioned angle. Additionally, the bounding boxes often fail to fit the cone perfectly, leading to them being smaller or larger than the intended size as seen in figure 5.2. For these reasons a new method that analyzes the content of the bounding box is developed.

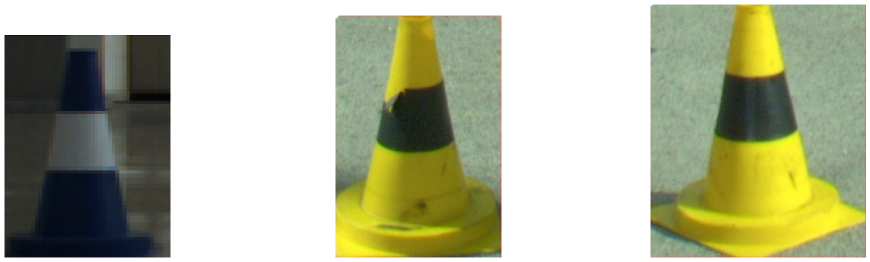


Figure 5.2: Poor quality bounding boxes hamper quality (left image), but even if the bounding boxes were perfect, cone orientation would still matter (middle and right images).

One can notice that the Formula Student cones feature certain distinguishable and orientation invariant local features. This new method entails measuring the distances of these corners from the center of the cone's base in the 3D world, finding the pixels at which they are depicted on the bounding box, and solving the ensuing Perspective-n-Point problem assuming the center of the cone's base as the origin of the 3D world. The resulting translation vector is equal to the distance between the origin of the 3D world, in this case the center of the cone's base, and the camera.

The first challenge with this method is finding the distinguishable and orientation invariant local features on each cone. Traditional computer vision approaches to finding local features, such as SURF[6], SIFT[23], and HOG[11], are not fit for the task since matching the results to predetermined 3D world points would be highly inconsistent. Instead, the approach of [36] was adopted. A convolutional neural network, containing convolutional, batch normalization, and ReLU/Sigmoid activation layers as well as residual blocks, was designed to detect the position of seven keypoints as shown in figure 5.3. The network receives as input the cropped image corresponding to each YOLO bounding box, expanded by a couple of pixels on each side, and resized to a standard size of 64x48 pixels. The output consists of seven heatmaps, one for each keypoint, where the maximum element of the heatmap corresponds to the position of said keypoint on the bounding box image.

The keypoint network was trained on a custom dataset. The dataset originally contained 3835 images of cropped cones, but after augmentation that included jittering, cropping, blurring, randomly applying gamma correction to certain parts of the image to mimic shade or overexposure, and randomly setting the colour of certain parts of the image to mimic obstacles, it was expanded to almost 8000 images. The loss function that was used was MSE loss.

With the coordinates of the keypoints on the original image and their 3D positions relative to the center of each cone's base known, a Perspective-7-Points problem can be set up. OpenCV's[8] solvePnP function is then used to derive the translation vectors connecting the camera to the center of each cone's base. Specifically, the IPPE method is used to solve the problem since the 3D world keypoints lie on the same plane.

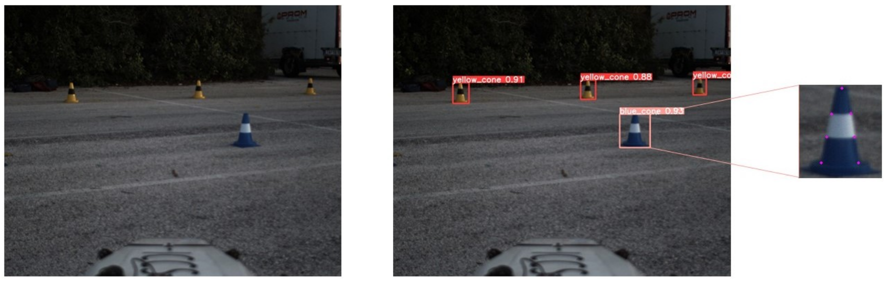


Figure 5.3: The perception pipeline with the keypoint network added.

5.2.3 Reprojection error version

Although the previous method tackled many of the problems the first one faced, it still has its weaknesses. It achieves a significant improvement on nearby cones where the cone orientation really comes into play, however, for more distant cones the accuracy actually deteriorates. This can attributed to the fact that cones further away appear blurred, making it especially hard to detect corners. The less common mistakes made on cones in a closer range can be attributed to multiple keypoints being assigned to the same corner or illumination messing with the placement of keypoints, both shown in figure 5.4.

Figure 5.4: On the left image two keypoints are assigned to the same corner, whilst on the right, the reflection of the sun leads to the misplacement of the bottom left keypoint.

One way of dealing with this issue is by allowing a solution of the PnP problem for less than seven keypoints when some of the keypoints are misplaced. The best way of deciding if a keypoint detection has failed is via the reprojection error. As previously mentioned, every time a PnP problem is solved a rotation-translation matrix is calculated that describes the pose of the camera in the 3D world frame. Using this as the extrinsic camera parameter matrix, the keypoints' 3D positions are projected on the image plane. If there is a significant difference between where they were reprojected and where they were inferred to be on the first place then one or more keypoints were misplaced.

Testing has shown that the best strategy is switching to P(n-1)P whenever PnP could not produce a result with reprojection error below a threshold. The best results occur when the least amount of points used for PnP is 6.

5.3 Results

For the testing of the methods described above, a dataset was created that contained images of cones along with their distance to the camera. In total, the testing set contains 486 images and 2745 cones.

For each method, the range Root Mean Squared Error (RMSE) is measured. Since the RMSE varies significantly based on the distance of the cones, 4 values are computed for ranges of 0-5m, 5-10m, 10-15m, and 15-20m. One parameter that directly affects the results of each method is the confidence threshold applied at the YOLOv5n non-max suppression. It is constantly set equal to 0.75 for the purposes of this experiment leading to a recall rate of 68.3%.

The results from the bounding-box only (#1), keypoint detection (#2A), and reprojection error (#2B) pipelines are presented in figure 5.5.

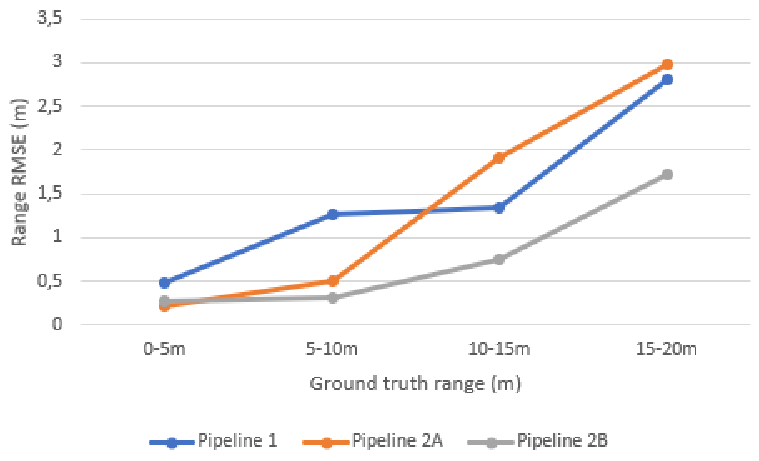


Figure 5.5: Range RMSE of the 3 pipelines. The keypoint detection approach beats the bounding-box only approach for closer cones and the reprojection error variant is the best option overall.

5.3.1 Runtimes

YOLOv5n inference on the TPU typically lasts 25ms per image. The keypoint network inference on the iGPU typically lasts 2ms per bounding box. Solving each PnP problem takes less than 1ms. As a result, it is possible to run YOLOv5n, the keypoint network, and the reprojection error module for two cameras at 10Hz.

Chapter 6

SLAM

The goal of the SLAM module is to construct a map of the circuit and to provide continuous estimates of the vehicle's position in it. Formula Student tracks are relatively flat and the only objects present are the cones that denote the boundaries. Consequently, a 2D map with the cones depicted as data points is used. To solve this kind of problem, there are two prevalent methods: filter and graph SLAM[1]. In this chapter, the theory behind these methods will be presented, followed by the discussion of the iSAM2-based implementation and its results.

6.1 Theoretical background

6.1.1 Filter SLAM

Filtering techniques involve two primary stages: a predictive phase and an updating phase. They are generally regarded as a Maximum A Posteriori (MAP) method, wherein data from the velocity estimation module are employed to estimate the robot's initial pose distribution. These estimates are then combined with data obtained from the perception module to establish the likelihood distribution.

The most widespread filtering SLAM is the EKF SLAM[5], which works in roughly the same manner as the EKF that is described in the velocity estimation chapter. The only difference is that the state vector is expanded to include the pose (x, y, ϑ) of the vehicle as well as the positions of the mapped cones (x, y) in cases where the track is unknown. The added types of measurements are the range, theta measurements from the perception pipeline, which based on the predicted state have a value of $\sqrt{(x - x_{cone})^2 + (y - y_{cone})^2}$ and $\arctan((y - y_{cone})/(x - x_{cone})) - \theta$, respectively.

6.1.2 Graph SLAM

Graph SLAM techniques aim to compute the complete robot trajectory by analyzing the entire set of inputs from velocity estimation and perception. This is viewed as an advantage when compared to filtering processes in terms of accuracy. However, one major drawback of graph-based SLAM is the substantial memory consumption it entails, as it incorporates all pose estimations within the computation process[1].

Graph SLAM's primary objective is to create a graph based on the provided control inputs and measurements, formulate an optimization problem, and ultimately determine the configuration that minimizes the cost function. The graph used to represent the SLAM problem is a weighted, directional graph consisting of two types of edges and two types of vertices, like the one in figure 6.1. Each vehicle pose is represented by a pose vertex and each mapped landmark is represented by a landmark vertex. Consecutive pose vertices are connected via odometry edges. Landmark vertices are connected to the corresponding vehicle poses using landmark edges.

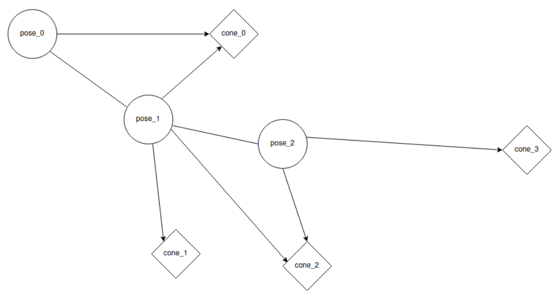


Figure 6.1: The graph representing a simple SLAM problem.

The cost function of the optimization problem expresses the likelihood of the map having a specific configuration and the vehicle being on a specific location based on the odometry and perception inputs received so far. Consequently, for each new odometry measurement the following factor is added to the cost function:

$$(x_t - g(u_t, x_{t-1}))^T Q_t^{-1} (x_t - g(u_t, x_{t-1}))$$

where x_{t-1} and x_t are the landmark vertices at t-1 and t, respectively, u_t is the velocity estimate at t, and Q_t is the covariance matrix of the displacement. Similarly, for each new perception measurement the following factor is added to the cost function:

$$(z_{t,i} - h(x_t, l_i))^T R_t^{-1} (z_{t,i} - h(x_t, l_i))$$

where $z_{t,i}$ is the measured range, theta between the vehicle and the cone, l_i is the associated cone vertex, and R_t is the covariance matrix of the perception measurement. In total the cost function is equal to:

$$J = x_0 \Omega_0^{-1} x_0 + \sum (x_t - g(u_t, x_{t-1}))^T Q_t^{-1} (x_t - g(u_t, x_{t-1}))$$

$$+ \sum (z_{t,i} - h(x_t, l_i))^T R_t^{-1} (z_{t,i} - h(x_t, l_i))$$

which leads to the following nonlinear least squares problem:

$$x^*, l^* = argmin_{x,l} \sum (x_t - g(u_t, x_{t-1}))^T Q_t^{-1} (x_t - g(u_t, x_{t-1}))$$

$$+ \sum (z_{t,i} - h(x_t, l_i))^T R_t^{-1} (z_{t,i} - h(x_t, l_i))$$

or in the case of a known track:

$$x^* = argmin_x \sum (x_t - g(u_t, x_{t-1}))^T Q_t^{-1} (x_t - g(u_t, x_{t-1}))$$

$$+ \sum (z_{t,i} - h(x_t, l_i))^T R_t^{-1} (z_{t,i} - h(x_t, l_i))$$

When dealing with nonlinear process models, g, and nonlinear measurement functions, h, particularly in cases where a suitable linearization point is not readily accessible, nonlinear optimization techniques come into play. Examples of such techniques include the Gauss-Newton method and the Levenberg-Marquardt algorithm[21]. These methods work by iteratively solving a series of linear approximations of the equation to progressively approach its minimum. In this case, each iteration involves linearization around the previous estimate of x,l and solving for the increment that will be added to it:

$$\Delta = argmin_{\Delta}(A\Delta + b)^T(A\Delta + b)$$

One way to deal with this problem is by applying QR factorization:

$$(A\Delta + b)^T(A\Delta + b) =$$
$$(Q\begin{bmatrix} R \\ 0 \end{bmatrix}\Delta - b)^T(Q\begin{bmatrix} R \\ 0 \end{bmatrix}\Delta - b) =$$
$$(Q^TQ\begin{bmatrix} R \\ 0 \end{bmatrix}\Delta - Q^Tb)^T(Q^TQ\begin{bmatrix} R \\ 0 \end{bmatrix}\Delta - b) =$$
$$(\begin{bmatrix} R \\ 0 \end{bmatrix}\Delta - \begin{bmatrix} d \\ e \end{bmatrix})^T(\begin{bmatrix} R \\ 0 \end{bmatrix}\Delta - \begin{bmatrix} d \\ e \end{bmatrix}) =$$
$$(R\Delta - d)^T(R\Delta - d) + e^Te$$

where $\begin{bmatrix} d & e \end{bmatrix}^T = Q^Tb$. This becomes minimal iff $R\Delta = d$.

6.1.3 iSAM2

iSAM2[20] is an algorithm that solves the above problems efficiently, enabling real-time graph SLAM. In iSAM2, instead of the type of graph typically used in graph SLAM approaches, factor graphs are utilized instead. A factor graph is a bipartite graph $G = (F, \Theta, E)$ with two node types: factor nodes $f_i \in F$ and variable nodes $\theta_i \in \Theta$ (Θ contains both vehicle and cone poses), like the one in figure 6.2. Edges $e_j \in E$ are always between factor nodes and variables nodes. When assuming Gaussian measurement noise:

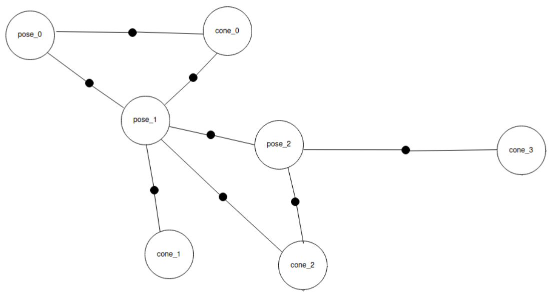


Figure 6.2: The factor graph representing the same simple SLAM problem.

$$f_i \propto exp(-\frac{1}{2}(h_i(\Theta_i) - z_i)^T\Sigma_i^{-1}(h_i(\Theta_i) - z_i))$$

A factor graph G defines the factorization of a function $f(\Theta)$ as:

$$f(\Theta) = \prod f_i(\Theta_i)$$

and optimizing it would be equivalent to:

$\Theta^* = argmax_\Theta f(\Theta)$

Just like with the original graph an iterative method is employed, linearizing the above formula around the previous estimate and subsequently updating it. However, since maximizing the product of the factors is hard, the logarithm of it is chosen instead, leading to:

$argmin_\Theta(-log(f(\Theta)) = argmin_\Theta(\frac{1}{2}\sum(h_i(\Theta_i) - z_i)^T\Sigma_i^{-1}(h_i(\Theta_i) - z_i))$

which upon linearization looks like this:

$argmin_\Delta(-log(f(\Delta)) = argmin_\Delta((A\Delta - b)^T(A\Delta - b))$

leading to the exact same problem as before. Then, the way it offloads most of the computational burden is through incremental variable re-ordering and fluid relinearization.

6.1.4 Data association

Another important aspect of any SLAM algorithm is the data association, which consists of matching measurements to their corresponding landmark. The most common method of conducting data association is the Nearest Neighbor (NN) algorithm. NN assigns each measurement to the closest predicted landmark measurement. If the best match is at a distance that exceeds a predetermined threshold, a new landmark is inserted into the map. The metric used for distance measuring can either be Euclidean or Mahalanobis distance. Employment of Mahalanobis distance constitutes the maximum likelihood solution to the data association problem.

6.2 Implementation

The SLAM module receives velocity estimates (u_x, u_y, ω) from the velocity estimation module along with the corresponding covariance matrix at a 50Hz frequency. It also receives a list of cones, specifically their color, their distance to the vehicle, and the angle at which they were observed, from the perception module at a 10Hz frequency. The covariance of the perception measurements is computed based on experimental data and is a function of the distance of the cone to the vehicle.

The factor graph in the case of an unknown circuit consists of two types of factors: odometry and perception factors. Adding perception factors is pretty straightforward, however, this is not the case with odometry factors. Velocity estimation provides velocity instead of displacement estimates. The transition from velocity to displacement is performed by the Euler method, which is adequately accurate since the integration step is equal to 1/50 of a second. Since the implementation of iSAM2 in the gtsam library[12] requires the displacement to be expressed on the vehicle coordinates, the resulting displacement is: $(u_x * dt, u_y * dt, \omega * dt)$. That means the covariance matrix provided by the EKF must change to: $(dt * I)^T P_t (dt * I)$.

In the case of a circuit that is known in advance, the positions of the cones on the map should not be subject to change. In other words, the perception factors, f_i, should be a function of vehicle poses but not of cone poses: $f_i(\Theta) = f_i(x)$. That is achieved by setting up a custom class of factors in gtsam.

For data association, the Nearest Neighbor (NN) algorithm is used. NN assigns each expected cone pose, based on the perception measurement, to the closest mapped cone that shares the same color. If the best match is at a distance that exceeds a threshold of 1.5m, a new landmark is inserted into the map. The threshold is determined based on simulations run on data collected from the vehicle.

A separate data structure is maintained that contains information about each cone on the factor graph, including its color and its estimated pose, but also how many times it has been observed. To prevent false positives from affecting the path planning procedure, the map published by the SLAM module only contains cones that have been observed 5 times or more. Likewise, this threshold is determined based on simulations run on data collected from the vehicle. The eventual map published is limited to a region around the vehicle, since parts that have been crossed in the distant past are of no interest to the motion planner.

The main drawback of using graph SLAM is the fact that the computational load of updating the factor graph increases over time with new measurements being added[25]. If the entire SLAM module is running on a single thread, a delayed update might prevent the vehicle from utilizing new measurements to estimate its latest position on the track, which in turn increases the risk of failing to complete the discipline or, even worse, of crashing. To tackle that, a second thread is used for updates exclusively. A deep copy of the data structure containing the new factors and initial estimates is created and used for the update, while the original is emptied and ready to receive new measurements at the same time that the update is executed.

6.3 Results

On the unknown tracks that the module is tested on, it achieves a 94.7% recall and 97.3% precision rate in terms of cone detection. A common theme on these tracks, however, is that the minimum distance between cones is larger than the one expected to be found in a typical Formula Student autocross track. That would normally require lower distance thresholds for introducing new cones to the map, which would in turn hamper the precision score. A map produced for an unknown test track is shown in figure 6.3.

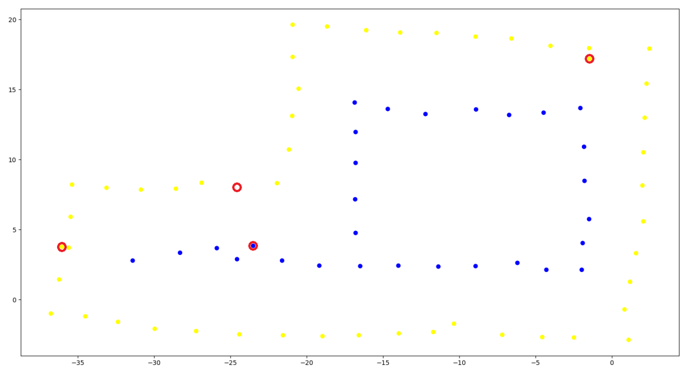


Figure 6.3: SLAM produced map of an unknown test track. The false positives and negatives are denoted using red circles.

Chapter 7

Motion planning

The motion planning module is responsible for both trajectory generation and path tracking. Given that Formula Student race tracks are narrow, the accuracy of the self-generated maps low, and the penalty for exiting the track higher than the reward for adopting a race line, the trajectory generator is tasked with extracting the track centerline instead of calculating a more optimized trajectory. Subsequently, the velocity profile of the centerline is calculated by taking the dynamics of the vehicle into consideration. Finally, the trajectory tracking problem is solved by combining a PID controller for longitudinal control and a Pure Pursuit controller of variable look-ahead distance for lateral control. In this chapter, each of the three aforementioned steps will be analyzed.

7.1 Centerline extraction

A commonly used technique for extracting the centerline of tracks is 2D shape skeletization[34]. However, to perform the skeletization the boundaries of the track need to be known in advance. Instead, a simpler, less computationally demanding approach[19] is adopted that leverages the strict structure of Formula Student circuits. The centerline extraction algorithm consists of four steps: first, the search space is discretized using a triangulation algorithm, second, a tree of possible paths is grown through the discretized space, third, the aforementioned paths are based on a plausibility metric, and finally, a spline is interpolated over the most likely path.

The triangulation algorithm used in this work is the Delaunay triangulation algorithm[26], which subdivides the xy space into connected triangles whose vertices are the positions of the cones on the 2D map. The centerline crosses through points equidistant from the boundaries, or in this case, cones, making all midpoints of the triangulation edges potential waypoints. It is worth noting that not all midpoints are good waypoints; there is no guarantee that midpoints between distant cones even lie within the track limits. Consequently, sliver triangles should be avoided, which is exactly what Delaunay triangulation manages to do by maximizing the minimum of all the angles of the resulting triangles. The result of running Delaunay triangulation on a past Formula Student circuit can be seen in figure 7.1.

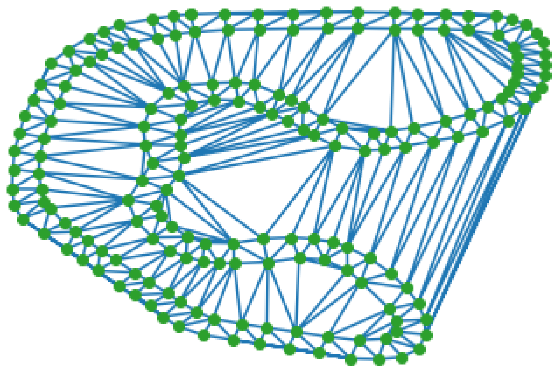


Figure 7.1: Example of running Delaunay triangulation on a Formula Student circuit.

The tree of possible paths is grown in a breadth-first manner. If a midpoint on the borders of the convex hull of the circuit is reached then there are no midpoints to be added to the tree. On the contrary, when the midpoint is not on the boundary then there are two midpoints from the neighboring edges to be added as children of the first midpoint. The tree is allowed to grow up to a certain depth as the map is innately less accurate on the borders of the perception range. The root of the tree is the current position of the vehicle which constitutes a separate case, as there are three midpoints on neighboring edges to be added to the tree.

To rank the possible paths, two facts about Formula Student circuits have to be taken into consideration. Firstly, the cones on each side of the track have a distinctive color that can accurately be identified by the perception pipeline. As a result, crossing edges of same colored cones is penalized. Secondly, large angle changes from one path segment to the next are unlikely because even in sharp corners, multiple cones are used, leading to more gradual changes. Consequently, large maximum angle changes are penalized accordingly. To limit the search domain without sacrificing the optimality of the final result, a penalty threshold is set for pruning branches with low potential of leading to a correct path.

In the end, a cubic spline is interpolated over the waypoints of the top ranked path. The resulting trajectory is continuous and differentiable, its first and second order derivatives are continuous and differentiable, and its curvature is continuous, too.

7.2 Velocity profile

For the calculation of the velocity profile, a bicycle model of the vehicle is used. The equations describing the position, the velocity, and the acceleration of the vehicle's center of gravity on the inertial frame are:

$$r_I = x(s)\hat{\imath} + y(s)\hat{\jmath}$$

$$u_I = \frac{dx}{ds}\frac{ds}{dt}\hat{\imath} + \frac{dy}{ds}\frac{ds}{dt}\hat{\jmath}$$

$$a_I = V^2\left(\frac{d^2x}{ds^2}\hat{\imath} + \frac{d^2y}{ds^2}\hat{\jmath}\right) + \dot{V}\left(\frac{dx}{ds}\hat{\imath} + \frac{dy}{ds}\hat{\jmath}\right)$$

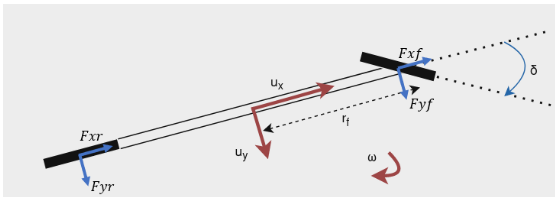


Figure 7.2: The bicycle model.

where s, V are the distance and the speed along the path, respectively.
The equations describing linear motion on the vehicle (body) frame can be summed up to:

$$\sum F_B = m * a_B = m * Rot(I \rightarrow B) * a_I \iff$$

$$F_{xr} + F_{xf} - \tfrac{1}{2}\rho C_d A V^2 = [m * Rot(I \rightarrow B) * (V^2(\tfrac{d^2x}{ds^2}\hat{\imath} + \tfrac{d^2y}{ds^2}\hat{\jmath}) + \dot{V}(\tfrac{dx}{ds}\hat{\imath} + \tfrac{dy}{ds}\hat{\jmath}))]_x$$

$$F_{yr} + F_{yf} = [m * Rot(I \rightarrow B) * (V^2(\tfrac{d^2x}{ds^2}\hat{\imath} + \tfrac{d^2y}{ds^2}\hat{\jmath}) + \dot{V}(\tfrac{dx}{ds}\hat{\imath} + \tfrac{dy}{ds}\hat{\jmath}))]_y$$

$$F_{zr} + F_{zf} + mg - \tfrac{1}{2}\rho C_l A V^2 = 0$$

where $Rot(I \rightarrow B)$ is the rotation matrix from the inertial frame to the vehicle frame, $\frac{1}{2}\rho C_d A V^2$ is the force applied to the vehicle due to aerodynamic resistance, and $\frac{1}{2}\rho C_l A V^2$ is the downforce applied to the vehicle.
Similarly, the equations describing rotational motion on the vehicle frame can be summed up to:

$$\sum M_B = I_B \dot{\omega}_B = I_B \ddot{\Theta}$$

$$F_{yf} l_a - F_{yr} l_b = I_B(\tfrac{d\Theta}{ds}\dot{V} + \tfrac{d^2\Theta}{ds^2}V^2)$$

where Θ is the orientation in space of the vehicle and of the vector tangent to the path, I_B is the moment of inertia of the vehicle on the z axis and l_a, l_b are the distances of the CoG from the front and the rear axle, respectively.
With γ denoting the percentage of the total drive or braking force allocated to the front axle and ζ denoting the percentage of the total downforce allocated to the front axle, the above equations can be solved for $F_{xr}, F_{xf}, F_{yr}, F_{yf}, F_{zr}, F_{zf}$:

$$F_{xf} = \gamma(m\dot{V} + \tfrac{1}{2}\rho C_d A V^2)$$

$$F_{xr} = (1-\gamma)(m\dot{V} + \tfrac{1}{2}\rho C_d A V^2)$$

$$F_{yf} = \tfrac{l_b m}{l_a + l_b}\tfrac{d^2y}{ds^2}V^2 + \tfrac{1}{l_a + l_b}(I_B\tfrac{d^2\Theta}{ds^2}V^2 + I_B\tfrac{d\Theta}{ds}\dot{V})$$

$$F_{yr} = \tfrac{l_a m}{l_a + l_b}\tfrac{d^2y}{ds^2}V^2 - \tfrac{1}{l_a + l_b}(I_B\tfrac{d^2\Theta}{ds^2}V^2 + I_B\tfrac{d\Theta}{ds}\dot{V})$$

$$F_{zf} = -\tfrac{l_b m}{l_a + l_b}g + \tfrac{h}{l_a + l_b}(F_{xr} + F_{xf}) + \zeta(\tfrac{1}{2}\rho C_l A V^2)$$

$$F_{zr} = -\tfrac{l_b m}{l_a + l_b}g - \tfrac{h}{l_a + l_b}(F_{xr} + F_{xf}) + (1-\zeta)(\tfrac{1}{2}\rho C_l A V^2)$$

where h is the height of the vehicle's CoG and load transfer between the front and the rear axle is taken into consideration.

To determine the velocity profile of the vehicle, the capabilities of the tires to generate forces must be translated into acceleration limits for the vehicle and then into maximum safe speeds. Using a friction circle model for the tires, the constraints are given by:

$$F_{xi}^2 + F_{yi}^2 \leq (\mu F_{zi})^2; i = f, r$$

The theoretical maximum speed at each point in the trajectory can be calculated by solving for V^2, while assuming that the tires produce the maximum amount of force ($F_{xi}^2 + F_{yi}^2 = (\mu F_{zi})^2; i = f, r$) and that there is no acceleration taking place ($\dot{V} = 0$). These calculations, however, only take into consideration the local curvature at each point while neglecting the maximum acceleration/deceleration that can take place between two points. To address that, the forward-backward pass technique described in [37] is adopted.

In the forward pass, the new speed upper limits are updated in the following manner:

$$V_n^2 = V_{n-1}^2 + 2\dot{V}_{max}(V_{n-1})ds$$

Note that $\dot{V}_{max}(V_{n-1})$ is a function of V_{n-1}, as it is calculated by solving the above equations for $\dot{V}$, while assuming that the tires produce the maximum amount of force ($F_{xi}^2 + F_{yi}^2 = (\mu F_{zi})^2; i = f, r$) and that the current speed is equal to V_{n-1}.

Similarly, in the backward pass, the new speed upper limits are updated in the following manner:

$$V_{n-1}^2 = V_n^2 - 2\dot{V}_{min}(V_n)ds$$

It is worth noting that whenever the equations are solved for either $\dot{V}$ or V^2, there arise two solutions, one for each axle. To ensure safety, the minimum (in terms of absolute value) of the two values is selected. Additionally, in the case of unknown circuits, the terminal velocity over a path is set to be equal to 0 so that no matter what lies on the edge of the map, there will be sufficient time for the vehicle to slow down, or even, stop.

In the end, the minimum speed upper limit from the two passes is kept for each point on the trajectory, thereby formulating a velocity profile that can be used for trajectory tracking.

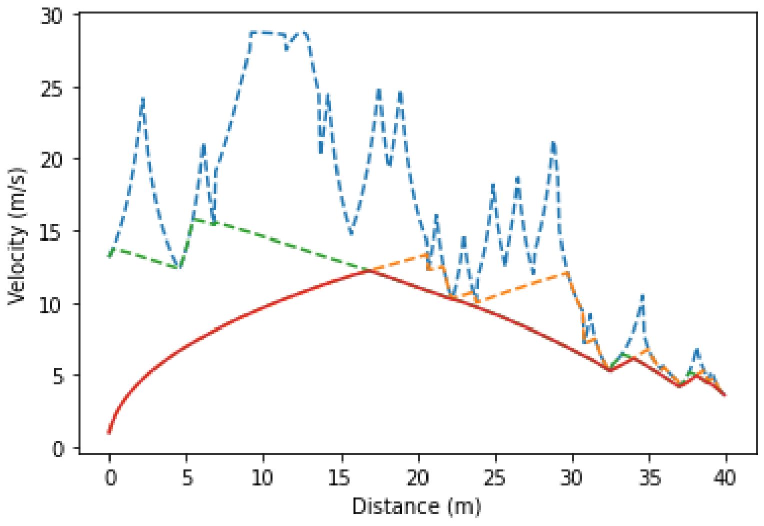


Figure 7.3: The velocity profile of the first 40m of a Formula Student circuit. The blue dashed line is the upper limit based on curvature, the orange dashed line is the upper limit based on the forward pass, the green dashed line is the upper limit based on the backward pass, and the red line is the final velocity profile.

7.3 Trajectory tracking

As previously mentioned, a decoupled approach is adopted for the control of the vehicle. A PID and a Pure Pursuit controller have been implemented for the longitudinal and lateral control of the vehicle, respectively.

The PID controller regulates the vehicle's velocity. The target speed is imposed by the velocity profile that is described above. The integral, derivative, and proportional constants are tuned using the Ziegler-Nichols method[4]. Provided that the PID controller does not take the dynamics into consideration, there is a chance that the torque commands it outputs lead to loss of traction. To address that, a minor check is added before the command gets published. Since both the drive and braking take place at the rear axle, this should be the only axle to check for loss of traction. The information coming out of the velocity estimation module is sufficient to calculate the longitudinal acceleration of the vehicle and the slip angle of the rear tires and, subsequently, the lateral and normal forces acting on said tires. Using the same tire friction model as above, the maximum longitudinal force that the tires can produce can be calculated in order to act as the upper limit of the requested drive or brake torque.

The Pure Pursuit controller sets up targets along the designed trajectory and calculates the steering angle that will allow the vehicle to reach them. One important parameter of any Pure Pursuit controller is the look-ahead distance, which is the distance between the vehicle and the target. Large look-ahead distances lead to smoother trajectories but also increase the probability of a corner being cut, making them ideal for straights. On the other hand, small look-ahead distances often lead to excessive oscillations around the designed trajectory but they are less likely to cut through corners, making them ideal for turns. Given that Formula Student circuits combine straights and turns, a variable look-ahead distance would solve the problem. The look-ahead distance is set to be a function of the target velocity at the vehicle's current position. High target velocities indicate that there is no need for braking and, therefore, turning in the immediate future, while low target velocities indicate that a braking zone and, therefore, turn is impending even if the vehicle is currently passing through a straight. This makes it a more ideal quantity to parameterize the look-ahead distance on instead of other quantities like the local track curvature. Similarly to the PID controller, a check for loss of friction is added before the command gets published.

7.4 Known tracks

In the case of known tracks much of the work is completed offline to reduce the computational burden. Specifically, the centerline of the circuit is extracted offline as well as its velocity profile. That leaves trajectory tracking as the sole process to be running online on known tracks.

7.5 Other approaches and future work

Dampening the drive, braking, or steering commands whenever a loss of traction is detected prevents the vehicle from reaching the target, jeopardizing the run in the process. This can be avoided by combining path planning and control into a single process.

7.5.1 Probabilistic Roadmap

One way to achieve this is by employing a kinodynamic Probabilistic Roadmap (PRM)[22]. The PRM initially samples the circuit and then uses a steering function to connect the samples, thereby forming a graph of potential paths. To include dynamics in this method, the samples have to contain not only x, y coordinates, but also the orientation of the vehicle, θ, and the velocities of the vehicle, u_x, u_y, ω. Efforts to sample on the six dimensional space, however, revealed that the memory capacity required is immense and that the graph size becomes so large that it becomes computationally impossible to run searches for paths in real time. Guided sampling or sample dimensionality reduction may make this method feasible, though.

7.5.2 Model Predictive Control

The most common way of combining path planning and control into a single process is through the use of Model Predictive Control (MPC). Examples of this in Formula Student settings can be found in [19] and [40].

Similar to the time optimal control problem solved offline in [40], a MPC on the space domain is used, which constitutes the time-optimal formulation of the problem.

Instead of the usual coordinates x, y, θ, the curvilinear states s, n, μ are employed for describing the location of the vehicle on the circuit. State s describes the progress (arc-length) along the centerline, state n expresses the orthogonal deviation from the centerline, and state μ is the local heading angle. In the time domain, the dynamic equations are based on a dynamic bicycle model and are as follows:

$$\dot{s} = \frac{u_x cos(\mu) - u_y sin(\mu)}{1 - n\kappa(s)}$$

$$\dot{n} = u_x sin(\mu) + u_y cos(\mu)$$

$$\dot{\mu} = \omega - \kappa(s)\dot{s}$$

$$\dot{u_x} = \frac{1}{m}(F_x - F_{y,F} sin(\delta) + m u_y \omega)$$

$$\dot{u_y} = \frac{1}{m}(F_{y,R} + F_{y,F} cos(\delta) - m u_x \omega)$$

$$\dot{\omega} = \frac{1}{I_z}(F_{y,F} l_a cos(\delta) - F_{y,R} l_b)$$

$\dot{\delta} = \Delta\delta$

The lateral forces are calculated based on the Pacejka tire formula[28]. Data from the FSAE Tire Test Consortium are utilized for the parameter fitting. Assuming no slip, the longitudinal force is set to be the sum of the drive force due to the motor torque, the tires' rolling resistance, and the drag force.

Since the change in motor torque can be considered instantaneous there is no need to include its rate of change in the state vector. On the other hand, the steering angle can change at a significantly lower pace. Therefore, its rate of change is included in the state vector so that it can be used on the constraints of the MPC problem.

To formulate the time-optimal problem a transformation to the space-domain is required. This transformation can be performed using the following reformulation:

$\frac{dx}{dt} = f(x, u)$

$\frac{dx}{ds}\frac{ds}{dt} = f(x, u)$

$\frac{dx}{ds} = \frac{1}{\dot{s}} f(x, u)$

As a result, the dynamic equations become:

$\frac{dn}{ds} = \frac{1}{\dot{s}} \frac{(u_x sin(\mu) + u_y cos(\mu))}{1 - n\kappa}$

$\frac{d\mu}{ds} = \frac{1}{\dot{s}}(\omega - \kappa\dot{s})$

$\frac{du_x}{ds} = \frac{1}{\dot{s}}\frac{1}{m}(F_x - F_{y,F} sin(\delta) + m u_y \omega)$

$\frac{du_y}{ds} = \frac{1}{\dot{s}}\frac{1}{m}(F_{y,R} + F_{y,F} cos(\delta) - m u_x \omega)$

$\frac{d\omega}{ds} = \frac{1}{\dot{s}}\frac{1}{I_z}(F_{y,F} l_a cos(\delta) - F_{y,R} l_b)$

$\frac{d\delta}{ds} = \frac{1}{\dot{s}}\Delta\delta$

Three types of hard constraints are utilized in this formulation. The first one requires the forces on each tire to not surpass the limits based on the friction ellipse. The second one ensures that the orthogonal deviation from the centerline remains below a threshold to avoid exiting the track. The last type of constraints is the result of the steering actuator's mechanical limits, keeping the required changes in steering angle feasible.

With the time interval between two consecutive steps being equal to $\frac{\Delta s}{\dot{s}}$, time optimality can be achieved by including the sum of time intervals in the cost function.

Convergence on the FORCES NLP solver[45] has not been achieved on a steady basis. Scaling of the state vector and conversion of hard constraints to soft ones were introduced, but the issue persisted. However, convergence has been achieved on the CasADi[2] solver. The resulting trajectory on an oval circuit is demonstrated in figure 7.4. The issue with the CasADi solver, however, is that it cannot be used online as solving the MPC problem takes more than 1 second.

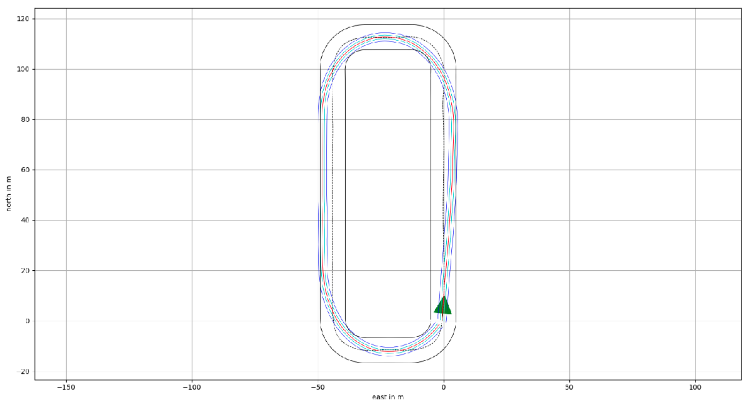


Figure 7.4: Time-optimal MPC's computed trajectory on an oval track.

Finally, there is a way for path planning and control to remain decoupled while keeping the risk of runs resulting in failure low. For the trajectory planning, the same methods as above can be used to set the target velocities at several samples of the centerline. Then a MPC on the space domain can be used for trajectory tracking. Similarly to the previous MPC approach, any efforts have failed to achieve convergence so far.

Chapter 8

New perception pipeline

While testing the vehicle for the 2023 season it became obvious that the existing perception pipeline acted as the bottleneck of the entire autonomous system. With the range RMSE exploding after the first 10 meters and the SLAM allowing only cones that were observed several times to enter the map, running at high speeds remains prohibitive even when the regulations require it. Additionally, the keypoints approaches overly relied on the cones adhering to a specific design. In other competitions, where the cones are of a slightly different design, or even in the scenario where the competition's cone supplier changed, these methods run a high risk of simply ceasing to work.

To tackle the aforementioned issues, a novel perception pipeline is proposed that is based on the YOLOv8 architecture. To take advantage of the features already extracted by the YOLOv8 backbone, an extra head is added to the structure with the sole purpose of calculating the distance to the cone included in each bounding box. To train the augmented YOLOv8[38] network, response-based knowledge distillation is employed.

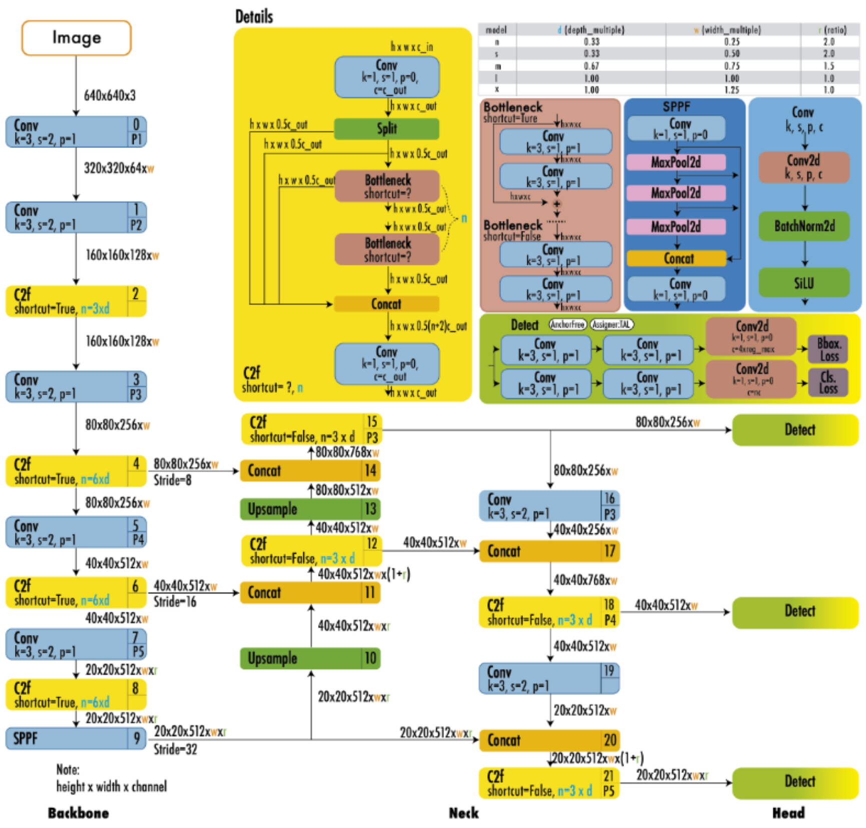
8.1 Theoretical background

8.1.1 YOLOv8

YOLOv8 was released in January 2023 by Ultralytics, the company that developed YOLOv5. It shares a similar backbone with its predecessor, but it uses a decoupled head to independently calculate class probabilities and bounding boxes. This configuration enables individual branches to concentrate on their respective tasks, thereby enhancing the overall accuracy of the model.

In YOLOv8's output layer, the activation function for the objectness score employs the sigmoid function, indicating the likelihood that the bounding box encompasses an object. Additionally, the softmax function is utilized for the class probabilities, reflecting the likelihood of objects belonging to each potential class. YOLOv8 uses CIoU and DFL loss functions for bounding box loss and binary cross-entropy for classification loss, boosting its object detection performance, particularly in dealing with smaller objects.

The overall structure of YOLOv8 can be seen in figure 8.1.



model	d (depth_multiple)	w (width_multiple)	r (ratio)
n	0.33	0.25	2.0
s	0.33	0.50	2.0
m	0.67	0.75	1.5
l	1.00	1.00	1.0
x	1.00	1.25	1.0

Figure 8.1: YOLOv8 architecture[38].

8.1.2 Multitask learning

In Single Task Learning (STL), each neural network is a function of the same inputs and has one output serving the assigned task. Backpropagation is applied to these networks by training each of them in isolation. Since they are not connected, it is not possible for knowledge acquired by one network to be communicated to another.

In Multitask Learning (MTL)[9], a common neural network is used with outputs equal to the total number of assigned tasks. Backpropagation is done in parallel on all the outputs in the network. Because the outputs share a common hidden layer, it is possible for internal representations that arise in the hidden layer for one task to be used by other tasks. Sharing what is learned by different tasks while tasks are trained in parallel is the central idea in MTL. In other words, MTL functions as an inductive transfer mechanism aimed at enhancing generalization performance. The training signals from additional tasks act as an inductive bias, contributing to improved generalization.

8.1.3 Knowledge distillation

To develop efficient deep neural networks, recent works usually focus on efficient building blocks for deep models and on model compression and acceleration techniques. One such technique is Knowledge Distillation (KD)[17], which involves distilling knowledge from a larger deep neural network into a small network.

One commonly used type of KD is response-based KD. In this case, the main idea is to directly mimic the final prediction of a larger teacher model. Despite its simplicity, response-based KD is effective for model compression and has been widely used in different tasks and applications.

Response-based knowledge distillation can be applied to diverse model prediction scenarios. For instance, in object detection tasks, the response may encompass logits along with the bounding box offset. In semantic landmark localization tasks, such as human pose estimation, the teacher model's response might involve heatmaps for each landmark. More recently, researchers have delved deeper into response-based knowledge, extending its use to incorporate ground-truth labels as conditional targets.

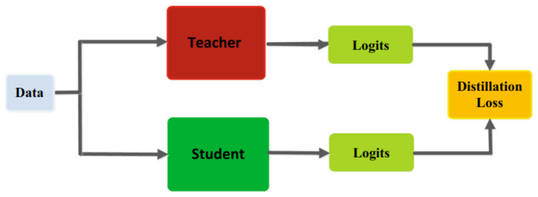


Figure 8.2: The generic response-based knowledge distillation.

8.1.4 Related work

Modified YOLO networks that estimate the distance of the objects in the bounding boxes have already been used. In [39], the authors extended the prediction vector of YOLOv3 to facilitate depth estimation. For training, they used data from the KITTI[14] 3D Object Detection Evaluation 2017 dataset, which consists of 7481 training images and 7518 test images with corresponding point clouds and calibration matrices. Similarly in [44], the authors added a head on the YOLOv4 architecture tasked with depth estimation and used the same dataset for training it.

8.2 Implementation

8.2.1 Dataset

Unlike the case of regular YOLO, where there was a publicly available dataset with images of cones alongside the respective ground truth bounding boxes and classes, there is no such dataset that includes the distances to the cones. To tackle this issue there are three options: 1) use a general purpose dataset with all the required information like KITTI, 2) set up a custom dataset or 3) use the inferences of a teacher model as ground truth.

Regarding the first option, due to the popularity of the monocular depth estimation problem, there are ample trained networks on these datasets that predict depth maps to experiment with. One such network is MiDaS[29] which was used to evaluate the potential of this option. Use of the dpt-large-384 variant resulted in an RMSE of 1.05 and 1.63 meters for cones in 0-5m and 5-10m, respectively. Since this is significantly outperformed by the current pipeline, this approach was abandoned.

Adopting the second option brings a higher level of customization, which is very important considering the specificity of the problem. However, at the same time, it entails going through a gruesome process of creating a custom dataset. With the augmented YOLO having 3.3M parameters, 1.1M of which belong to the Detect layer which cannot be frozen, the dataset needs to contain thousands of pictures. Consequently, this approach was abandoned for practical reasons.

Choosing the third option, the inferences of a teacher model on unlabeled images are used as a form of response-based knowledge distillation. The teacher model shares the same overall structure with the best performing pipeline of chapter 5. To achieve superior performance without taking inference time or memory footprint into consideration, YOLOv5n is swapped for YOLOv5m6 and the keypoint detection network sees an increase in both layers and channels. Its accuracy compared to the other pipelines can be seen in figure 8.3. Using this method, a dataset of over 5000 images is created.

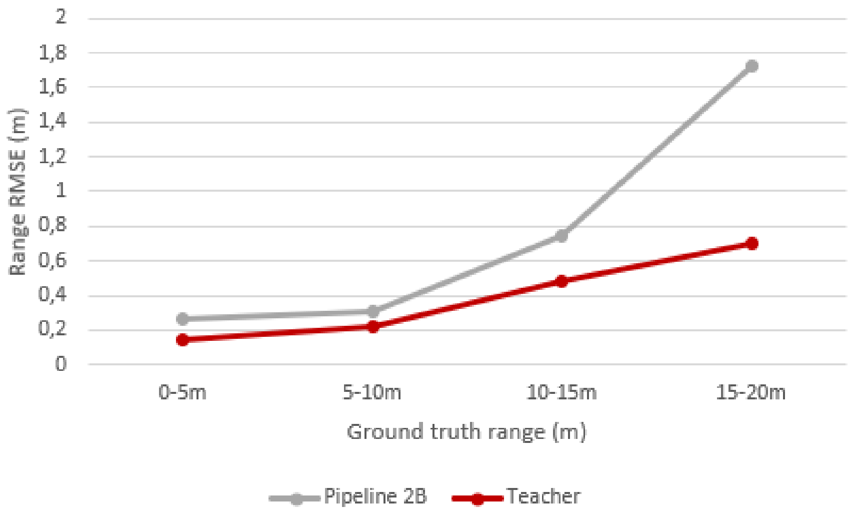


Figure 8.3: Comparison of the reprojection error pipeline (2B) and the teacher model in terms of range RMSE.

8.2.2 YOLOv8 modification

Given the limited memory capacity of the TPU, the YOLOv8n variant is used. As shown in figure 8.1, YOLOv8n features two 3-layer-deep convolutional neural networks on the Detect layer, the tip of its head, for computing class probabilities and bounding box coordinates.

Following the same pattern, a 4-layer-deep convolutional neural network is added in parallel to the aforementioned two, sharing the same inputs from YOLOv8n's neck. Similar to the bounding box coordinates neural network, no activation function is applied to the final output of the network. Apart from the last convolutional layer, each layer is followed by a batch normalization layer and a SiLU activation function.

The loss function is modified to include the depth estimation task. Mean Squared Error (MSE) loss is chosen and, after applying a weight to it, it is simply added to the sum of all pre-existing losses.

8.2.3 Training

YOLOv8n is first trained on the FSOCO dataset[41]. Even after the head is modified, the weights from the backbone and the neck of the network can still be transferred for future trainings. Afterwards, the augmented YOLO is trained on the inference of the teacher model.

8.3 Results

The augmented YOLO is tested on the same dataset as the other perception pipelines. It slightly outperforms the previous pipeline for cones within 15m, but the biggest gain is spotted on cones 15-20m away from the camera. Its performance relative to that of the previous pipeline and the teacher model is depicted in figure 8.4. What is also impressive is that the recall rate has risen from 68.3% to 94.4%.

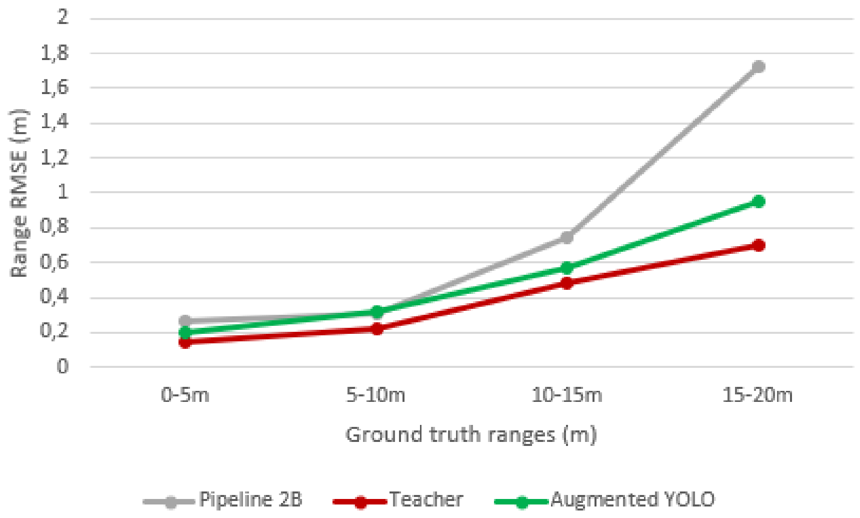


Figure 8.4: Comparison of the augmented YOLO, the previous pipeline, and the teacher model in terms of range RMSE.

The impact of the new perception pipeline is also evident on the results of SLAM. Recall rises from 94.7% to 96.5% and precision rises from 97.3% to 100%. A comparison of the map produced using the old and the new perception pipeline is shown in figure 8.5.

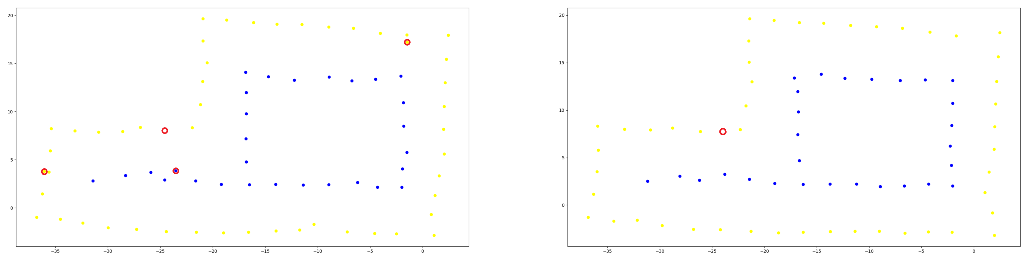


Figure 8.5: The map produced by the previous perception pipeline (left) and the map produced by the new perception pipeline (right). False positives and negatives are noted with red circles.

Finally, the inference time of the entire pipeline amounts to 26ms per image. That constitutes a massive improvement in comparison to the previous pipeline that took 25ms to run YOLOv5n and roughly 3ms per cone detected.

Chapter 9

Future Work

The design of the autonomous system of P23 focuses primarily on reliability. As the next iterations will progressively shift their focus to performance, there are several alternative approaches to consider for each of the four modules.

9.1 Velocity Estimation

The Extended Kalman filter is characterized by particularly poor performance when the state transition and observation models are highly non-linear. This is because the covariance is propagated through linearization of the underlying nonlinear model. To tackle this issue, an Unscented Kalman filter[42] or a particle filter[13] can be used in its place.

The Extended Kalman filter that is currently in use employs a kinematic model for the prediction step. A dynamic, bicycle or two-track, model can be used instead. This will lead to a significant improvement in accuracy, especially in cases where the accelerations are further away from zero.

Finally, the current Velocity Estimator does not filter out any outlier measurements from the sensors. Outlier rejection based on the chi-squared test is a potential solution to this problem and has even been proposed for a similar setting[19].

9.2 Perception

In its current form, the augmented YOLO is trained solely on data from the camera setup of P23. This means that the same process would have to be repeated should the position and/or orientation of the cameras changed. Progressively, however, by alternating the camera's place on the vehicle, enough data can be gathered for the pipeline to work regardless of the camera setup.

On a similar note, all the training data originate from the same camera-lens pairing, meaning that a change of camera and/or lens might cause the pipeline to cease functioning properly. To tackle that, a relative depth metric has to be adopted that is invariant to camera intrinsic parameters, just like the one used in the depth maps of MiDaS[29].

The training dataset is the result of inferences stemming from a teacher model and, therefore, is inherently as inaccurate as the teacher model itself. Given reliable odometry measurements, the graph SLAM can refine the perception edges, leading to higher quality ground truth data on the respective images. It is worth noting that the distances in this case will be between the vehicle's center of mass and the cone and, consequently, the transition from the camera to the vehicle frame will no longer be required.

9.3 SLAM

The main potential improvements on the SLAM module are related to the process of data association. The Nearest Neighbor algorithm that is used allows for two observed cones to be associated to the same cone on the map. To avoid this, the Joint Compatibility Branch and Bound (JCBB) algorithm[25] can be used its place. Additionally, since the perception measurements' covariance matrix can be estimated, Euclidean distance can be exchanged for Mahalanobis distance, which incorporates the uncertainty of the measurements when matching observed with mapped cones.

9.4 Motion Planning

Moving forward, the Model Predictive Control approach needs to be adopted as it presents a unified framework for optimal path calculation while taking into consideration the dynamic model of the vehicle and other constraints. In its present form, it is an iterative process of linearizing the system around the current solution and solving again. In case the system is highly non-linear, this process will have a great difficulty converging to a single optimal solution. One way to tackle this is by employing Koopman MPC[24] instead. The state vector, the constraints, and the cost function will be encoded in such a way that the original non-linear MPC problem will transform into a linear one. In the encoded space, the state transition will become a linear process, while the constraints and the cost function will turn into linear functions of the encoded state and its quadratic form, respectively. The resulting linear MPC will then have no issue converging to a single optimal solution.

Bibliography

[1] Bashar Alsadik and Samer Karam. The simultaneous localization and mapping (slam)-an overview. *Surv. Geospat. Eng. J*, 2:34–45, 2021.

[2] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11:1–36, 2019.

[3] Leiv Andresen, Adrian Brandemuehl, Alex Honger, Benson Kuan, Niclas Vödisch, Hermann Blum, Victor Reijgwart, Lukas Bernreiter, Lukas Schaupp, Jen Jen Chung, et al. Accurate mapping and planning for autonomous racing. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4743–4749. IEEE, 2020.

[4] Karl Johan Åström and Tore Hägglund. Revisiting the ziegler–nichols step response method for pid control. *Journal of process control*, 14(6):635–650, 2004.

[5] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot. Consistency of the ekf-slam algorithm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3562–3568. IEEE, 2006.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, pages 404–417. Springer, 2006.

[7] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[8] Gary Bradski. The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123, 2000.

[9] Rich Caruana. Multitask learning. *Machine learning*, 28:41–75, 1997.

[10] Toby Collins and Adrien Bartoli. Infinitesimal plane-based pose estimation. *International journal of computer vision*, 109(3):252–286, 2014.

[11] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[12] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. *Georgia Institute of Technology, Tech. Rep*, 2:4, 2012.

[13] Petar M Djuric, Jayesh H Kotecha, Jianqui Zhang, Yufei Huang, Tadesse Ghirmai, Mónica F Bugallo, and Joaquin Miguez. Particle filtering. *IEEE signal processing magazine*, 20(5):19–38, 2003.

[14] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[17] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

[18] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N Zeilinger. Learning-based model predictive control for autonomous racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, 2019.

[19] Juraj Kabzan, Miguel I Valls, Victor JF Reijgwart, Hubertus FC Hendrikx, Claas Ehmke, Manish Prajapat, Andreas Bühler, Nikhil Gosala, Mehak Gupta, Ramya Sivanesan, et al. Amz driverless: The full autonomous racing system. *Journal of Field Robotics*, 37(7):1267–1294, 2020.

[20] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[21] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[22] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[23] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II. IEEE, 2004.

[24] Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, 2018.

[25] Nick Le Large, Frank Bieder, and Martin Lauer. Comparison of different slam approaches for a driverless race car. *tm-Technisches Messen*, 88(4):227–236, 2021.

[26] Der-Tsai Lee and Bruce J Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980.

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[28] Hans B Pacejka and Egbert Bakker. The magic formula tyre model. *Vehicle system dynamics*, 21(S1):1–18, 1992.

[29] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020.

[30] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[31] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[32] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[33] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[34] Punam K Saha, Gunilla Borgefors, and Gabriella Sanniti di Baja. Skeletonization and its applications–a review. *Skeletonization*, pages 3–42, 2017.

[35] Sirish Srinivasan, Inkyu Sa, Alex Zyner, Victor Reijgwart, Miguel I Valls, and Roland Siegwart. End-to-end velocity estimation for autonomous racing. *IEEE Robotics and Automation Letters*, 5(4):6869–6875, 2020.

[36] Kieran Strobel, Sibo Zhu, Raphael Chang, and Skanda Koppula. Accurate, low-latency visual perception for autonomous racing: Challenges, mechanisms, and practical solutions. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1969–1975. IEEE, 2020.

[37] John Subosits and J Christian Gerdes. Autonomous vehicle control for emergency maneuvers: The effect of topography. In *2015 American Control Conference (ACC)*, pages 1405–1410. IEEE, 2015.

[38] Juan Terven and Diana Cordova-Esparza. A comprehensive review of yolo: From yolov1 to yolov8 and beyond. *arXiv preprint arXiv:2304.00501*, 2023.

[39] Marek Vajgl, Petr Hurtik, and Tomáš Nejezchleba. Dist-yolo: fast object detection with distance estimation. *Applied sciences*, 12(3):1354, 2022.

[40] José L Vázquez, Marius Brühlmeier, Alexander Liniger, Alisa Rupenyan, and John Lygeros. Optimization-based hierarchical motion planning for autonomous racing. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2397–2403. IEEE, 2020.

[41] Niclas Vödisch, David Dodel, and Michael Schötz. Fsoco: The formula student objects in context dataset. *arXiv preprint arXiv:2012.07139*, 2020.

[42] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000.

[43] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[44] Jongsub Yu and Hyukdoo Choi. Yolo mde: Object detection with monocular depth estimation. *Electronics*, 11(1):76, 2021.

[45] Andrea Zanelli, Alexander Domahidi, Juan Jerez, and Manfred Morari. Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, 93(1):13–29, 2020.