Combination of Stochastic Model Predictive Control and Vision System on a Donkey Car for Circuit Following Task

Ambre Ricouard
ambre.ricouard@ensta-bretagne.org

October 1, 2024

Ensta Bretagne
2, rue François Verny
29806 Brest cedex
France
Tel +33 (0)2 98 34 88 00
www.ensta-bretagne.fr

Abstract

This project focuses on automating a small-scale model car to navigate a fixed circuit using only a camera for guidance. The objective is for the car to follow a track centered between two guiding lines. The circuit used in reality is also available in simulation, allowing for process validation before testing in a real environment where significant visual noise may be present.

A crucial initial step is to examine the system specifications and camera limitations to anticipate all possible scenarios, thereby ensuring that the developed controllers are robust. Depending on the car's position and orientation relative to the track, it may detect both lines, only one line, or even none. Therefore, the system must be able to handle each of these situations effectively.

The process begins with image processing, where the visible lines are detected to determine the car's orientation and desired future position. With this information in hand, the next step is to address vehicle control. This requires establishing the car's kinematic model, which involves defining a projection frame and understanding how the system inputs relate to the vehicle's kinematics.

Initially, a simple Proportional-Integral-Derivative (PID) controller is implemented. While this provides a good starting point, it is inherently limited as it only accounts for past positions. To improve system robustness, a stochastic Model Predictive Control (MPC) algorithm is introduced, enabling for the prediction of future trajectories and more effective decision-making.

Résumé

Ce projet se concentre sur l'automatisation d'une voiture miniature naviguant sur un circuit fixe, en utilisant une caméra comme unique capteur. L'objectif est que la voiture suive une piste délimitée par deux lignes continues. Le circuit réel est également disponible en simulation, permettant de valider les traitements avant de les tester dans l'environnement réel, où le bruit visuel est significatif.

La première étape consiste à examiner les spécifications du système et les limitations de la caméra afin d'anticiper tous les scénarios possibles, assurant ainsi la robustesse des contrôleurs développés. Selon la position et l'orientation de la voiture par rapport à la piste, celle-ci peut détecter les deux lignes, une seule ligne, ou aucune. Le système doit donc être capable de gérer efficacement chacune de ces situations.

Vient ensuite le traitement des images de la caméra pour identifier les lignes visibles, permettant ainsi de déterminer la position et l'orientation futures de la voiture. Avec ces informations, on aborde le contrôle du véhicule, ce qui nécessite d'établir le modèle cinématique de la voiture, incluant la définition d'un repère de projection et la compréhension de la relation entre les entrées et le mouvement du véhicule.

Initialement, un contrôleur Proportionnel-Intégral-Dérivé (PID) simple est mis en œuvre. Bien que cela constitue un bon point de départ, il présente des limitations intrinsèques, car il ne prend en compte que les positions passées de la voiture. Pour améliorer la robustesse du système, un algorithme de contrôle prédictif basé sur une commande prédictive (MPC) est introduite, permettant de prévoir les trajectoires futures et de prendre des décisions plus efficaces.

Keywords

Autonomous Car, Camera, Image Processing, Line Following, PID Control, MPC Control.

Acknowledgements

I would like to express my deepest gratitude to Alexander Watcher and Gerald Ebmer for their invaluable assistance in helping me initiate this project. Their unwavering support and guidance have been instrumental in my progress, as they were always available to answer my questions and provide clarity. I am also profoundly thankful to my internship supervisor, Minh Nhat Vu, whose supervision and insightful advice have been pivotal to the successful completion of this project. Thank you for your dedicated mentorship and continuous encouragement.

Contents

Introduction

Automating vehicles holds the promise of significantly reducing human error, a major factor in road accidents. According to the World Health Organization, approximately 90 % of road traffic accidents are attributed to human error [1]. By eliminating these errors, autonomous vehicles have the potential to save thousands of lives and reduce severe injuries.

However, ensuring that automated systems are truly effective requires them to be both robust and reliable. Achieving this reliability is a complex challenge due to the diverse and variable environments in which vehicles operate. Road conditions, the behavior of other road users, and weather variations make the task particularly difficult.

Given these challenges, real-world testing of these systems is often costly and risky. To address these issues, researchers are increasingly turning to reduced-scale model cars. These smaller models enable safe and cost-effective experimentation while providing valuable insights into the behavior and performance of autonomous driving technologies ([2]). By using scaled-down versions, we can rigorously test and refine autonomous driving systems in a controlled setting, accelerating their development while managing the financial and safety concerns associated with full-scale testing.

The goal of this project is to make an existing reduced-scale model car autonomous, enabling it to navigate a given track using only a camera. The system must be optimized to ensure stability, achieving a smooth trajectory that is resilient to external noise and variations in lighting conditions.

The first step in the process is to thoroughly understand the specific characteristics and limitations of the existing model car system. Once this foundational knowledge is established, the next phase involves processing the camera images to accurately detect the circuit lines. With the track detection in place, a suitable controller must then be designed to effectively manage the car's movements and ensure stable navigation.

In this approach, a PID controller will be employed first, followed by a transition to using a MPC. Starting with the PID controller ensures a simple and straightforward control mechanism. The subsequent implementation of MPC will enhance the system's performance by leveraging predictive capabilities and optimization, allowing for more precise, efficient and robust control. This phased approach provides a balance between simplicity and advanced functionality, ultimately improving the overall system performance.

1 System Overview

1.1 Description of the Car Components

The model car system consists of several key components that work together to enable autonomous navigation. Figure 1 provides a detailed description of each component and its role in the system. All electronic components are controlled by the onboard Raspberry Pi 4 computer.

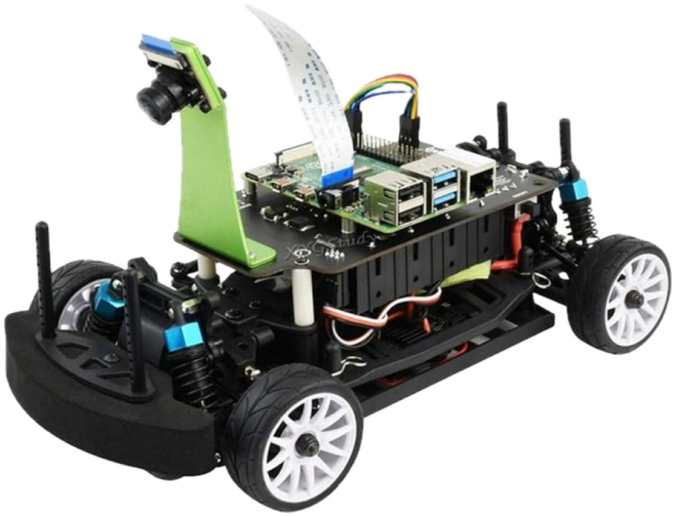


Figure 1 – General view of car.

Component	Description
Chassis	The main frame of the car, which supports all other components.
Driving motor	Electronic Speed Controller (ESC), Rear-Wheel Drive
Servo motor	Steering the front wheels with a range of 90°
Raspberry Pi 4	The onboard computer that processes sensor data and controls the car.
Pi Camera V2	Captures visual data of the track for navigation purposes.

Table 1 – Detailed description of the car's components.

As indicated in Table 1, the system's only sensor is the Pi Camera. Thus, image processing is the only means of determining the car's position relative to the track boundaries. Using the onboard camera, the system captures real-time images that are then processed by the Raspberry Pi.

The car is equipped with two types of motors for control: a servo motor and an ESC. The servo motor is responsible for angular speed, allowing the front wheels to turn and steer the car. The input command for the servo motor, which determines the steering angle, rannonges from -1 to 1. A value of -1 corresponds to maximum left steering, 0 to a neutral position, and 1 to maximum right steering.

The ESC controls the car's linear speed. It also regulates the power sent to the rear wheels, allowing the car to accelerate or decelerate. The input command ranges from 0 to 1, where 0 corresponds to a complete stop and 1 to maximum speed.

These two commands are essential for driving the car, and their combination allows for effective navigation of the track, following the defined trajectory.

The circuit's Command and Control (C2) Architecture, as shown in Appendix A, illustrates the interactions between the car's components and clearly distinguishes the hardware aspects from the observation and control processes.

1.2 Environment and circuit

The real circuit is available in simulation, allowing for an initial focus on achieving the objectives in a controlled environment. This helps mitigate environmental noise and facilitates

effective validation of image processing and control algorithms before applying them in the real world.

(a) Real Circuit (b) Simulated Circuit

Figure 2 – Circuit the car must follow: (a) Real circuit, (b) Simulated circuit

Figure 2 shows that the track the car needs to follow consists of two straight sections interrupted by two U-turns, bordered by two orange lines. A dashed white line runs down the center, but due to the light gray background, the contrast between the white line and the background is low, as seen in the real circuit image (Figure 2a). The contrast appears slightly higher in the simulation (Figure 2b). Additionally, the discontinuity and curvature of the white line make a Canny filter a less optimal choice for detection.

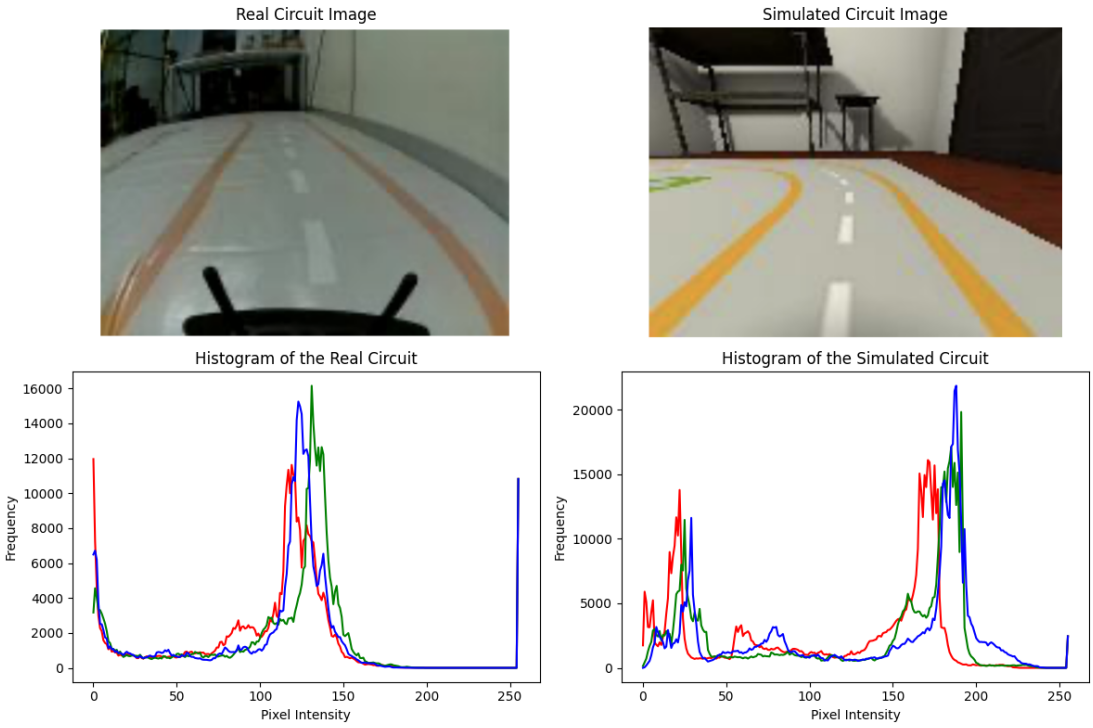


Figure 3 – Comparison of camera imagery between simulated and real environments

The images and histograms in Figure 3 compare the visual characteristics and pixel intensity distributions of the real and simulated circuits as captured by the robot's camera. The histogram of the real circuit exhibits a smoother distribution with fewer distinct peaks, highlighting the impact of real-world lighting variations and noise. In contrast, the histogram of the simulated circuit features more distinct peaks, indicating controlled and consistent lighting in the virtual environment. To ensure reliable performance in real-world conditions, incorporating filters into the robot's vision system is essential to reduce noise and enhance robustness against lighting variations.

1.3 Camera

To establish a reliable and robust controller, it is crucial to understand all possible input data scenarios. This knowledge ensures that the controller is designed to handle all cases effectively, allowing us to anticipate potential issues and implement solutions that enhance its performance and reliability under various conditions.

Characteristics	**Details**
Resolution	8 Megapixels (3280 x 2464 pixels)
Sensor	Sony IMX219
Horizontal Field of View (FOV)	Approximately 62.2°
Vertical FOV	Approximately 48.8°
Focal Length	3.04 mm
Aperture	f/2.0
Lens Type	Fixed
Connection Type	MIPI CSI-2

Characteristics	Details
Image Output Format	RAW
Shutter type	Rolling shutter
Camera Dimensions	Approximately 25 x 24 x 9 mm
Weight	Approximately 3 g

Table 2 – Technical specifications of the Pi Camera V2 used in the car system ([3]).

The camera employs rolling shutter technology, which captures images by scanning the sensor line by line, in contrast to global shutter technology, which captures the entire image simultaneously. In scenarios where the model is moving, the rolling shutter method can theoretically introduce distortions, such as skew or wobble, particularly at high speeds due to the sequential nature of image capture. However, for our small-scale model, the operational speed is relatively low, which minimizes the risk of noticeable distortion. The time required to read out the entire sensor is short enough relative to the model's speed, resulting in negligible positional changes during capture. Consequently, experimental tests have shown no significant image distortion under the current operating conditions. Given the controlled environment and the moderate speed of the model, the rolling shutter is sufficient for our application, balancing image quality, cost, and system complexity effectively.

The camera on the car may not always capture both track lines due to the combined effects of its FOV, the car's position, and its orientation. With a horizontal FOV of 62.2°, as specified in Table 2, the camera covers a limited angular range. When the car is centered and aligned on the track, both lines fall within this range, as observed experimentally. However, if the car deviates from the center or is angled, one line may fall outside the FOV. This issue is particularly noticeable during turns or when the car is closer to one line.

Given the circuit's U-turns, the focus must be on the car's immediate surroundings without anticipating the turns. Attention should be directed to the area between 3 cm and 10 cm in front of the vehicle, based on on-site measurements. The perceivable width limits for the car are as follows

$$\text{Horizontal width at 3 cm} = 2 \times (3\,\text{cm} \times \tan\left(\frac{\text{FOV}}{2}\right)) \approx 3.62\,\text{cm}$$

$$\text{Horizontal width at 10 cm} = 2 \times (10\,\text{cm} \times \tan\left(\frac{\text{FOV}}{2}\right)) \approx 12.06\,\text{cm}$$

Given that the field of view at 3 cm is relatively narrow, the car's vision is restricted, making it difficult to see both lines during turns or even on straight sections if the system tends to oscillate. This limited visibility emphasizes the importance of focusing on the immediate area directly in front of the vehicle, especially when navigating sharp bends like U-turns.

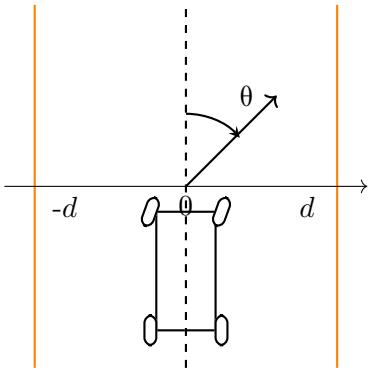


Figure 4 – Definition of θ and the axis d with 0 corresponding to the centerline of the circuit, $-d$ the left line, and d the right line.

Based on Figure 4, let's define θ as the orientation angle of the car relative to the track limits and d as the distance from the center. With these definitions, the interdependence of θ and d can be studied to understand the specific scenario, whether one or two lines are detected, depending on their variations.

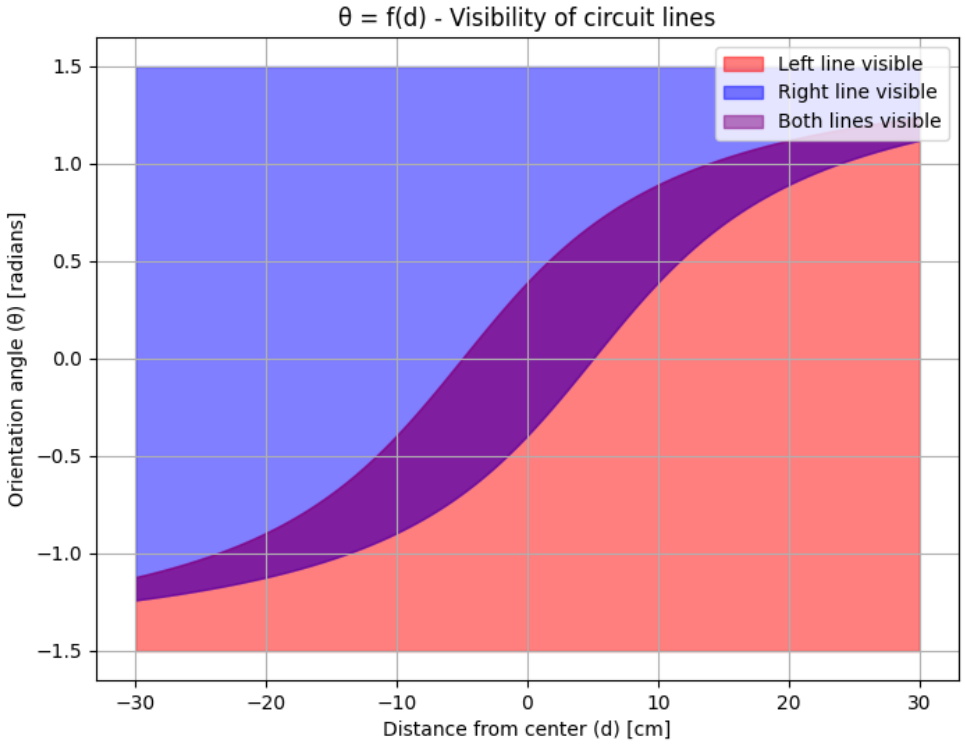


Figure 5 – $\theta = f(d)$ - Visibility of circuit lines as a function of the car's distance from the center and its orientation angle.

According to Figure 5, when positioned at the center ($d = 0$), there is a range of approximately

45.84° for consistently detecting both lines. This range decreases significantly as d moves from 0 to 20 cm, reducing to around 11.46°, indicating a decrease of about 75%.

Therefore, it is crucial to have a controller with minimal oscillation to remain within the detection zone of both lines on the straight portions of the circuit. Otherwise, the system risks constantly switching between detecting one line and the other without ever detecting both simultaneously, which would compromise the controller's robustness. Additionally, it is important to consider scenarios where only one line is detected (e.g., during U-turns) or no lines are detected (e.g., when leaving the circuit).

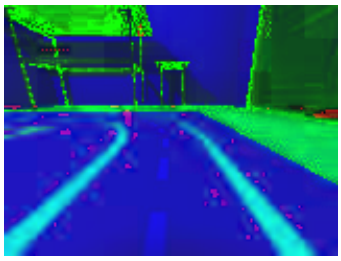
2 Image Processing

2.1 Two Lines Detected

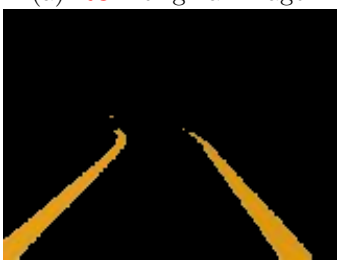
The first approach for detecting the circuit lines would be to use a line detection filter such as the one provided by the Canny algorithm. This algorithm is effective at detecting edges by highlighting sharp transitions between different pixel intensities. However, this method has several drawbacks specific to our context. Firstly, the circuit includes U-turns, which complicates the reliable detection of lines because the Canny algorithm might struggle to follow tight curves. Secondly, the drivable surface of the circuit is rectangular, and the algorithm might mistakenly detect the edges of this surface as relevant lines, which would be an error.

To overcome these issues, we will take advantage of the distinct color contrast between the orange lines of the circuit and the light gray background. By applying a color filter, we can effectively isolate the orange lines without being disturbed by the edges of the circuit or the tight turns. This color-based approach is better suited to our specific situation and should provide more reliable results.

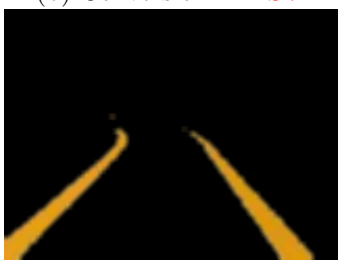
(a) RGB original image



(b) Conversion in HSV



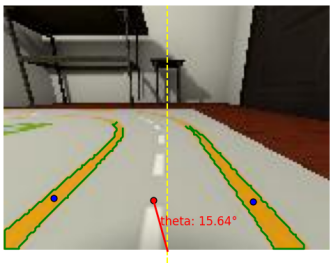
(c) Mask filter



(d) Blur filter

(e) Dilated image

(f) Contours detected



(g) Determination of the angle θ

Figure 6 – Different steps in the image processing pipeline.

Figure 6 provides a detailed, step-by-step visual representation of the image processing workflow. Each step in the figure illustrates a different stage of the process, showing how the image is progressively transformed through various filtering, masking, and contour detection techniques.

The RGB (Red, Green, Blue) image (Figure 6a) is first converted to the HSV (Hue, Saturation, Value) color space (Figure 6b) to facilitate color segmentation. This transformation allows for easier identification and isolation of specific colors within the image. A mask is then applied to isolate the regions containing the orange lines that define the circuit boundaries (Figure 6c).

Subsequently, a blur filter is used to reduce noise and smooth out fine details in the Figure

6d. This step helps in minimizing minor variations and enhancing the prominence of significant features. Following the blurring, the image is dilated to close gaps and unify disjointed regions (Figure 6e). Even though these processing steps might not appear to dramatically alter the image, they play crucial roles in ensuring the robustness and consistency of the subsequent analysis.

Contour detection is then performed on the processed image (Figure 6f). This step identifies the edges of objects within the image, and the two largest contours are retained as the circuit lines.

For each of these contours, the moment is calculated (blue points on Figure 6g). The average of these two centroids determines the target point (red point). Finally, θ, defined as the angle between the vertical line from the camera's position (the bottom center of the image) to the target point, is calculated. This angle is crucial for understanding the vehicle's orientation relative to the track.

2.2 One Line Detected

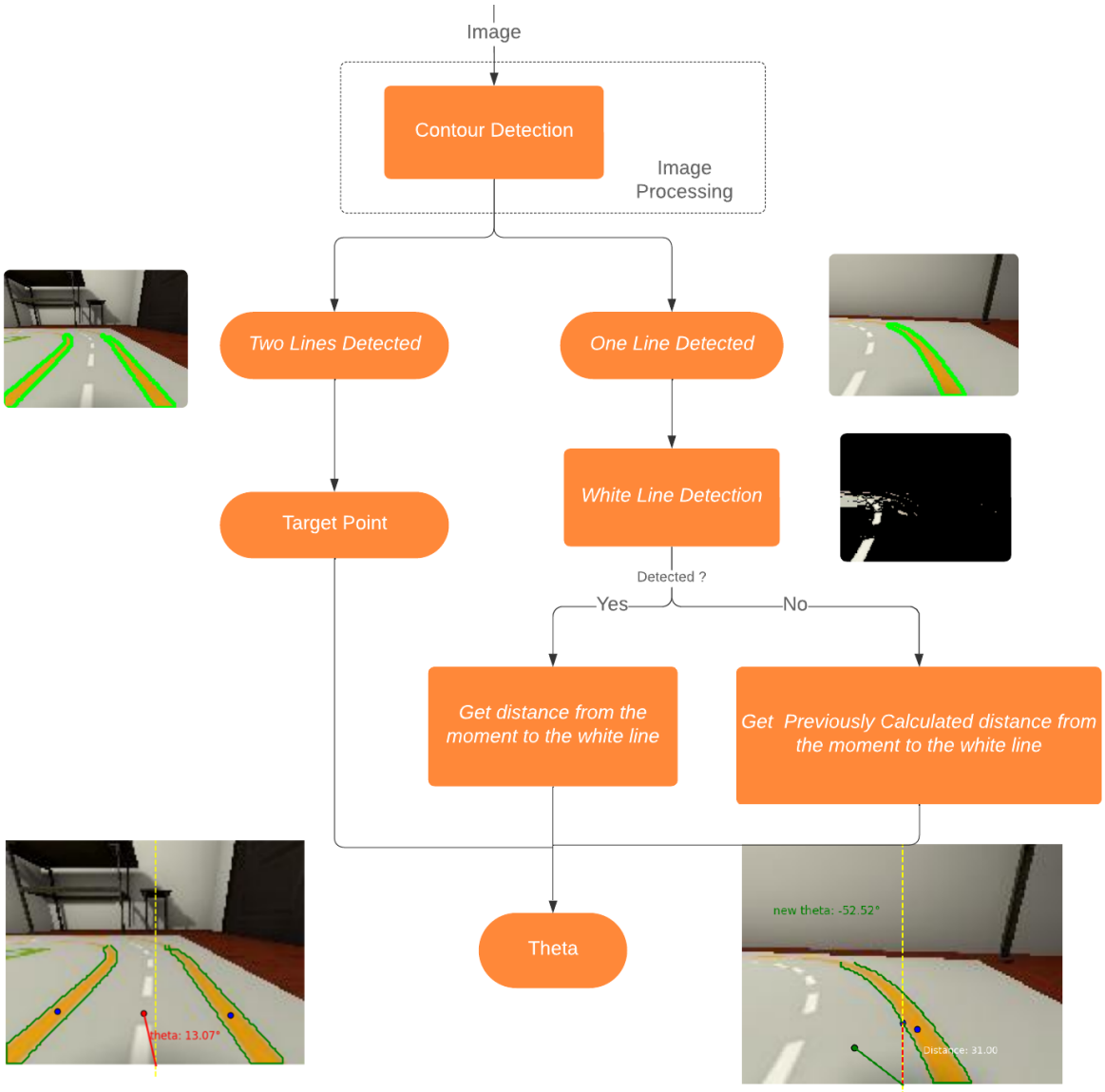


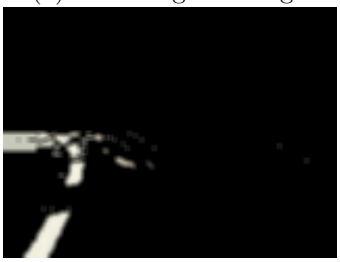
Figure 7 – Decision diagram for determining θ based on line and white line detection.

The objective of the image processing is to determine the angle θ. Figure 7 illustrates the process for determining this angle based on the configuration. When both lines are detected, θ is calculated using two parameters: the vertical axis of the car and the target point. In cases where only one line is detected, an additional data point is needed. To acquire this missing data, the algorithm attempts to detect the white dashed line on image 8a (Figure 8). This involves additional processing steps similar to the previous ones, including masking (8b), blurring (8c), and dilation (8c). The white contours of the dashed line are then used to reconstruct the median white line through linear regression (**??**). The second point is then determined by selecting the point on the line that has the same ordinate as the centroid of the detected contour. If case the white line is poorly detected, the algorithm defaults to using the white line detected in the previous frame; if this is not available, it uses the midpoint of the image's width. If the car

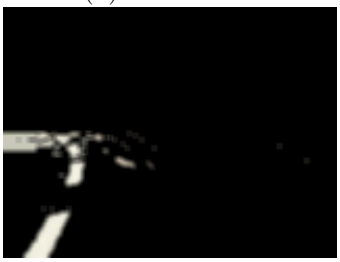
deviates and leaves the circuit to the extent that it no longer detects any lines, one solution is to retain the last value of θ and increment it slightly to allow the car to find its way back. However, given the narrowness of the circuit, two scenarios are common: either the car takes a considerable amount of time to recalibrate and regain its trajectory, or it has already collided with an obstacle, such as a wall or another surrounding object.

(a) RGB original image

(b) Mask filter

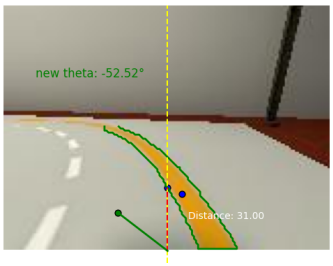


(c) Blur filter



(d) Dilated image

(e) Linear regression of white contours.



(f) Determination of the angle θ

Figure 8 – Different steps in the image processing pipeline.

3 Kinematics Model

To determine the kinematic model of the system, it is essential to understand the roles of the throttle input v and the steering angle ψ. Unlike differential drive robots, cars have four wheels, with the two front wheels dedicated to steering and the two rear wheels to propulsion. We will examine how these inputs influence the overall movement of the vehicle.

3.1 Rear Wheels Analysis and Throttle

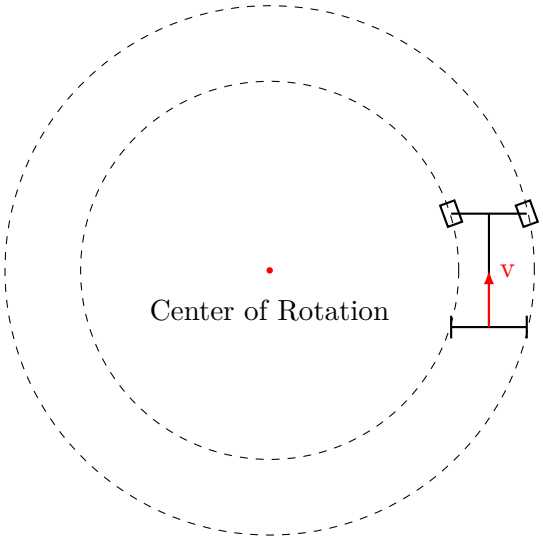


Figure 9 – Diagram illustrating the kinematics of the rear wheels during a turn with throttle engagement.

When the vehicle turns, each wheel follows a circular path around a center of rotation (Figure 9). Since the rear wheels are mounted on the same axle, they share identical movement characteristics relative to this center. The rear axle aligns with the radii of these concentric circles. Consequently, the velocity of each rear wheel can be represented as a vector originating from the center of the axle and directed perpendicular to it.

3.2 Front Wheel Analysis and Steering Angles

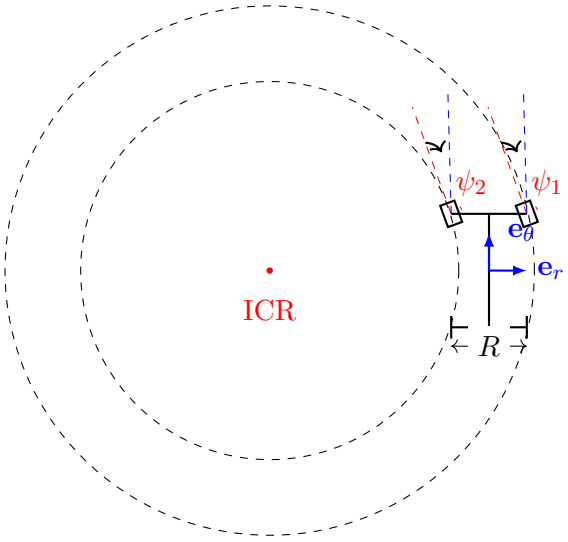


Figure 10 – Diagram illustrating the orientations of the front wheels in a polar system.

The polar coordinate system is centered at the front axle of the vehicle (shifted in Figure 10 for clarity). Thus, the velocities are $\dot{r}\mathbf{e}_r + (r - \frac{R}{2})\dot{\psi}_1\mathbf{e}_\theta$ for the inner wheel and $\dot{r}\mathbf{e}_r + (r + \frac{R}{2})\dot{\psi}_2\mathbf{e}_\theta$ for the outer wheel.

To avoid slipping while turning, the front wheels must be tangent to concentric circles centered at a common point, known as the Instantaneous Center of Rotation (ICR). Thus, the angular velocities $\dot{\psi}_1$ and $\dot{\psi}_2$ must satisfy:

$$(r - \frac{R}{2})\dot{\psi}_1 = (r + \frac{R}{2})\dot{\psi}_2$$

This equation shows that the inner wheel must turn at a sharper angle than the outer wheel. This configuration allows the car to turn smoothly along a defined circular path without tire slip.

According to [4] the difference in front angles is known as the Ackermann angle. This can be more easily modeled by introducing a virtual central wheel, positioned between the two front wheels.

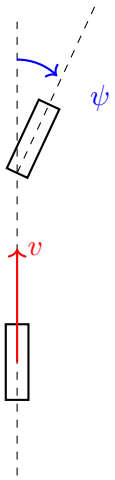


Figure 11 – Diagram illustrating the kinematics used in our model.

The commands for speed and steering angle are now clearly defined with respect to the car (Figure 11). Thus, it is now possible to fully determine the kinematic model of the system.

Subsequently, the car will be illustrated with four wheels for better understanding, but the modeling will remain as shown in Figure 11.

3.3 System Modeling

The control of the car is solely based on images from the camera, whose position (x, y) corresponds to the middle of the front axle. The angle ψ represents the steering of the front wheels, while θ measures the angle between the car's chassis and the target point (x_d, y_d). The distance r is defined as the separation between the target point and the car's current position.

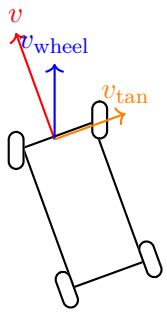


Figure 12 – The car with its corresponding speed vectors.

Let L be the wheelbase. The lever arm equation is therefore given by

$$L \cdot \dot{\theta} = v_{\tan},$$

knowing that, according to Figure 12,

$$\begin{cases} v_{\text{wheel}} \cos(\psi) = v \\ v_{\text{wheel}} \sin(\psi) = v_{\tan} \end{cases}$$

Thus,

$$v \cdot \tan(\psi) = v_{\tan}.$$

Therefore,

$$\dot{\theta} = \frac{v \cdot \tan(\psi)}{L}.$$

Given that ψ belongs to the interval $[-1, 1]$ (the control limits), we perform a first-order Taylor expansion around $\psi \approx 0$

$$\dot{\theta} \approx \frac{v \cdot \psi}{L}.$$

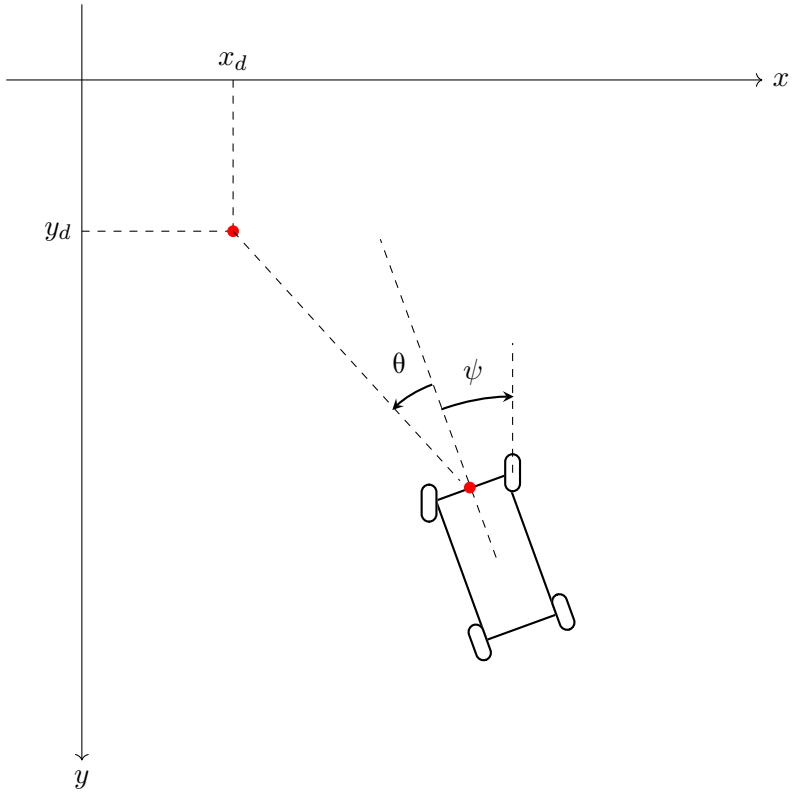


Figure 13 – Schematic representation of the problem in a Cartesian coordinate system: definition of the target point (xd, yd) and the angles θ and ψ.

Since the target point is to the left of the car, the angle θ is defined as negative (13). Thus, the coordinates (x, y) (represented by the red point on the car) are given by

$$\begin{cases} x = -r\sin(\theta) + x_d \\ y = r\cos(\theta) + y_d \end{cases}$$

with:

$$r = \sqrt{(x - x_d)^2 + (y - y_d)^2}$$

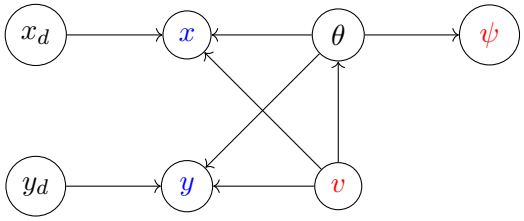


Figure 14 – Dependency diagram of state variables (in blue) and commands (in red).

The diagram of dependencies (Figure 14) demonstrates that x and y are directely linked to v and indirectely linked to ψ.

The delay matrix is therefore:

$$R = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$$

The minimum coefficients of each row in R appear in both columns. Therefore, the system is controllable.

4 Controller

4.1 PID Controller

Let's start with a PID controller, a logical first step in managing the vehicle's dynamics. Its simplicity and proven effectiveness make it an ideal choice for initial testing and fine-tuning before moving on to more advanced control strategies. The PID controller, by adjusting the proportional, integral, and derivative terms, offers a well-balanced approach to handling errors in real-time. This enables precise control over the car's speed and steering.

The structure of the delay matrix R indicates that some state variables have second-order delays, requiring double differentiation to accurately express $\begin{pmatrix} x \\ y \end{pmatrix}$ in terms of the inputs v and ψ.

The derivatives of $\{x, y\}$ are

$$\begin{cases} \dot{x} = -\dot{r}\sin(\theta) - r\dot{\theta}\cos(\theta) + \dot{x}_d \\ \dot{y} = \dot{r}\cos(\theta) - r\dot{\theta}\sin(\theta) + \dot{y}_d \end{cases}$$

Using the first-order approximation for $\dot{\theta}$,

$$\begin{cases} \dot{x} = -v\sin(\theta) - \frac{rv\psi}{L}\cos(\theta) + \dot{x}_d \\ \dot{y} = v\cos(\theta) - \frac{rv\psi}{L}\sin(\theta) + \dot{y}_d \end{cases}$$

Therefore,

$$\begin{cases} \ddot{x} = -\dot{v}\sin(\theta) - v\dot{\theta}\cos(\theta) - \frac{1}{L}[\dot{r}v\psi\cos(\theta) + r\dot{v}\psi\cos(\theta) + rv\dot{\psi}\cos(\theta) - rv\psi\dot{\theta}\sin(\theta)] + \ddot{x}_d \\ \ddot{y} = \dot{v}\cos(\theta) - v\dot{\theta}\sin(\theta) - \frac{1}{L}[\dot{r}v\psi\sin(\theta) + r\dot{v}\psi\sin(\theta) + rv\dot{\psi}\sin(\theta) + rv\psi\dot{\theta}\cos(\theta)] + \ddot{y}_d \end{cases}$$

Considering the second-order approximation where $\dot{v} \cdot \psi = 0$ (implying that the speed varies little and ψ is small), the second derivatives $\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix}$ can be expressed in terms of matrices and vectors as follows,

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \underbrace{\begin{pmatrix} -\sin(\theta) & -\frac{rv}{L}\cos(\theta) \\ \cos(\theta) & -\frac{rv}{L}\sin(\theta) \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} \dot{v} \\ \dot{\psi} \end{pmatrix}}_{U} + \underbrace{\begin{pmatrix} \frac{-2v^2\psi\cos(\theta)}{L} + \frac{rv^2\psi^2\sin^2(\theta)}{2} + \ddot{x}_d \\ \frac{-2v^2\psi\sin(\theta)}{L} - \frac{rv^2\psi^2\cos^2(\theta)}{2} + \ddot{y}_d \end{pmatrix}}_{B}$$

Thus, let's consider the following PID-based control with feedback linearization,

$$\begin{aligned} U &= A^{-1}(V - B) \\ V &= \ddot{X} = k_i(W - X) + k_p(\dot{W} - \dot{X}) + k_d\ddot{W} \end{aligned}$$

where $W = \begin{pmatrix} x_d \\ y_d \end{pmatrix}$ and k_i, k_p, and k_d are the integral, proportional, and derivative gain matrices

This controller is implemented in Python and tested within the simulation environment, with the implementation details provided in Algorithm 1.

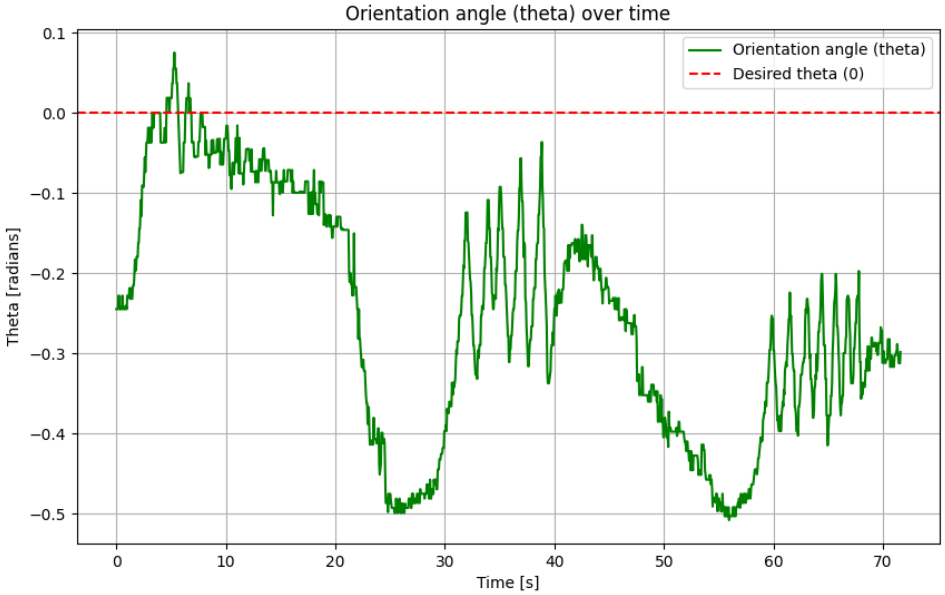


Figure 15 – Evolution of θ over time.

The desired position for the car is to always have its target point directly in front of the camera, which means that θ should ideally be close to 0. However, if θ were constantly at 0, the car would follow a straight path, which isn't feasible given the circuit's curves. As a result, θ will inevitably deviate from 0 when navigating turns, but the goal is to keep these deviations as small as possible.

It's important to note that the controller is not 100% robust. There are instances where the car deviates so far from the intended trajectory that it struggles to return to it. To mitigate this, the steering is limited between -0.8 and 0.8, and the speed is capped at 0.02 (Figure 15). These constraints are set to reduce oscillations and give the controller enough time to allow the system to converge back to the desired path. Even with these adjustments, while the car usually manages to complete at least one full lap, it can still deviate from its trajectory afterward.

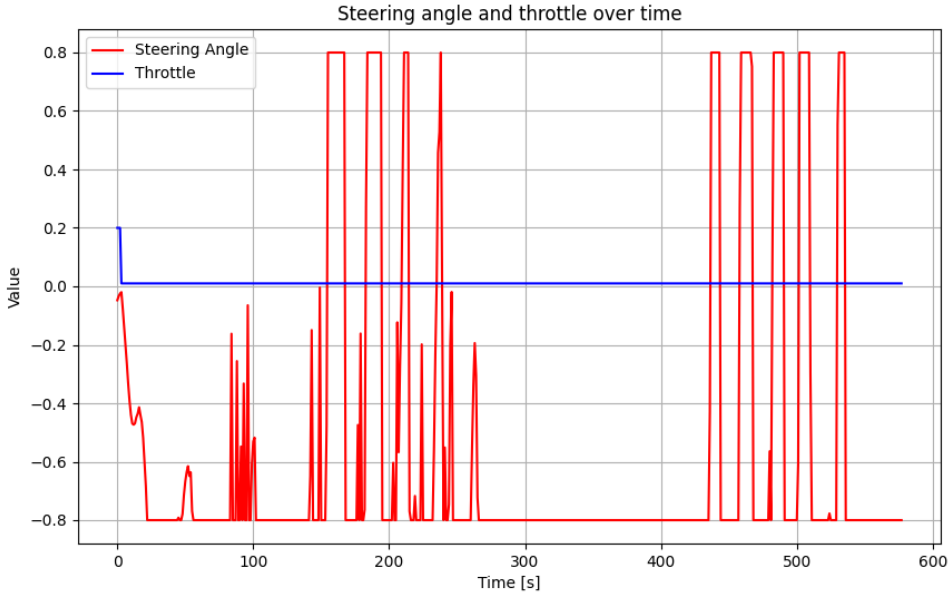


Figure 16 – Evolution of steering and throttle over time.

The curved and straight segments in the graphical evolution of the steering (Figure 16) can be readily distinguished: rapid and large oscillations correspond to the straight segments, while the segments that exhibit nearly constant behavior indicate the curves. This pattern can also be observed, though less distinctly, in the graph presented in Figure 15. The system exhibits significant oscillatory behavior on straight sections of the path, leading to instability as speed increases. Consequently, when the speed becomes too high, the system diverges entirely and loses the track, especially given the tight and short nature of the circuit.

The error graph (Figure 17) suggests that the controller might be overreacting to small deviations, possibly due to overly aggressive gain settings, particularly for the proportional term. This overreaction leads to oscillations in the system rather than allowing it to smoothly converge to the desired state. However, when the proportional gain is reduced, the system fails to complete even a single lap before diverging. Additionally, while the system appears to be diverging based on the error graph, this is not always the case in the simulation. It's important to consider the scale of the errors, which are measured in pixels (elements of the image's numpy array). An error range of 20 pixels, therefore, does not necessarily indicate that the system is completely diverging.

While PID controllers are well-suited for point tracking, they often struggle with dynamic trajectories, particularly when the trajectory is unknown and evolves in real-time, frame by frame and point by point. As the target changes, the system oscillates in its attempt to stabilize, only to be disrupted by further trajectory adjustments. This continuous need for adaptation can result in persistent oscillations and a tendency to diverge. To address these challenges, we turn to MPC, which is specifically designed to handle such dynamic environments more effectively by anticipating future trajectory changes and optimizing control actions accordingly.

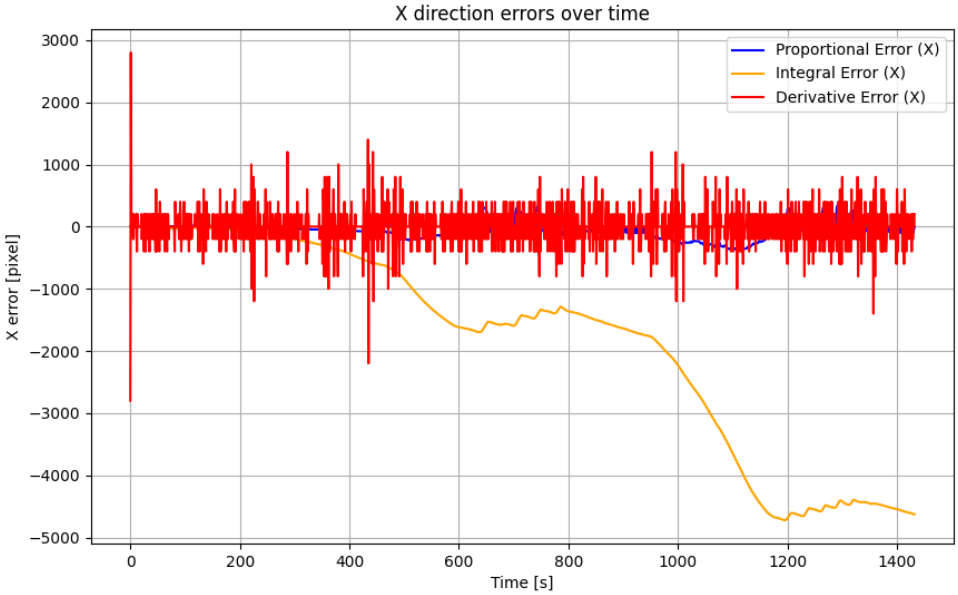


Figure 17 – Horizontal error over time.

4.2 MPC

MPC is a sophisticated control strategy that optimizes a system's trajectory over a specified time horizon by predicting its future states. This prediction is crucial for achieving optimal control performance, especially in scenarios that demand precise maneuvering. Therefore, accurately knowing the near-future trajectory is essential.

In our case, the only sensor available is a camera. Therefore, the trajectory prediction capability is constrained by the camera's limitations. Given the vertical FOV of 48.8° (2), the depth of field, or the distance the camera can see ahead on the ground, is estimated to be 11 cm (as calculated in appendix Appendix B).

This means that the system can at best predict the trajectory over the next 11 cm. However, this theoretical depth of field is actually reduced. To accurately predict the future trajectory, the system needs a clear detection of the track boundaries, which becomes unreliable beyond 10 cm.

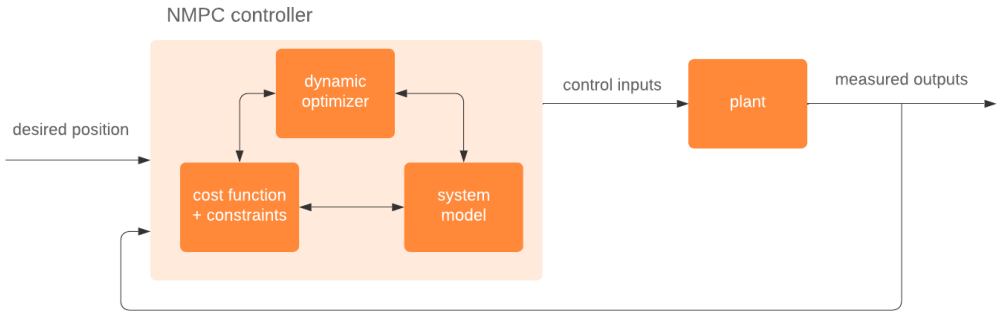


Figure 18 – Functional diagram of an NMPC controller

To address this limitation, the trajectory is projected into an external reference frame that allows the system to generate and follow a predicted path over a greater distance. This longer future trajectory is essential for MPC, as it enables the system to anticipate and smoothly navigate upcoming path segments, whether they are straight or curved. By optimizing control actions over a longer horizon, MPC can better manage speed, steering, and other variables, leading to more stable and efficient vehicle behavior.

The external reference frame used in this approach is a global coordinate system that is independent of the camera's viewpoint. In this frame, the vehicle's position is represented by its coordinates (x, y) and orientation θ relative to a fixed origin, typically set at the start of the path.

Within this reference frame, a clear distinction is made between straight and curved path segments. For straight segments, the trajectory is represented as a simple linear path, while for curves, the trajectory follows a circular or otherwise appropriate curve based on the road geometry.

Let's now delve into the workings of the MPC controller. At each time step, the controller solves an optimization problem that minimizes a cost function while ensuring that the system adheres to its dynamic constraints and input limitations.

The cost function plays a crucial role in determining the controller's behavior. In our case, we prioritize maintaining the vehicle's position within the track boundaries and ensuring that the steering angle remains within acceptable limits. Therefore, the cost associated with deviations in position and steering is set high. On the other hand, the cost related to speed is deliberately kept lower because controlling speed is not as critical for our primary objective. This approach allows the MPC to concentrate more on accurately following the trajectory and making precise steering adjustments, without being overly concerned with achieving a specific speed, which is less crucial in our context.

The optimization problem is formulated as follows ([5])

$$\min_{\mathbf{U}} J = \sum_{k=0}^{N-1} \left(\mathbf{x}_k^\top \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^\top \mathbf{R} \mathbf{u}_k \right)$$

subject to

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$$

and

$$\mathbf{x}_0 = \mathbf{x}(t_0)$$

— $\mathbf{x}_k = \begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix}$ is the state vector at time step k,
— $\mathbf{u}_k = \begin{pmatrix} v_k \\ \psi_k \end{pmatrix}$ is the control input vector at time step k.
— $\mathbf{Q}$ and $\mathbf{R}$ are weighting matrices associated with the states and controls, respectively, and are critical in defining the cost function's sensitivity to deviations in states and control efforts.
— N is the prediction horizon, which dictates how many future time steps the MPC will consider in its optimization process.

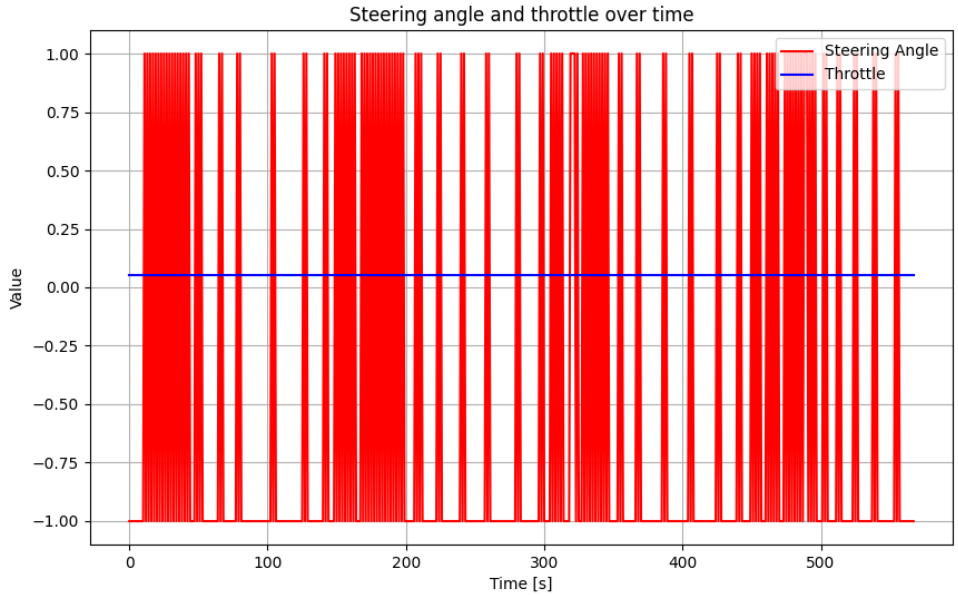


Figure 19 – Evolution of steering and throttle over time.

The reference to the implementation of the controller is available in Algorithm 2.

Figure 19 the evolution of the steering angle (in red) and throttle (in blue) over time for a vehicle controlled by an MPC controller. The throttle value remains relatively constant, which indicates stable speed control throughout the maneuver. In contrast, the steering angle exhibits rapid oscillations, frequently switching between positive and negative values and even reaching its limits of -1 and 1.

These oscillations indicate that the MPC controller is making frequent, fine-tuned adjustments to maintain the desired path. The high frequency of these adjustments suggests a highly responsive system that continuously corrects itself to keep the vehicle centered. This responsiveness is a key strength of MPC, allowing it to react quickly to deviations and ensure the vehicle remains on track.

Although the graph shows significant oscillations, these are centered around zero, which means that the vehicle consistently attempts to return to its intended path. Unlike the PID controller, which may exhibit larger and slower adjustments that could lead to deviations or even trajectory divergence, the MPC's quick corrections indicate a robust and reliable control strategy. Despite the increased oscillation rate, the system does not diverge, which demonstrates the MPC's

effectiveness in maintaining stability and ensuring precise control of the vehicle, making it better suited for dynamic and uncertain environments.

Conclusion

In the simulated environment, the single camera sensor demonstrates sufficient capability to meet system requirements. Despite its average resolution, the high contrast between the track and its boundaries within the simulation environment enables consistent and reliable detection. This strong detection performance, when paired with an appropriate controller like the MPC, delivers a highly robust outcome, although minor oscillations are present that could be dampened with further optimization.

However, when transitioning to the real-world scenario, the sensitivity of the system to environmental changes became apparent. Even slight variations, such as shadows, considerably impacted the system's ability to detect track boundaries effectively. Despite extensive parameter tuning and filtering attempts, variations in ambient light, which could differ significantly from day to day, compromised the robustness of the image processing pipeline. These challenges highlight the significant difficulty in utilizing a single vision-based sensor as the exclusive source of information in a real-world application where lighting and environmental factors vary unpredictably.

Addressing these challenges would require a more sophisticated approach. A potential improvement could involve integrating supervised learning techniques with the MPC controller. Machine learning could help the system generalize across varying light conditions by enabling it to produce appropriate control commands even when direct visual detection of boundaries becomes unreliable. For instance, a trained model could leverage previous learned scenarios to generate effective responses in ambiguous lighting conditions.

Moreover, integrating additional sensors such as GPS (Global Positioning System) or LIDAR (Light Detection and Ranging) could further improve the reliability of the system. GPS data could supplement the visual information, providing positional accuracy even when the visual sensor is compromised. This would lead to a more robust multi-sensor fusion system capable of coping with the dynamic real-world environment. While the camera remains a valuable sensor for tasks such as obstacle detection, relying solely on vision for all navigation and control underlines the inherent limitations, especially in a system where robustness is paramount. Thus, adding redundancy with multiple types of sensors would not only mitigate the limitations of a camera but also enhance overall system resilience and accuracy.

Acronyms

PID Proportional-Integral-Derivative
MPC Model Predictive Control
ESC Electronic Speed Controller
C2 Command and Control
FOV Field Of View
RGB Red, Green, Blue
HSV Hue, Saturation, Value
ICR Instantaneous Center of Rotation
NMPC Nonlinear Model Predictive Control

GPS Global Positioning System

LIDAR Light Detection and Ranging

A Command and Control Architecture

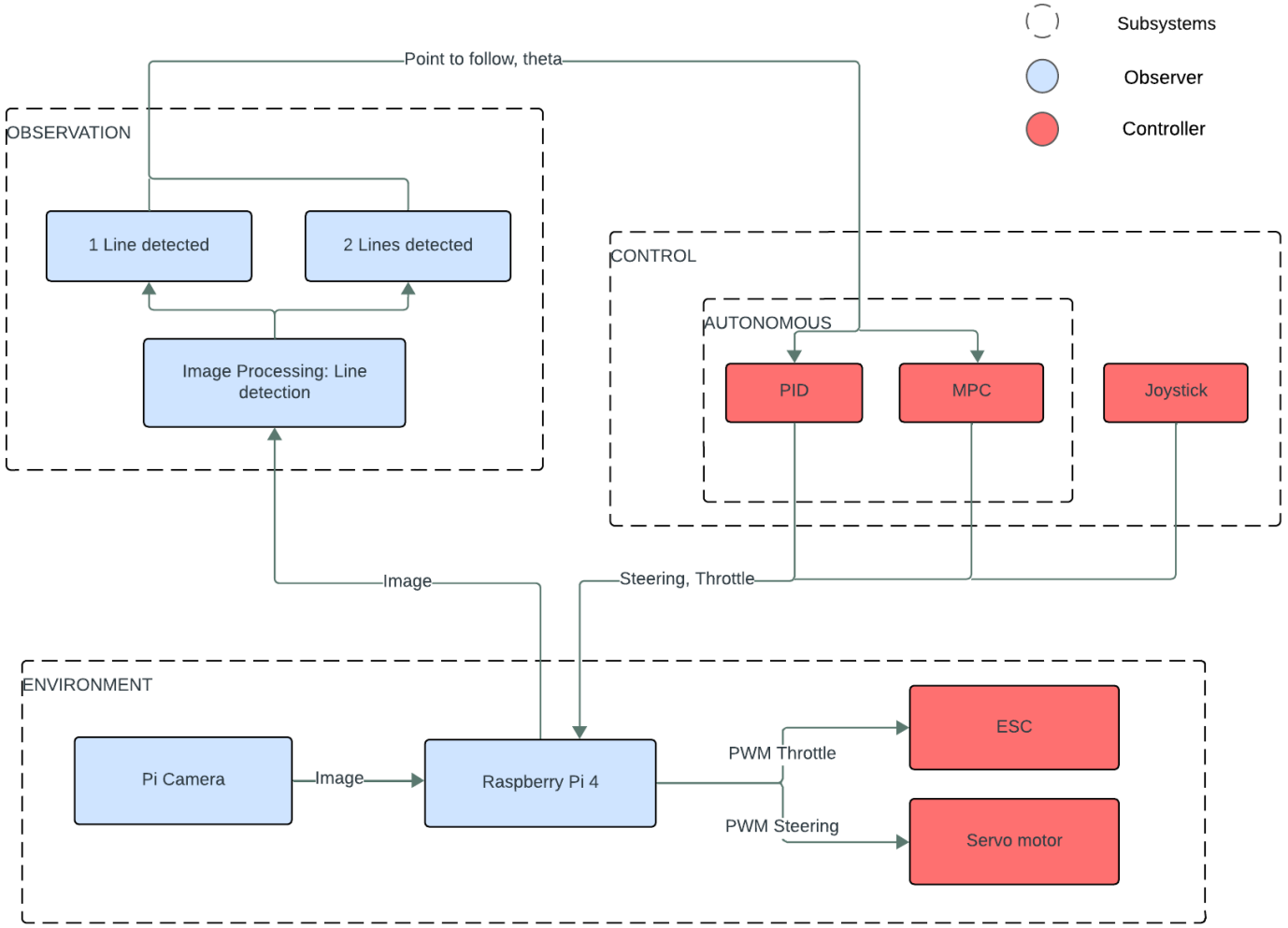


Figure 20 – C2 Architecture

B Depth of Field Calculation of the Camera

To determine the depth of field d, we use the camera's mounting height h and the downward angle of the field of view, which is half of the vertical FOV. The relevant angle θ_{bottom} is:

$$\theta_{\text{bottom}} = \frac{48.8^\circ}{2} = 24.4^\circ$$

The distance d, that the camera can see ahead on the ground, can be calculated using the following trigonometric relationship:

$$d = \frac{h}{\tan(\theta_{\text{bottom}})}$$

Where h represents the camera's height above the ground. Given that it is mounted at $h = 0.05$ meters (5 cm), $d \approx 0.11m$.

Algorithms

Algorithm 1: PID-based control with feedback linearization

Input: dt: Time step; L: Vehicle Length; θ, ψ, v: Initial states; r: Radius; X: Current state; W: Desired position $\begin{bmatrix} xd \\ yd \end{bmatrix}$; W_{prev}, W_{prev2}: Previous desired positions; k_i, k_p, k_d: PID gains

Result: Updated control inputs v, ψ

$dW \leftarrow \frac{W - W_{prev}}{2 \times dt}$;

$ddW \leftarrow \frac{W - 2 \times W_{prev} + W_{prev2}}{dt^2}$;

$W_{prev2} \leftarrow W_{prev}$;

$W_{prev} \leftarrow W$;

`f(`$X, \theta, \psi, L, v, r, dW$`)` $\mathbf{M} \leftarrow \begin{bmatrix} -\sin(\theta) & -\frac{r \cdot v \cdot \cos(\theta)}{L} \\ \cos(\theta) & -\frac{r \cdot v \cdot \sin(\theta)}{L} \end{bmatrix}$;

$dX \leftarrow \mathbf{M} \cdot \begin{bmatrix} v \\ \psi \end{bmatrix} + dW$;

return dX;

$dX \leftarrow f(X, \theta, \psi, L, v, r, dW)$;

$X \leftarrow X + dX \times dt$;

$A \leftarrow \begin{bmatrix} -\sin(\theta) & -r \times v \times \cos(\theta)/L \\ \cos(\theta) & r \times v \times \sin(\theta)/L \end{bmatrix}$;

$B \leftarrow \begin{bmatrix} -2 \times v^2 \times \psi \times \cos(\theta)/L + r \times v^2 \times \psi^2 \times \sin(\theta)/L^2 + ddW[1] \\ -2 \times v^2 \times \psi \times \sin(\theta)/L - r \times v^2 \times \psi^2 \times \cos(\theta)/L^2 + ddW[2] \end{bmatrix}$;

$V \leftarrow k_i \times (W - X) + k_p \times (dW - dX) + k_d \times ddW$;

$errors \leftarrow k_i \times (W - X) + k_p \times (dW - dX) + k_d \times (ddW - V)$;

$U \leftarrow A^{-1} \times (V - B)$;

$u1 \leftarrow U[1], u2 \leftarrow U[2]$;

$v \leftarrow v + u1 \times dt$;

$\psi \leftarrow \psi + u2 \times dt$;

Algorithm 2: CasADi-based MPC

Given: N: Prediction Horizon;
dt: Time step;
L: Vehicle Length;
Q, R: Cost Matrices;
max_steer, max_vel: Constraints on Steering and Velocity;
X_0: Initial State;
$Segment$: Type of Circuit Segment ('straight', 'curved_left', 'curved_right');
$radius$: Radius for Curved Trajectories (optional);

while *task not completed* **do**
  $ref \leftarrow$ GenerateReferenceTrajectory(X_0, $Segment$, $radius$);
  $opti \leftarrow$ InitializeOptimizationProblem();
  $X, U \leftarrow$ opti.variable(3, N+1), opti.variable(2, N);
  $cost \leftarrow 0$;
  for $k \leftarrow 0$ **to** $N-1$ **do**
    $state\_error \leftarrow X[:,k] - ref[:,k]$;
    $control\_input \leftarrow U[:,k]$;
    $cost += state\_error^T \times Q \times state\_error + control\_input^T \times R \times control\_input$;
    $dynamics\_output \leftarrow$ Dynamics($X[:,k], U[:,k]$);
    $opti.subject\_to(X[:,k+1] == X[:,k] + dt \times dynamics\_output)$;
  $opti.subject\_to(X[:,0] == X_0)$;
  $opti.subject\_to(-max\_steer \leq U[1,:] \leq max\_steer)$;
  $opti.subject\_to(0 \leq U[0,:] \leq max\_vel)$;
  $opti.minimize(cost)$;
  $sol \leftarrow$ opti.solve();
  if $sol.failed$ **then**
    DebugAndLogValues($opti$);
    RaiseError;
  $control\_input \leftarrow$ sol.value(U[:,0]);
  $X_0 \leftarrow$ sol.value(X[:,1]);
  SendToActuators($control\_input$);
  UpdateInitialState(X_0);

Bibliography

References

[1] World Health Organization. "Global status report on road safety 2023". In: *World Health Organization* (2023). 1

[2] Andrea Stocco, Brian Pulfer, and Paolo Tonella. "Mind the Gap! A Study on the Transferability of Virtual vs Physical-world Testing of Autonomous Driving Systems". In: *IEEE Transactions on Software Engineering* XY.X (2022), pp. 1–XY. 1

[3] Raspberry Pi Foundation. "Camera". In: *Raspberry Pi Documentation*, n.d. [Online]. Available: https://www.raspberrypi.com/documentation/accessories/camera.html. [Accessed: Month, Day, Year]. 5

[4] S. Liu. "Analysis of the Car Model and Ackermann?s Steering Geometry". In: *Highlights in Science, Engineering and Technology* 37 (2023), pp. 1–13. 13

[5] B. Takács, J. ?tevek, R. Valo, and M. Kvasnica. "Python code generation for explicit MPC in MPT". In: *2016 European Control Conference (ECC)*, Aalborg, Denmark, 2016, pp. 1328–1333, doi: 10.1109/ECC.2016.7810473. 20

ENSTA Bretagne

RAPPORT D'EVALUATION
ASSESSMENT REPORT

Merci de retourner ce rapport par courrier ou par voie électronique en fin du stage à :
At the end of the internship, please return this report via mail or email to:

ENSTA Bretagne – Bureau des stages - 2 rue François Verny - 29806 BREST cedex 9 – FRANCE
☎ 00.33 (0) 2.98.34.87.70 / stages@ensta-bretagne.fr

I - ORGANISME / *HOST ORGANISATION*

NOM / *Name* Automation & Control Institute (ACIN), TU WIEN

Adresse / *Address* Gusshausstraße 27-29, 1040 Wien, Austria

Tél / *Phone (including country and area code)* +43 1 58810

Nom du superviseur / *Name of internship supervisor*
Dr. Vu Nhat Minh

Fonction / *Function* Postdoctoral Researcher

Adresse e-mail / *E-mail address*

Nom du stagiaire accueilli / *Name of intern* RICOUARD Ambre

II - EVALUATION / *ASSESSMENT*

Veuillez attribuer une note, en encerclant la lettre appropriée, pour chacune des caractéristiques suivantes. Cette note devra se situer entre **A (très bien)** et **F (très faible)**
*Please attribute a mark from **A (excellent)** to **F (very weak).***

MISSION / *TASK*

- La mission de départ a-t-elle été remplie ? A (B) C D E F
  Was the initial contract carried out to your satisfaction?
- Manquait-il au stagiaire des connaissances ? ☐ oui/*yes* ☑ non/*no*
  Was the intern lacking skills?

  Si oui, lesquelles ? / *If so, which skills?*

ESPRIT D'EQUIPE / *TEAM SPIRIT*

- Le stagiaire s'est-il bien intégré dans l'organisme d'accueil (disponible, sérieux, s'est adapté au travail en groupe) / *Did the intern easily integrate the host organisation? (flexible, conscientious, adapted to team work)*

  (A) B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here*

COMPORTEMENT AU TRAVAIL / *BEHAVIOUR TOWARDS WORK*

Le comportement du stagiaire était-il conforme à vos attentes (Ponctuel, ordonné, respectueux, soucieux de participer et d'acquérir de nouvelles connaissances) ?

Did the intern live up to expectations? (Punctual, methodical, responsive to management instructions, attentive to quality, concerned with acquiring new skills)?

(A) B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

INITIATIVE – AUTONOMIE / *INITIATIVE – AUTONOMY*

Le stagiaire s'est –il rapidement adapté à de nouvelles situations ? (A) B C D E F
(Proposition de solutions aux problèmes rencontrés, autonomie dans le travail, etc.)

Did the intern adapt well to new situations? A (B) C D E F
(eg. suggested solutions to problems encountered, demonstrated autonomy in his/her job, etc.)

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

CULTUREL – COMMUNICATION / *CULTURAL – COMMUNICATION*

Le stagiaire était-il ouvert, d'une manière générale, à la communication ? (A) B C D E F
Was the intern open to listening and expressing himself /herself?

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

_____

OPINION GLOBALE / *OVERALL ASSESSMENT*

❖ La valeur technique du stagiaire était : A (B) C D E F
Please evaluate the technical skills of the intern:

III - PARTENARIAT FUTUR / *FUTURE PARTNERSHIP*

❖ Etes-vous prêt à accueillir un autre stagiaire l'an prochain ?

Would you be willing to host another intern next year? ☑ oui/*yes* ☐ non/*no*

Fait à _____, le _____
In _____, on _____

Univ. Prof.
Dr. Andreas Kugi

Signature Entreprise _____ Signature stagiaire
Company stamp _____ *Intern's signature*

Institut für Automatisierungs- und Regelungstechnik
Technische Universität Wien
A-1040 Wien 4, Gußhausstraße 27
Tel. 58 8 01

Merci pour votre coopération
We thank you very much for your cooperation