UNIVERSITY OF TRENTO

Department of Industrial Engineering

Mechatronic engineering, electronics and robotics course

Development of an on-board system for measuring the velocity vector of a driverless Formula Student racing car

Thesis project year:	2021 – 2022	
Thesis committee:	Prof. Bosetti Paolo,	University of Trento, supervisor
	Res. Strobbe Cristiano,	University of Trento, co-supervisor
Student name:	Spadetto Matteo [214352]	

Preface

I would like to express my sincere gratitude to my family, Luca, mom and dad, for the support they gave me during these years. I would also like to thank the supervisor, Prof. Paolo Bosetti, who assisted and guided me through this project. More importantly, as faculty advisor of E-agle Trento Racing Team, he gives me the possibility to directly apply the knowledge gained during the lessons in real mechatronic systems. I would like to thank the co-supervisor, researcher and very good friend Cristiano Strobbe for the time spent in the laboratory during this thesis and the Formula Student experience. Last but not least, I wish to thank all my friends who accompanied me during this university adventure.

Trento, October 2022

Summary

The goal of this thesis is to design a new sensor for measuring the velocity vector direction in racing vehicles. In order to overcome the complexity of nowadays systems used in vehicle dynamics, the system is implemented starting from the physical concept of velocity. Sensors are acquiring world information so it is fundamental to perform the study as close as possible to the real quantity to analyze. The studied approaches are based on camera samples representing the ground asphalt mixture texture, giving the possibility of using contactless and absolute methods. This thesis is focused on the analysis of the direction of the velocity vector, for which the prototype is developed, while the magnitude information is treated in more theoretical terms. Different concepts and methods are taken into account and kept or discarded according to the concurrent engineering approach to achieve the best final solution. The project covers the system requirements, the concepts generation, the implementation and the overall system performances achieved during the study of the GVS (Ground Velocity Sensor). The camera model is defined in the real and simulated environment to acquire the ground frames. The new sensor is based on the motion blur effect which is present in the frames when the camera exposure time is comparable with the velocity vector magnitude. Due to The methods used as main algorithms are based on FD (Frame Difference) and ORB(Oriented fast and Rotated Brief) detector at low speed while HoG (Histogram of Gradients), FFT (Fast Fourier Transform) are used at higher velocity vector magnitude. The speed threshold level used for algorithm switching is discussed. LF (Laplacian Filter) is a treated as possible processing computation to adopt in speed analysis. Finally, the overall system performances and functionality are described. To contextualize the system, an application is described as the first test prototype: the Formula Student driverless racing car of the University of Trento.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
GVS	Ground Velocity Sensor
FS	Formula Student
QFD	Quality Function Deployment
CV	Combustion Vehicles
EV	Electric Vehicles
DV	Driverless electric vehicles
ECU	Electric Control Unit
AI	Artificial Intelligence
LIDaR	Laser Imaging Detection and Ranging
SLAM	Simultaneous Localization and Mapping
ML	Machine Learning
CoM	Center of Mass
IMU	Inertial Motion Unit
GPS	Global Positioning Sensor
FD	Frames Difference
ORB	Oriented fast and Rotated Brief
SIFT	Scale Invariant Feature Transform
SURF	Speeded Up Robust Features
HoG	Histogram of Gradients
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
LF	Laplacian Filter
CCD	Charge-Coupled Device
PpI	Pixels per Inch
FpS	Frames per Second
MSE	Mean Squared Error
IS	International System
ID	Identifier
RGB	Red Green Blue
HSV	Hue Saturation Value

1

Formula student competitions

The first application of the GVS prototype is designed to be applied to FS (Formula Student) racing vehicles. The University of Trento competes in these championships as E-agle Trento Racing Team, from 2016 with two cars: Chimera Evoluzione and Fenice Fig.1.1. The first one is also used to test new and innovative systems that can make the difference in score competitions. As in the case of this thesis, the Formula Student teams are a suitable environment to develop new technologies.

As mentioned later, innovation and creativity are scored in the FS events and they are fundamental in order to achieve high results. In this context, not only Universities but also companies are interested in the systems developed in thousands of FS teams spread around all the World.

A brief introduction to this championship is now treated in order to understand the requirements set to design the GVS.

Figure 1.1: Chimera Evoluzione on the left and Fenice on the rigth designed by E-agle Trento Racing Team.

Formula Student is a worldwide competition for racing cars where only students can compete. The designed vehicles are prototypes built following the concurrent engineering approach, focusing the design on the most useful requirements, in order to increase the overall final quality of cars every year. The common technical flow passes through the following steps: set of requirements, QFD (Quality Function Deployment), concept generation and evaluation, design, development and tests, both static and dynamic. The goal of each team is very different: from the moment not only races are evaluated, teams can also compete just presenting new systems and concepts if they are motivated following the mentioned steps. Moreover, good documentation is necessary: first of all internally, due to the high rate of members changing, and externally as proof of concepts and designs at the events. The GVS project is part of this documentation

available to the team in order to demonstrate the rate of innovation of E-agle Trento Racing Team.

The participation in races is granted by online quiz tests used to evaluate the overall knowledge of each team. Quizzes are not only covering the event rules but also the detailed knowledge of mechanics, electronics, and economics. E-agle Trento Racing Team, passed the test and competed in Italy, Spain, Hungary, Croatia and Germany achieving good results and prizes.

Each competition event is divided into three main vehicles categories: CV (Combustion Vehicles), EV (Electric Vehicles) and DV (Driverless electric vehicles). In particular, E-agle Trento Racing Team build EV and DV prototypes: the first vehicle, Chimera, upgraded in a second version, Chimera Evoluzione, which is now used as base for the new driverless car and a completely new electric vehicle, called Fenice, which competed in summer 2022.

The championship scores are not only given for track races, the dynamic events, but also on static ones such as: business plan, cost and manufacturing and engineering design presentations. These last events aim to guarantee the quality of the design evaluating the reasons for the implemented mechanics and electrical systems. Moreover, the money spent is taken into account together with the presentation of a plan of how it would be possible to place the car on the market. The concept behind FS championship is not just to race but also to give a full working experience to the students. For this reason teams need to think as much as possible as a real company and produce the necessary documentation and certifications as required by the rules book and judges.

More in the details, the dynamic events are:

- The acceleration event, where the car has to run 75 m as quickly as possible. The goal is to evaluate the maximum acceleration of the cars;
- The skidpad event, consisting of two pairs of concentric circles in a figure of eight pattern to run as quickly as possible. The goal is to evaluate the grip of the cars;
- The autocross event layout is a handling track with a length under 1.5 Km. Also in this case it has do be completed as quickly as possible;
- The trackdrive event is performed on the same track of the autocross but it is reserved to driverless vehicles only. DV cars have one lap to run the circuit before the race;
- The endurance and efficiency event consists of a run on a closed lap circuit. The lap length is approximately 1 Km and the complete endurance is 22 Km long. Driverless vehicles are not admitted. Not only the time but also the used energy, or fuel, of the car are evaluated at the end of the race. The goal is to evaluate both efficiency and reliability.

The events are designed to push the car performances to the limit in all aspects: acceleration, grip, maximum speed, efficiency and reliability.

As mentioned before, static events are scored. For this reason it is fundamental to bring new technologies to the competition every year and, from the moment no cars are equipped by an absolute velocity sensor matching the vehicle overall requirements, the GVS system can be considered one of them.

An important aspect of the FS championship is related to safety. Before every dynamic event all cars must pass several inspections such as: mechanical, electrical, rain and tilt test. Moreover, safety is not only related to the race: the rules book also covers the aspects outside the track keeping the possibility of disqualifying the team for every possibly dangerous behavior.

DV is a special category of Formula Student that implements a variety of very innovative technologies based on AI (Artificial Intelligence) and ML (Machine Learning) techniques in order

to implement autonomous driving features. These cars are usually built on pre-existing vehicles with a high level of reliability. New sensors, architecture, code and hardware are needed in order to match the requirements of the DV category Fig.1.2.

Figure 1.2: Example of a driver-less Formula Student car designed by AMZ team.

Going more in the details about driverless vehicles, they have to be driven both by the AI and a human driver. Their tracks are delimited by yellow, blue and orange cones to help the algorithms during the path planning and object segmentation steps. Nowadays, the majority of these vehicles are equipped with several sophisticated sensors as LIDaR (Laser Imaging Detection and Ranging) and cameras used to map the environment thanks to SLAM (Simultaneous Localization and Mapping), computer vision and ML techniques [9].

The teams approaches are very different and it is the most advanced category, in terms of knowledge and difficulty, of the competition. Upgrading a car with such features means not only working on complex algorithms but sometimes also on the original sensors and architectures. The general concept, followed by every team, is to modify an EV car architecture by adding new nodes to the network and changing the less possible elements of the original vehicle. This is done in order to decrease the implementation time maintaining a reliable base car. Sometimes, unfortunately, completely new systems are necessary, as in the case of the GVS (Ground Velocity Sensor) treated in this thesis.

As mentioned before, the rules set a more controlled environment than the case of open world vehicles, limiting the operative ranges at certain values and defining specific events and tests. This gives the possibility to a lot of students, companies and researchers of using this championship to develop new high level technologies at prototype level. Thanks to the information given, the new technologies don't need to work in all situations but just in constrained cases. After this step it is possible to upgrade the systems also for other environments such as other racing competitions or also consumer markets.

In particular, in the GVS case, the most useful constraints to take into account, based on the rules book of the FS championship, are the following:

- By defying as 0 degrees angle the car moving straight, the maximum steer angle $\theta(t)$ is around ± 15 degrees;
- Maximum speed in the acceleration event is around 160 Km/h;
- Wheels slip is possible and so the velocity vector can be wrong if just the wheels speed or the steering angle are taken into account;
- In FS vehicles, GPS (Global Positioning System) and IMU (Inertial Motion Unit) are used together with encoders to retrieve the velocity vector information;
- FS events are taken on tracks, so asphalt ground is always present under the car;

- Races start at variable hours of the day and in an open environment, so different light conditions must be taken into account for camera frame acquisitions.

These information are briefly exposed here just to understand the general problems and ranges the development of the GVS can face and they are taken into detailed considerations in the next chapter for the requirements definition.

The today state of the art velocity sensors in FS championship are divided in two main groups:

- The systems based on most common sensors as IMU (Inertial Motion Unit), GPS (Global Positioning System), Pitot tubes and encoders;
- The systems based on AI and cameras.

Starting from the first category and analyzing the listed sensors, the main problems related to their technologies are [12]:

- IMU measures the accelerations and neither the velocity or the direction of the car directly. Due to the integration needed to calculate the vector of interest, a loss of accuracy and precision is intrinsically added;
- GPS is connected to external satellites requiring good connection. This kind of sensor retrieve a good measure of the direction of the vehicle but there is not a direct acquisition of the car speed;
- Pitot tubes are very good speed detector but, due to their structure and technology, they are not used for measuring the direction of the velocity vector;
- Encoders on wheels are a suitable solution for speed detection but they cannot retrieve information about the direction of the vehicle. To overcome this problem, another encoder can be mounted on the steering wheel to detect the steering angle through a cinematic chain. Passing time, mechanics can deteriorate changing configuration and resulting in a wrong measure of the real wheel angles. More importantly, slip can happen invalidating the measurement.

In the second group, instead, technologies are related to computer vision algorithms and cameras from the moment driverless cars are already equipped with these sensors [27]. Anyway, the systems designed for these acquisition systems are more focused on autonomous driving and so on obstacles recognition and image segmentation. Some velocity sensors based on the information these cameras acquire are available but, as explained later, they are usually based on sensors pointing in front of the car. Adapting these frames to the velocity vector analysis can be complex due to the high number of different features present in the frames. These kinds of approaches usually end in not very precise measurements due to the amount of noise present in the scene.

To overcome these issues and achieve better results, sensor fusion techniques are used increasing the overall complexity. These methods produce very good results but require a lot of experience and knowledge by the team.

It is important to remember that FS teams are subjected to a very high member change rate. Also due to this fact it is important to produce a detailed documentation to let the work continue in the following years by different people. Moreover, teams try to subdivide the biggest systems and designs in order to schedule tasks to end in just a year.

As it is possible to see, FS championship is a very innovative and creative environment where design and test new systems. For this reason and for the possibility of having a real DV prototype inside the University of Trento, the GVS system is intended as a new sensor to be applied, as first step, to the E-agle Trento Racing Team FS car.

2

The problem and the goal

Starting from the analysis of the requirements imposed by a velocity sensor system, the thesis treats in detail all the design processes used to retrieve the velocity vector of a moving vehicle, a FS racing one in particular.

The velocity vector is completely defined by its magnitude and direction, setting as goal the measurement of these two quantities. In the plot Fig.2.1 $|V(t)|$ and $\theta(t)$.

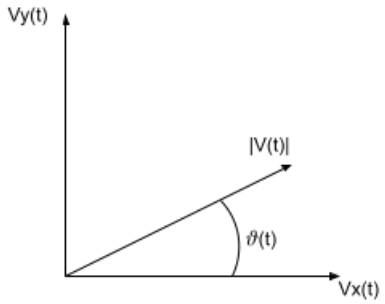

Figure 2.1: Representation of the velocity vector.

The reference frame set for all the following analysis has its origin in the CoM (Center of Mass). The x axes points in front of the car, the y axes to the left and the z axes to the top. Following this convention, the velocity axes are graphically represented as follows Fig.2.2.

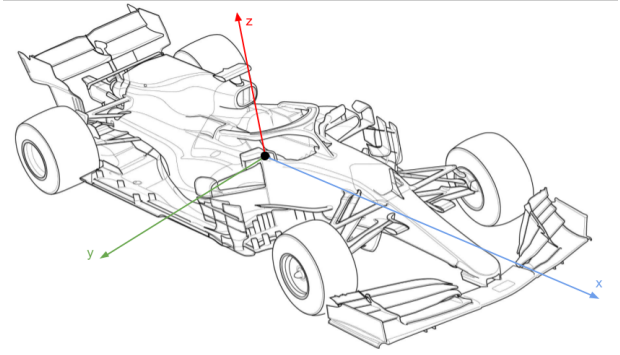

Figure 2.2: Frame reference system used in this thesis.

For the purpose of this thesis, just the x-y plane information is taken into account. From the moment the first implementation is based on camera frames acquired in a simulated environment, roll and pitch can be not considered. This approximation is introduced in the case the sensor is mounted on racing cars with very high chassis and suspension stiffness, for consumers vehicles, instead, with less rigidity, this assumption cannot hold.

As it is possible to understand from the short overview of the technology used, in the first chapter, the listed sensors are not suitable stand alone systems to retrieve the velocity vector quantities. For this reason, sensor fusion techniques are used to combine the information of some or all these kinds of sensors and calculate the magnitude and direction of the velocity vector. These approaches are elaborated due to the higher number of hardware elements. Moreover, the complexity is then reflected also in the developing time and level of knowledge necessary to design, test and validate them. Even worse is the case in which one of the sensors can be wrong or the sensors are not synchronized between each other. A single device acquiring the desired measures is usually preferable than a distributed one.

This short paragraph highlights how difficult it is nowadays to retrieve the velocity vector information, the direction in particular. Even if different sensors are available for the speed measurement, the direction analysis requires more than one data type. A stand alone sensor that can provide the desired information to the users would be a more easy system to handle. Moreover, to overcome mechanical issues and not need sensor fusion techniques, the system should perform a direct measurement of the quantity. Finally, the system should not depend on external elements as satellites. In some circumstances GPS can be not available or the connection can be not reliable enough for the requirements set by an architecture such as the one of a racing car. Loss of data, in vehicles, can cause control unit faults.

Before starting the design process, the requirements of the sensor are set. As it is possible to see from the following QFD Fig.2.3, the most important characteristics to be met are accuracy and precision. These two requirements are the base of the design for every sensor. Precision, related to the spread of the data acquired and accuracy, related to how much the measurement is reflecting the reality, are necessary to define an acquisition system.

			QFD velocity methods	Specifications											Competitors		
				Mean error speed	Mean error direction	Speed standard deviation	Direction standard deviation	Maximum speed	Maximum direction	Acquistion time	Processing time	Mesaure both speed and direction	Number of sensors	Time between wrong measure	IMU	GPS	Encoders
Requirements weights			Improvement direction	V	V	V	V	Λ	Λ	V	V	--	V	V			
Ground Velocity Sensor			Units	[m/s]	[deg]	[m/s]	[deg]	[m/s]	[deg]	[ms]	[ms]	[bool]	[#]	[ms]	Ratings		
23	Requirements	Performance	High accuracy	●	●										4	4	3
20			High precision			●	●	○	○						4	3	3
17			Fast computation							●	●				4	2	4
15		Usability	Compatibility									●			2	4	3
15			Low ext. complexity										●		3	4	3
10			Reliability											●	5	2	3
100				Specifics quantitative values											Influence legend		
															No correlation		
			Target (delighted)	0.5	0.2	1	0.5	600	90	3	13	YES	1	8000	Mid correlation		○
			Target (disgusted)	2	1	3	2	300	30	10	50	YES	3	3000	Hard correlation		●

Figure 2.3: QFD used to design the GVS system.

Moreover, the system's external complexity has to be reduced with respect to nowadays

approaches. External complexity is intended as the amount of work required by the final user to introduce the GVS system in an existing car. The best result is the case in which the new device is just giving as output the quantities about the velocity vector as a stand alone system. In this condition the user needs just to mount the GVS and calibrate it. Calibration, in this case, is intended as the set of parameters defining the camera model.

One of the most difficult parts to develop in a racing vehicle is the implementation of dynamic controls. In FS cars torque vectoring and slip controls are largely used requiring different sensors cooperating together. This is necessary to increase the final control quality but also because sometimes just one kind of data is not enough to extrapolate true and reliable information. Racing vehicles are subjected to high accelerations which sometimes are not possible to be handled by the driver. This condition happens when the response time of the human being is not fast enough compared to the response of the car, the engine in particular. In the first application of the GVS, the car where the sensor has to be mounted is an electric one, meaning that acceleration and responsiveness of the vehicle are very high. In these conditions, if the wheels start spinning the driver can lose seconds in the race or the car control.

The maximum operative ranges are taken into account starting from the data acquired during E-agle Trento Racing Team dynamic tests. Analyzing the logs on maximum car speed and change of direction it is possible to find a maximum speed of 134 Km/h and a maximum steer angle of ± 12 degrees. The target maximum speed is set at 140 Km/h and the angle direction is set for the at ± 20 degrees for the GVS application. Anyway, the final system can be used also at higher speeds and angles. To be more general, the maximum speed and angle ranges derived from the car performances in FS acceleration and skidpad events are taken into account [11]. The skidpad event, in the pattern of figure Fig.2.4, pushes the cars almost to the maximum steer angle. The centers of these circles are 18.25 m apart. The inner circles are 15.25 m in diameter and the outer circles are 21.25 m in diameter.

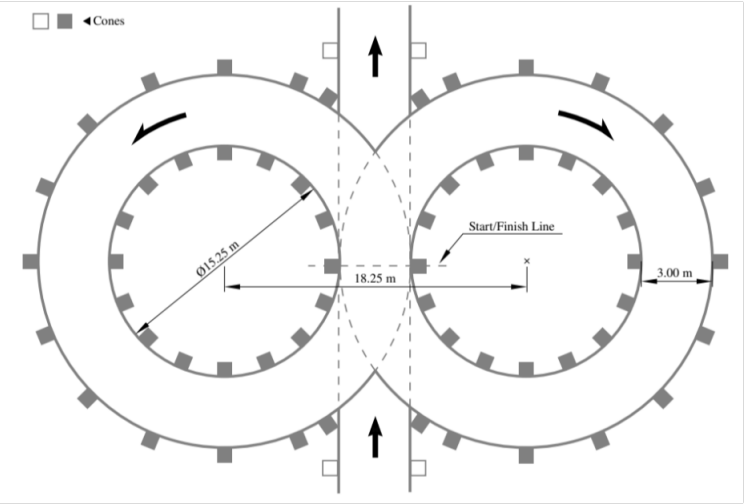

Figure 2.4: Skidpad pattern.

In order to meet the technical requirements defined above, the most reasonable choice is to design an absolute sensor whose reference is the ground. From the moment the velocity vector of the car CoM is measured with respect to the world, the concept is to implement a direct connection between the car and the asphalt on which the vehicle moves. Moreover, the sensor cannot be subjected to wheel slip and should be able to measure not just the steering angle

but the real behavior of the car, also in case the vehicle spins. These statements exclude the possibility of implementing approaches in which contact between car and ground are used.

The selected concept is to use a camera pointed on the ground collecting the asphalt mixture for processing. An image is directly acquiring the information of the real world without touching it. The asphalt flowing under the moving car is a texture changing at each instant. From the moment velocity is position and time dependent, the information is fully contained in the frame once the acquisition time is known. Having all the information in one device means it is possible to not use external sensors to mitigate errors or to check for invalid data due to wheel slip or ineffective steering wheel movements simplifying the overall complexity.

Moving on the frame features, explained later in the details, they have to be related to the velocity information. From the moment images are, for definition, shots of instants of the real world, they are not related to motion. Possible ways to introduce the notion of time into frames are to analyze multiple consecutive frames at known time or to work on the exposure time. Increasing the exposure time of a camera means introducing motion blur effects. As it is possible to read in this thesis, if the motion blur effect is evident enough, the second approach is the one with best performances. In order to work on blurred frames, algorithms based on FFT can be used. Moreover, switching to the Fourier domain, the computation speeds up considerably.

In the image frames shown in the next chapter it is possible to clearly see the direction of movement of the ground with respect to the car. What is not so straightforward is the relation between samples information and speed. In particular, the second derivative of the image is necessary to analyze the velocity vector magnitude. The concept here is to measure the amount of blur in the image, or better, the rate of change of pixels in the frame.

As final result, the output is composed by two quantities: the direction, (t) in degrees units, and the magnitude, $|V(t)|$, in m/s units, of the velocity vector. In such a way the final user can use the GVS as a stand alone device retrieving the velocity information.

As proof of the readiness of the implementation of such a sensor, it is possible to find few new applications in very high quality racing cars and competitions [19]. For example, in the Formula 1 2021-2022 tests, a light under the car floor by RedBull Racing Team is present Fig.2.5. The best assumption on what it could be used for, is that there is a camera sensor collecting frames of asphalt mixture for processing. In fact, lights are largely used in combination with cameras to enchant frames features and equalize the illumination. The best purpose to use a camera in that position is to measure the velocity vector [17]. Moreover, also in the summer 2022 FS Italy event it was possible to see a red light under Tallinn University car. Also in this case, the best assumption is that the sensor mounted is used to retrieve the velocity vector magnitude of the car. In this case a slightly different technology is used: using the Doppler effect is possible to measure the car speed but the system does not focus on direction analysis.

Figure 2.5: RedBull Racing Team shot of the light illuminating the frame of the velocity sensor and the FS Team Tallinn speed sensor solution.

3

Simulator and data acquisition

The goal of this project is not just to design but also to implement a real velocity sensor. For this reason it is important to acquire data for the concepts generation, design, implementation and test phases. As explained next in detail, data are used to understand which concept can better perform in a real environment and to validate the final prototype of the GVS system.

The GVS has as its first application the Formula Student car of the University of Trento and, in particular, a driverless racing electric vehicle. Though a real vehicle is available for the data acquisition, scheduling with the team the dynamic tests is hard. Unfortunately, due to the restricted time the team has during the year, the track test session necessary to the GVS system implementation is not added to the planned ones. Anyway, it is faster and a more flexible approach to start the design in a simulated environment.

In the next paragraph, the simulator characteristics are discussed. The virtual physics proposed is very similar to the real one in terms of vehicle dynamics and asphalt mixture features. All the data used from now on are to be considered acquired in a simulated environment defined as follows.

3.1. The simulator

The simulation environment is based on Unreal Engine [28] and customized by the Antemotion team, a joint venture of EnginSoft Spa (Italy), LHP Engineering Solutions Inc (USA) and V2R Srl (Italy). It is an Italian engineering startup specialized in simulation tools for the Automotive ADAS/AV virtual testing (Training Validation) with strong knowledge on real-time simulators and HIL/SIL systems.

The simulated scene is composed of a complex world including several elements: track, cones, obstacles, buildings and other cars Fig.3.1. For the thesis purpose, what is necessary to focus on are just the camera sensor, the ground asphalt mixture and the vehicle dynamics.

To run the simulator, first of all it is necessary to record a scene. This is done by driving the simulated car on a track. During this first phase, all the dynamics data of the car are acquired and saved in Matlab format thanks to the vehicle model designed by professors, PhD and researchers of University of Trento. The simulation takes into account: time, acceleration, speed, position, forces and moments of each element of the vehicle. The car is divided into sub-blocks as body, wheels, suspensions and others more. What is fundamental for the GVS application

Figure 3.1: Example of Antemotion virtual environment.

design are the data related to the wheels directions, car position and the vehicle speed. What is important to retrieve for the final performance analysis is the instantaneous car direction and speed.

After these data are acquired in different tests taken from FS events (sine steer, ramp steer, acceleration, autocross), the recordings can be loaded and used in the Unreal Engine environment adding cameras and sensors to the virtual car. The power of this approach is to have reliable ground truth data which can be compared with the information sampled by the sensors mounted later in the graphic engine.

At the end of the rendering, the data stored are the front wheel angles, position, vehicle speed and the frames at 60 FpS (Frames per Second) of the car. Finally, the raw data are necessary in the model validation while the frames are used as input to the GVS application.The velocity vector sensor bases its measurement just on the images acquired but it can, in future, be fused with an IMU to compensate for the pitch and roll effects which are neglected in this thesis for the assumptions explained next.

3.1.1. The simulated vehicle

The car used in the simulator is Chimera Evoluzione, as mentioned before, the second car built by E-agle Trento Racing Team, which is now being converted to race in the driverless FS category. The real vehicle is back wheel traction composed of two electric motors of 72 kW each. Torque vectoring and slip controls are not implemented at the moment on this car, simplifying the design of the simulated model. The vehicle in Unreal Engine and Matlab [18] models are configured with the real car specifications to approximate as much as possible the real car behavior.

The virtual car brings all the racing vehicles characteristics in it. First of all, probably the most important aspect for the GVS application, is the height from ground. Due to the fact the Formula Student cars are racing on tracks, the distance between floor and asphalt is small: 5 cm. The CoM, instead, is located at 100 mm from the ground, right under the driver seat on the x axes of the car.

The last relevant aspect about the simulated vehicle is the fact that the overall stiffness is very high. Thanks to this information it is possible to suppose small roll and pitch angles do not affect the data acquired. In this thesis, the camera will be considered ideally not dependent on height and orientation changes. Must be remembered that this is true for track cars but not in case of lower stiffness. Anyway, the vehicle dynamics is very well simulated including also yaw, roll, and pitch due to forces applied to the vehicle meaning also the camera is subjected to these forces. The frames used for the analysis have different orientation contributions but as it will be

possible to see from later plots, they are not affecting the overall performances.

A further upgrade of the system should also introduce an IMU in the analysis to compensate for these effects. Anyway, it is possible to integrate the acceleration information later with the already present inertial sensors mounted on car.

3.1.2. Ground asphalt mixture

The choice about the camera position is the following: pointed to the ground, the closer possible to the CoM of the car. The camera acquires samples of the asphalt mixture, and so, ground is one of the most important elements to analyze. From the moment the data are sampled in Unreal Engine, the asphalt texture is built in computer graphics. Here, some requirements have to be met in order to make the GVS application working and not exceed the real world characteristics, giving the chance of studying an image similar to the one acquired in the real environment Fig.3.2.

Figure 3.2: Example of the frame acquired.

The mixture cannot be homogeneous because the analysis is based on the motion blur features [6]. To detect the edges related to the car movement, the algorithm needs to analyze the over exposure of stones in the compound. Plain texture in the frame does not produce motion blur to be studied. The best situation in this case is to have a lot of different gray scaled pixel values. Moreover, if very different elements are present, they can be easily detected and tracked in the image processing phase. In order to match these requirements, the asphalt mixture is randomly generated and presents colored elements.

Another aspect to take into account is the light in the scene hitting the asphalt. Shadows can be helpful if they are constant in time, otherwise they introduce noise in the frames. To avoid this condition the camera is positioned in a very dark place, under the car, where external light does not affect the environment. Due to darkness, anyway, a light source is needed to enchants the features. It is important to choose the correct intensity of light to not introduce pixel values too near to 255 level. This kind of over exposure can deteriorate the frame information due to out of scale pixel values.

In the simulator it is possible to choose just the diameter and the intensity of the light source illuminating the scene so white light is adopted. A different solution can be an infrared source enchanting the features without introducing the over intensity issue.

Mounting an external light right near the camera can also be helpful during night events where light can be not enough to optimize the analysis. Moreover, this method constraints the acquisition environment standardizing it whatever the external world conditions are.

3.1.3. Camera acquisition system

The pinhole camera model is used to sample the ground flowing under the car in the GVS system due to the fact this approach can be replicated in reality at low cost. Other cameras present in Unreal Engine have too many configurations and high complexity, more difficult to find in normal real embedded acquisition systems. As mentioned, at this point the camera is mounted on the simulated vehicle and it is used to acquire the asphalt mixture frames while the car is running the tests. The final outputs are compared with the data collected by the Matlab simulator model to analyze the GVS performances.

From the moment the camera has to acquire the asphalt, it cannot be placed in positions where the floor covers it. On the other side, more the camera is close to the front wheels axes and more large the blur direction features to track are evident. Mounting the sensor near the CoM, instead, means acquiring more reliable information. Following these assumptions, the choice is to mount the camera on the CoM pointing the camera to the floor at a height of 50 mm from ground.

The position of the camera chosen is reflecting the real car behavior giving a more direct measurement. It is also possible to scale the system to the best real configuration just changing the acquisition parameters during the calibration process.

3.2. Camera model and motion blur

The GVS system bases its analysis on image frames acquisition. For this reason, both in the simulated and real world environments the camera model must be defined. The most famous and largely used pinhole camera model is adopted and defined now.

This approach is the most direct abstraction of the real world collecting it in $2D$ samples. From the moment that also in the simulated environment it is possible to define all the parameters of this particular kind of acquisition model, the configurations defined as following are working for both the simulated and real application.

3.2.1. Pinhole camera model

Camera models are used to represent a good approximation of the real world, recording $2D$ projections of $3D$ scenes. Thanks to a parametric representation they give the possibility to describe the world that a human eye can see [7].

The parameters used to configure a camera are very important to completely define the model and to replicate the acquisition system in different vehicles and environment conditions.

Due to the intrinsic discretization of digital systems, every model is a lower level approximation of reality. This is reflecting the concept of CCD (Charge-Coupled Device) and pixels. The transformation performed is the following (3.1), (3.2), where X, Y, Z are the real world coordinates and x, y the ones related to the camera frame:

$$f : \mathbb{R}^4 \rightarrow \mathbb{R}^3 \tag{3.1}$$

$$f : (X, Y, Z, t) \rightarrow (x, y, t) \tag{3.2}$$

The pinhole camera model is schematized in the following figure Fig.3.3.

The characteristics of this model have to be explained in order to understand its functionalities and how the image used for the GVS analysis is generated:

- If the objects is far, it appears smaller;

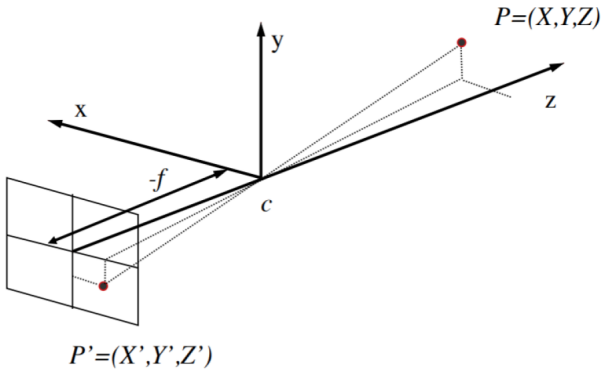

Figure 3.3: General pinhole camera model.

- Parallel lines converge to a single point;
- Parallel lines on the same plane lead to collinear vanishing points;
- The line generated by collinear vanishing points is called horizon;
- Vertical lines are perpendicular to the horizon.

Moreover, there are two possible types of projection:

- In the perspective one all rays pass through the center of projection, corresponding to the lens. To avoid picture flipping, usually, the image plane is considered on the same side of the real world object. Moreover, to simplify the model, the center of projection corresponds to the origin of the $3D$ space and the plane (X, Y) is parallel to (x, y) as shown in Fig.3.4.

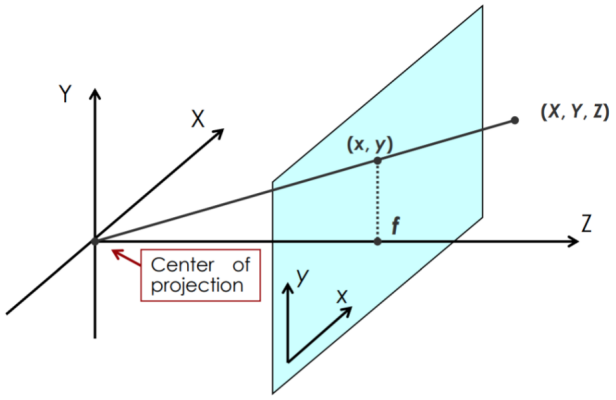

Figure 3.4: Perspective projection of the pinhole camera model.

These approximations are allowed just in case $Z \gg f$, with f the focus length. In the described model the following equations (3.3) hold;

$$x = \frac{fX}{Z} \qquad y = \frac{fY}{Z} \tag{3.3}$$

- In the orthographic projection all the rays originating from the $3D$ scene are parallel to each other Fig.3.5. In this case the distance of the object from the camera does not affect the intensity of the image projected in the $2D$ plane.

  This approach is a good approximation only when the distance of the object is much bigger than the depth of the object itself. In this case the equation (3.4) holds.

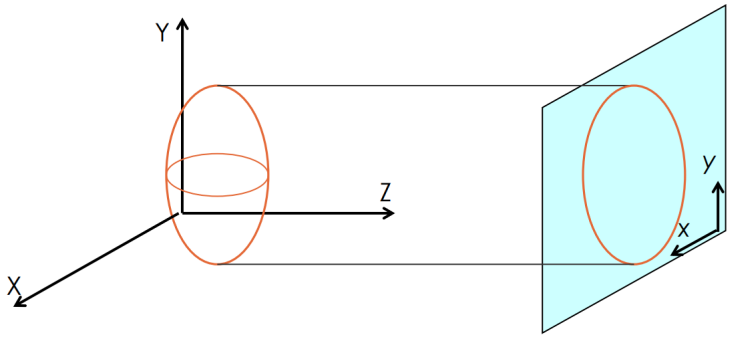

Figure 3.5: Orthographic projection of the pinhole camera model.

$$x = X \qquad y = Y \tag{3.4}$$

Due to the fact cameras are usually designed for general purposes, the most used approach is the perspective projection model. Also in the GVS case it is the adopted one but, due to the fact the camera is very near to the ground and the asphalt texture elements are almost flat, the final frame appears as an orthographic projection. This is true also in the real world environment.

Camera focus defines the amount of blur of a static image. It depends on the distance between camera and image plane: moving the image plane without changing the focus means acquiring an out of focus sample containing blur, and so, losing edges information. In this situation, for example, a point becomes a circle which is not a good approximation of what the real information is. From the moment the camera models are themselves approximations, there is not a way to acquire completely unblurred images but a threshold is set to define the focus condition. In the GVS case it is important to acquire a focused image due to the fact that the motion blur effect is used. These features are very similar to the ones generated by an out of focus image, introducing noise difficult to be filtered during the analysis.

The depth of field, instead, is the range in which it is possible to retrieve a focused image and it is related to the specific camera, or better, lens.

Another parameter to control, that produces blur in the frame, is the camera resolution. It depends on the number of $N_x \times N_y$ CCD of the sensor. The choice of the right resolution is related to the number of details the application should detect. In particular, the physical resolution can be determined using the distance between two dots or lines that move to each other with a certain gap. If the function value at the gap is less than 50% of the value at the dots, the dots count as resolved. This is illustrated in the next image Fig.3.6 where the green line shows the original dots with infinite resolution and the blue function shows the result after performing acquisition with a certain resolution. In this example case the spatial resolution is around $1mm$ [23].

The image resolution, instead, is given by the size of an image pixel. If the $M_x \times M_y$ image is associated with a certain image size (width w and height h), the resolution can be also reported as pixel size (3.5). The reciprocal is usually used as PpI (Pixels per Inch).

$$\Delta x = \frac{h}{M_x} \qquad \Delta y = \frac{w}{M_y} \tag{3.5}$$

Another important aspect related to camera acquisition is the illumination. Too high light environment intensity results in over exposure of the frames meaning that the image information will be lost due to saturation at high scale pixel levels. On the other side, too dark frames result in saturation at low pixel levels meaning the camera acquires almost black images. This case

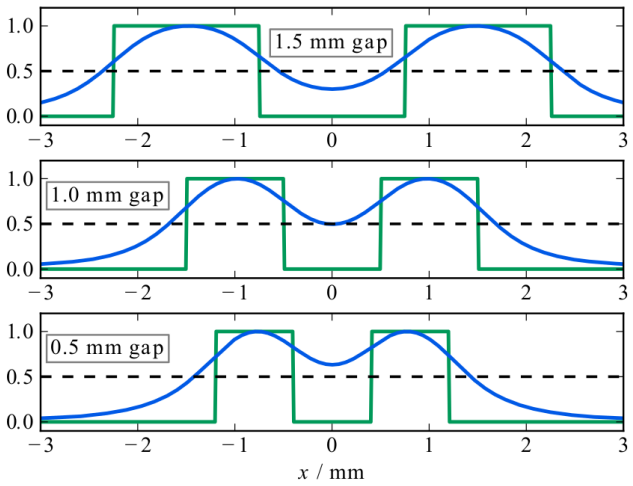

Figure 3.6: Example of spatial/physical resolution analysis.

is illustrated in the following histogram Fig.3.7. What the acquisition should be, instead, is an histogram spread all over the range $[0, 255]$. In this situation it is easier to analyze the image pixel levels and so the collected information.

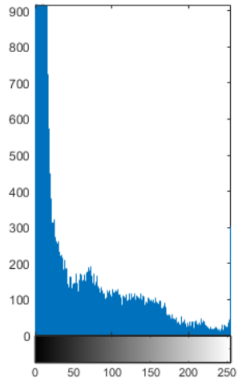

Figure 3.7: Example of histogram of a too dark image. On x axes the pixel intensity and on y axes the number of pixels of every value in range $[0, 255]$.

In general, the light hitting an object can be absorbed, reflected or transmitted. Perception of objects is possible thanks to reflection that can be specular, concentrated in the light source direction, or diffuse, constant in all directions. Surfaces vary in specularity and some of them can be matte or glossy. In the GVS case, the best case is to illuminate the all scene, the asphalt mixture image, constantly. It is possible to assume the camera's final position is under the floor of the vehicle, a very dark environment. For this reason it is fundamental, as mentioned before, the use of a light source covering all the acquired area. Using just contrast enchanting algorithms can be not enough to detect all the relevant features of the image. Moreover, these algorithms are elaborating every pixel increasing the computational time. The light should have an intensity which does not over or under expose the image already in the physical environment, before acquisition. There is a trade-off between the area size acquired and the light size: bigger the image is and bigger the area covered by the light should be to generate an equalized image. Increasing light source size also increases the space required by the hardware. A good solution could be a led ring placed around the camera but due to the fact the application is at the moment designed and tested in a simulated environment, a point light with a big diameter is used. As mentioned previously, using an infrared source could be a good choice to avoid too high intensity

pixel levels.

Illumination is also related to contrast. Low contrast means the image histogram does not cover all the intensity range. As it is possible to see from the following pictures Fig.3.8 a low contrast makes the analysis and the edge detection more difficult due to the high difference in the image pixel values.

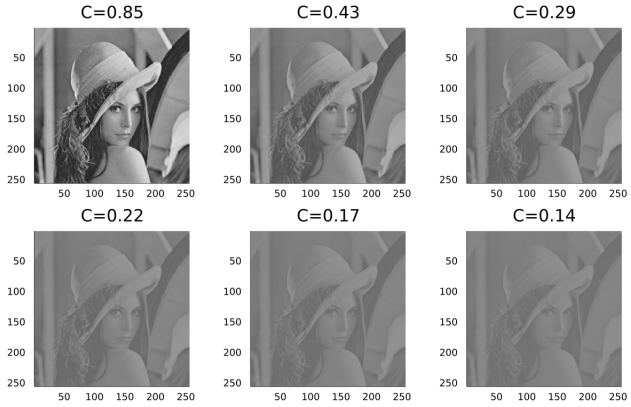

Figure 3.8: Example of different contrast values.

With $f(x, y)$ the pixel intensities, usually, the contrast is defined by the Michelson contrast (3.6). To improve contrast, it is possible to equalize the image via software or to use light sources to improve the illumination of the scene.

$$C(f) = \frac{max(f) - min(f)}{max(f) + min(f)} \tag{3.6}$$

The last but not least problem the frame acquisition can have is the amount of noise present in the image. The next images Fig.3.9 show how this parameter can affect the acquisition. Defining $f_{true}(r)$ as the true noise free image and $\epsilon(r)$ the noise image that contains in each pixel an uncorrelated random number with a certain mean and standard deviation, the image contaminated by noise can be modeled by (3.7). To fix noise artifacts several software approaches are present in the literature and most of them are based on morphology and filter convolutions.

$$f(r) = f_{true}(r) + \epsilon(r) \tag{3.7}$$

This overview of the camera model and its characteristics is necessary to define the most suitable camera parameters for the GVS system. In particular, in the following table Tab.3.2.1 it is possible to see which are the suggested parameters. The height of the sensors from ground is set in order to have a 50×50 mm^2 square frame collecting the asphalt mixture features. Too big image size means more calculations, not necessary for the GVS application, increasing the overall algorithm computational time. Focus and light parameters are set experimentally in order to have clean and enough illuminated pictures.

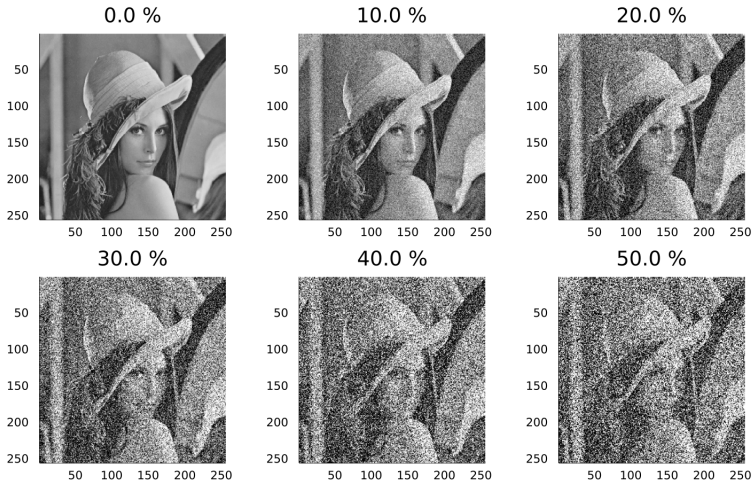

Figure 3.9: Example of different amount of noise.

x position	On the front wheels axes
y position	0 mm
Height from ground	50 mm
Orientation	Pointing ground
Focus length	50 mm
Focus mode	manual
Exposure time	3 ms
Camera model	Pinhole camera
Camera projection	Perspective projection
Resolution	1920×1080 px
Image covered area	50×50 mm^2
Light position	Ring around camera
Light diameter	70 mm
Light intensity	0.5 cd

3.2.2. Motion blur

The GVS application works under dynamic conditions. This means the application must be time dependent and so also the camera frames acquired. In particular, the domain is $\mathbb{R}^3$ as defined previously (x,y,t). The parameter related to time is the exposure: the period of time the CCDs are acquiring [15], [1]. Usually, the desired exposure is very fast in order to collect a sharp instant of the real world. In the GVS case, instead, it is important to retrieve information about the velocity. Increasing the exposure time also means that it is not possible to have data for a certain amount of time, when the CCD is sampling. This statement sets a trade-off between sample rate and exposure time.

Increasing the exposure time means also generating blurred images because the intensity values are collected while the camera or the object plane is moving and the shutter is open [26]. The blur, usually undesired, in this case collects the needed information about time and the dependency to how fast the object moved with respect to the camera. To see an example, the two following figures Fig.3.10 are showing two frames taken from the same camera on a car at 0 m/s speed and moving.

As it is possible to see, the blurred image brings in not only the information of time but also

Figure 3.10: Example of sharp frame at 0 Km/h and motion blur affected image at 30 Km/h.

the direction of motion relative to the camera frame. The biggest issue is that if only the blur is used for the GVS analysis, in the case of near stop condition, the measurement is undefined because blur is not evident enough. For this reason, as explained later, the frame difference methods are introduced in the analysis of the velocity vector at low speed.

Studying which can be the right exposure time there are two possibilities:

- Set a fixed exposure time for all the speed range. The amount of blur is speed dependent by a logarithmic proportion. As explained in the velocity vector magnitude analysis, over a certain velocity vector magnitude, the model becomes almost flat and it is more difficult to measure the real quantity. In the case of speed analysis, an adaptive exposure time should be taken into account. Anyway, changing the camera shutter behavior does not affect the direction analysis from the moment motion blur is present in both exposure cases. In practice, the direction analysis algorithms remain the same if adaptive exposure is used. Taking these considerations into account, the fixed exposure approach is used for the direction analysis;
- Adaptive exposure time. It is possible to shift the speed logarithmic model changing the exposition time. To do this also an estimator of next frame speed is necessary. Adaptability can improve not only the working range but also the accuracy and precision of the speed model. This upgrade can be implemented in future but, for the moment, the choice is to focus the thesis on the algorithms theory behind this approach.

Due to the fact the simulator is not in a released version yet, in the Antemotion environment the motion blur feature is not present during acquisitions. To overcome the problem, the motion blur was generated after the sampling phase thanks to the Matlab motion blur function and the data retrieved by the ground truth of the vehicle model recorded Fig.3.11. First of all, the original picture is loaded. Then the filter mask is generated matching the velocity magnitude and direction information. Finally the image is filtered with the designed mask and saved in a new .png file. These operations are performed for each image acquired from the Unreal Engine tests.

```
A = imread(['../../../test_sine_60fps/frame_' num2str(i, '%d') '.png']);
H = fspecial('motion', amp, ((theta * 180 / pi)) + 90);
MotionBlur = imfilter(A ,H,'replicate');
imwrite(MotionBlur, ['../../../test_sine_7_0_60fps_matlab/frame_'  num2str(i,'%d')  '.png']);
```

Figure 3.11: Matlab script example used for the motion blur generation.

The algorithm performs a low pass filter convolution but instead of using a squared 3×3 matrix, as in most computer vision applications, is adopting a mask that takes into account also direction and magnitude. It is a good enough approximation for the analysis but it can be

interesting to directly acquire motion blurred frames from the simulator or from the real world to validate this model. Speaking with the Antemotion team, developing the simulator, the request about the motion blur implementation was accepted and it will be present in the new release. This Unreal Engine improvement can open new tests for the already implemented algorithms.

4

Velocity direction analysis

As treated in the introduction, the measurement of the velocity vector direction requires complex algorithms and a lot of hardware. The GVS system, instead, uses just one camera for the acquisition of blurred frames of the asphalt mixture.

The design of the GVS sensor started from low speed acquisitions, under 20 Km/h, where motion blur effect is not sufficiently evident to be studied. In this condition, the methods can be based on static frame features only. The information about time is coming from the knowledge of the set frame rate, in this case 60 Fps (6.1). A period of 16 ms matches the output frequency requirements, as described later in the overall performances section. In the real racing car application the ECU (Electric Control Unit) usually needs one set of sensor samples every 20 ms. This real-time constraint is set in order to match the dynamic controls under development in the team. A slower rate means delaying the overall control or using old data to guarantee the car stability. The time margin after acquisition is occupied by the computation phase.

$$SampleTime = \frac{1}{SampleRate} = \frac{1s}{60Fps} \approx 0.016s \approx 16ms \tag{4.1}$$

.

Due to the fact the slow speed approaches base their analysis on consecutive frames, a one step memory is necessary. At low speed, the GVS internal output is relative to the frame acquired 16 ms previously.

At higher speed, the motion blur is present, and other algorithms are designed in order to use edge information improving the overall system performance. Moreover the implementations are based on single frames reducing the limit acquisition period to the exposure time, in this case 3 ms.

Starting the analysis at low speed ranges, two possible algorithms and implementations are proposed:

- The pure and manual frame difference. In this approach the displacement vector between two consecutive frames is retrieved. Center point features are necessary in order to measure their difference vector. This method is working good in the low speed range but it can be improved in precision and accuracy;
- AI based algorithm. In this case, the key points necessary to measure the displacement vector are found in an optimal way by the vision algorithm. The features are also weighted

and they can be labeled by ID (Identifier) in consecutive frames. With this information it is then possible to retrieve the velocity vector direction.

These two approaches are based on key points detection, intrinsically meaning that they perform better with sharp images. This condition decays when speed increases due to the motion blur effect. For this reason, at speed higher than 15 Km/h, other two concepts are defined for the design:

- The algorithm HoG based, analyzing the first derivative direction of each pixel. When the image is blurred, it is possible to detect a main direction of stretch which is given by the car direction of movement. The histogram divides the orientations of the image in bins. The final distribution is a Gaussian where the mean value is the most probable direction of the car. This method performs very well but it is slow due to the amount of computation required to generate the histogram. Moreover this method introduce a discretization due to the bins size;
- The FFT based algorithm. Here, switching to the Fourier domain, it is possible to split the image information in magnitude and phase. In particular, magnitude, bring in artifacts which are related to image orientation. This behavior, usually undesired, becomes very useful in the GVS application. Using FFT computational time decreases considerably and the discretization is reduced to the pixel size.

In the following sections, all the cited algorithms are exposed in the details. In order to test the concepts and produce a working prototype of the GVS they are also implemented in a C++ project.

4.1. Two models for high and low speed

The static analysis treated in the next sections decreases in performance with the increasing of speed. The issue is related to the image amount of blur and saturation of stretched stones in the frame. As it is possible to see in figure Fig.4.1, the asphalt texture, at very high speed, can exit the scene making the detection of features in consecutive frames more difficult.

Figure 4.1: Frame with amount of blur producing image saturation.

To overcome this problem different considerations on the exposure time are done. Changing the configuration of the camera model, increasing or decreasing exposure time accordingly with the vehicle speed, can be a good approach. With this technique it is possible to move the frame to the blurred or unblurred condition and use just the FFT method explained later. At very high speed, this means to use very short camera time exposure, possibly also not supported by the sensor. At low speed, instead, this means that it is possible to not detect the information for a

time longer than the one required from other devices in the car network architecture. In order to develop a reliable prototype, the first approach is to fix the camera model parameters, the exposure time in particular, and set a transition speed in the software when switching from static to dynamic algorithms. This is possible due to the fact the speed can be used as a reliable reference for all the working range. Must be remembered that having the same exposure period can create problems at very high speed where the image becomes completely homogeneous.

To set the best exposure time, the following calculations are performed. Considering to use the frame difference approach up to 40 Km/h or 11.11 m/s and basing the analysis on a squared frame of size 0.05×0.05 m ^{2}, the concept is the next. The features, also the slightly blurred ones, must be contained in the image. To make this possible, the features must lie inside the circle area contained in the frame with a margin of the 25 % with respect to the biggest diameter of the circle contained in the image, as shown in the following figure Fig.4.2.

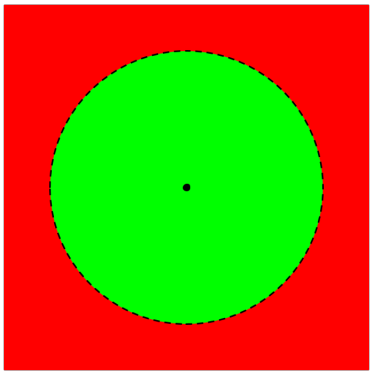

Figure 4.2: Area in which, up to 40 Km/h, there are features to track.

With these assumptions, the biggest feature blur size should be around 0.0375 m long. Inverting the velocity law the most suitable exposure time is computed(4.2).

$$Speed = \frac{Space}{Time} \rightarrow Time = \frac{Space}{Speed} = \frac{0.0375m}{11.11m/s} \approx 0.003s \tag{4.2}$$

A 3 ms exposure time is used to guarantee a clean image up to 40 Km/h. From the moment the motion blur effect is present also before this threshold and the dynamic algorithms have better performances, in the GVS system the method switching is done at 15 Km/h. In the following sections it is possible to see how much the deterioration of measurement is evident and understand the importance of changing the concept from static to dynamic analysis.

4.2. Frame difference method

The basic concept to develop the manual frame difference algorithm is to define relevant features of the static asphalt texture and track them over consecutive samples. The key points centers positions are necessary to calculate the mean displacement vector of the actual image with respect to the previous picture. The angle defined by the vector is the same as the velocity one with respect to the previous frame.

Moving to the details of the GVS FD implementation, first of all, the frames are equalized and preprocessed to enchant the features detection. Key points of interest must be defined: looking at the static asphalt texture, it is possible to see that the mixture is composed of different stones, each of them with a size and a color, mainly in the grayscale range. Sometimes, accent colored stones are present in the scene. Unfortunately, this second information, also if it is more easy to

detect, is not reliable. Starting from these assumptions, the gray intensity attribute is taken into account for the preprocessing phase. Thanks to this approach it is possible to reduce the overall computation from the moment the output is based just on a small relevant subset of images information. The parameter set on filters must be the same for the two consecutive images, otherwise, the key points positions are not comparable. This condition must be considered as possible failure but the threshold set on the algorithm steps generates a large enough number of positions to be averaged mitigating the overall final error.

The software implementation starts loading the current frame and the previous one Fig.4.3 acquired by the camera sensor mounted on CoM of the vehicle. From now on, the first frame will be intended as the actual one while the second the oldest. The images are acquired in a RGB (Red Green Blue) color scale and the sample size is of 5×5 cm^2. As mentioned, the sample rate is 60 Fps and so the two images are ≈ 16 ms older one with respect to the other.

Figure 4.3: Actual (left) and previous (right) frames acquired at 5 Km/h for the direction analysis.

The color domain is shifted from RGB to HSV (Hue Saturation Value) Fig.4.4. This transformation gives the possibility of working, instead of pixel color values, on three separated channels describing hue, saturation and value, or brightness, of the image intensities. Also the HSV domain is an additive model whose primary colors are red, green and blue that are described on the cylinder base [5]. On the radial axes saturation increases linearly while in the vertical one the brightness is described.

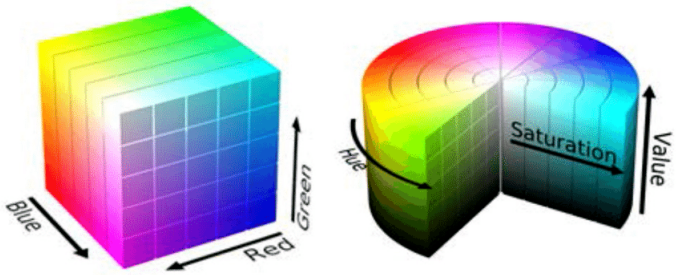

Figure 4.4: RGB to HSV domain change.

Thanks to the domain shift, it is possible to filter the relevant features. The following images represent the transformed sampled frames Fig.4.5. In the HSV model, the colors near to accent colors, with high brightness, are highlighted, while the ones more near to dark are less evident [21]. Differently to the RGB domain, here it is possible to use just one of the three channels in a consistent way.

Working on high brightness means keeping the elements more near to white, discarding the more dark ones. A threshold is set according to this assumption on the samples in order to

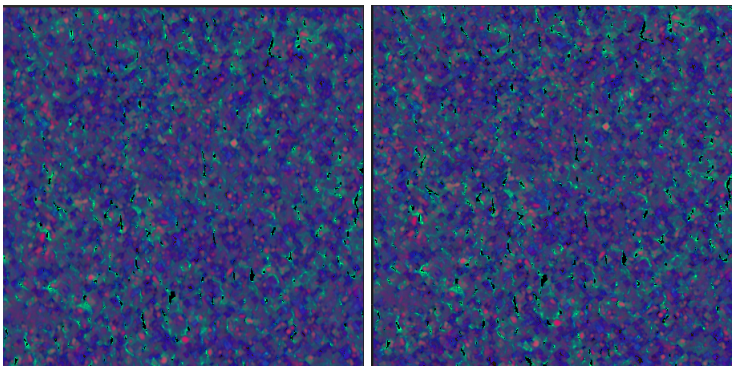

Figure 4.5: Actual (left) and previous (right) acquired frames in HSV domain.

use just the accent colors in the frame Fig.4.6. The elements in range [(0, 0, 50), (255,255,255)] are taken into account keeping the information of hue and saturation but discarding the low brightness areas.

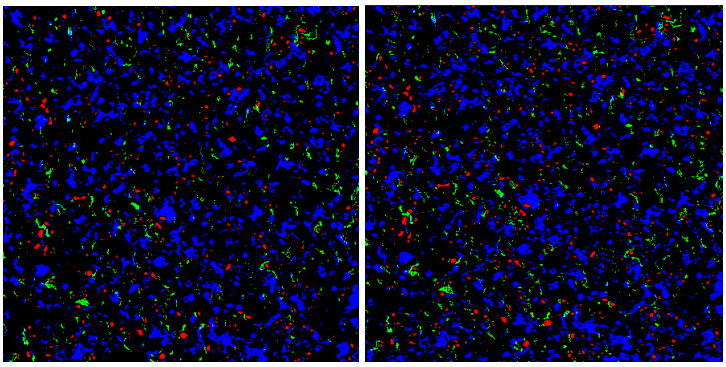

Figure 4.6: Actual (left) and previous (right) HSV frames after threshold.

Now it is possible to binary the image by applying a new threshold on colors Fig.4.7. Here just the most high pixel values are taken into account as 255 intensity pixels. This step is necessary, not just to work on binary samples, which are faster to be elaborated, but also to reduce the number of elements in the scene. Computing FD on the filtered HSV domain can be so slow to push the system out from the requirements of 50 Hz set for the output rate.

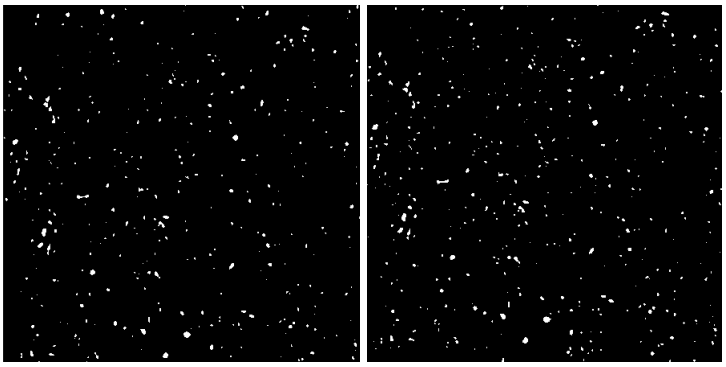

Figure 4.7: Actual (left) and previous (right) binary frames.

As it is possible to see, some elements are close one to each other but also very small. Erosion is performed to delete small particles not easy to detect and producing noise [4]. These elements are not reliable data on which to perform the computation from the moment, in the second sample, they could be not detected at all. Erosion, as the name suggests, also makes

the blobs of interest smaller. Dilation is performed to make too near neighbors high value pixels just one single blob, improving the algorithm efficiency and the center detection reliability Fig.4.8.

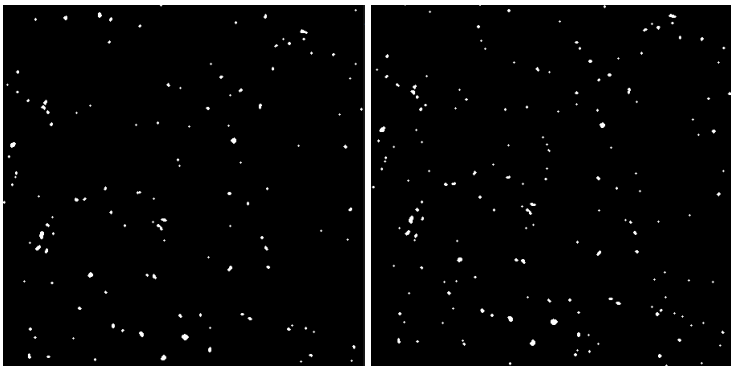

Figure 4.8: Actual (left) and previous (right) frames after morphology transformations.

Erosion and dilation are performed on each pixel of the frame. This approach slows down the overall computational time needed to produce the output but at the same time is necessary to increase the reliability of the data. The more resources needed with this approach are explained by the fact the mask (4.3) must be convoluted on the entire image. A disk mask is used in the GVS application in order to not disturb the direction analysis. In fact, in this case the mask is not amplifying a particular direction during erosion and dilation.

$$S = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.3}$$

Mathematically, the two operations are described as follows (4.4), (4.5) where A is the original image, S is the mask, or structuring element, a are the pixels of A and s the elements of S.

$$A \ominus S = \{a | a + S \in A, \forall s \in S\} \tag{4.4}$$

In erosion, each position where every 1 of the structuring element covers a 1 of the binary image, the binary image corresponding to the origin is OR'ed with the output image.

$$A \oplus S = \bigcup_{a \in A} S_a \tag{4.5}$$

In more practical terms, in dilation, every time the origin of the structuring element touches a 1 of the image, all pixels of the structuring element are OR'ed to the output image.

Following the computer vision literature, performing erosion and dilation consequently, means calculating the opening of the image (4.6). Thanks to this operation, first, small blobs are removed, and then, residual information is refined.

$$A \bullet S = (A \oplus S) \ominus S \tag{4.6}$$

Once blobs, representing the features of interest, are defined, their centers must be detected in order to compute the displacement vector in consecutive frames. Elements contours are extrapolated using Canny edge detector. The function also defines the area and center of each element. The Canny algorithm bases its analysis on the following assumption: contours are the boundaries of a shape with the same intensity. Looking at the white to black change of intensity, the method collects each point of the element boundary in (x, y) coordinates. From the moment

the contour is known, it is also possible to compute the geometric characteristics of the element as perimeter, area and more important, the center.

An edge detector should have a low error rate, which means that the detection should accurately catch as many edges as possible. At the same time, the edge point detected from the operator should accurately localize on the center of the edge. A given edge in the image should only be marked once, and where possible, image noise should not create false edges.

Going in the details of Canny detector [24], to match accurately the requirements of edge detectors, the algorithm perform as following:

- Apply Gaussian filter to smooth the image in order to remove the noise. These kinds of filters are low pass and they smooth the frame, decreasing the sharpness of edges but also deleting the noise blobs in the acquisition. The box filter K is mathematically defined here (4.7), where k is the amount of blur to be applied.

$$K = k \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{4.7}$$

  As it is possible to see from the mask, the Gaussian is centered in the middle of the filter. Gaussian filter operation is defined by (4.8), where (x, y) are the pixel coordinates;

$$g(x, y) = ce^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{4.8}$$

- Find the intensity gradients of the image. The gradient represents the first derivative in horizontal G_x and vertical G_y directions. Due to the fact the computation is easier to be handled if polar coordinates are used, the following transformations are performed (4.9), (4.10);

$$G = \sqrt{{G_x}^2 + {G_y}^2} \tag{4.9}$$

$$\Theta = atan2(G_y, G_x) \tag{4.10}$$

- Apply gradient magnitude thresholding to get rid of spurious response to edge detection;
- Apply double threshold to determine potential edges;
- Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

At this point, the position of the centers are known in both frames and it is possible to compare them in order to find the displacement vector. The approach is similar to a key nearest neighbor algorithm. The software, for each center of the first frame, checks in a certain circle area if there is another key point in the second image. It is important to perform the analysis in a rounded shaped area in order to avoid relations with the direction of computation. When all vectors are retrieved, it is possible to calculate the mode of the displacement angles to find the most probable direction of transformation of the frame Fig.4.9. This direction is the same as the one the CoM follows in the last 16 ms. In the following figure it is possible to see the displacement vectors between centers taken into account, in blue, the centers of the previous frame, in green, and the ones of the actual frame, in red.

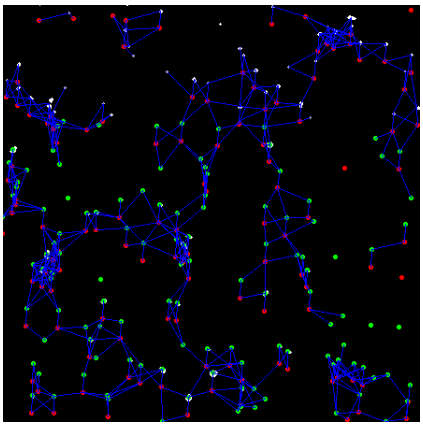

Figure 4.9: Frame difference result.

This measure is relative to the one step older frame, meaning that the final car direction, with respect to the ground, is given by the sum of the angle of rotation that the vehicle has at the previous step and the actual direction (4.11).

$$AbsoluteAngle_{deg} = PreviousAngle_{deg} + NewAngle_{deg} \tag{4.11}$$

This method depends on the frame size and the number of features discovered. A larger area means more features to track. This reflects, in the camera model, the change of the camera height or of the camera lens, changing also the illumination level of the image. More distant the camera is from the ground and more light intensity is necessary to have the same brightness level in the frame avoiding not equalized frames. Due to the fact the application is related to racing cars, the assumption is that the height from ground is small and also the height in the previous table is a good value for the application. Moreover, increasing the computational time can push the output rate out of the working range described in the requirements section.

It is also possible to improve the software quality by increasing the number of detected features or analyzing different image characteristics. Intensity is not the only feature a vision algorithm can track.

Looking at the next plot Fig.4.10, some considerations can be stated. At low speed the accuracy and precision are good but increasing the speed, and so the amount of blur, the system loses performance. This is due to the fact, using constant exposure time, the motion blur effects produce big contours and so the centers are not precise and accurate. Moreover, precision of the data acquired is related to the amount of features. This method is designed to work with a high number of small blobs. Some outliers are present but they can be filtered using the history of the data thanks to a moving average filter. The choice is to leave the data as computed and give the possibility of working on this raw information at a higher level. Anyway, the outliers are still in the range set by the requirements.

As it is possible to see, the algorithm performs well just at low speed. For this reason the FD approach is intended to work at maximum ≈ 15 Km/h.

In particular, the characteristics of the frame difference algorithm are listed below Tab.4.2:

Mean	45.01 deg
Standard deviation	0.2536 deg
Mode	45 deg
Min value	44.38 deg
Max value	46.4 deg

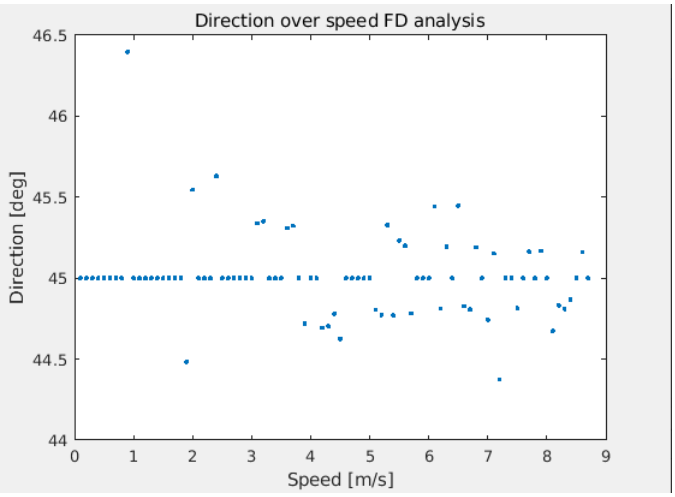

Figure 4.10: Frame difference velocity vector direction analysis.

4.3. Oriented fast and Rotated Brief descriptor method

In the manual frame difference method, the most relevant limit is related to the features detection efficiency. Several transformations are necessary in order to detect the blobs of interest [25]. Moreover, they can be inconsistent and not find in both frames. Finally, the computation time is high due to the necessity to perform image filtering and more importantly, matching the centers manually. All these considerations bring to a new concept maintaining the approach of analysis on consecutive frames changing the way to find the relevant key points.

In the computer vision literature different key point detectors are available, for example SIFT (Scale Invariant Feature Transform), SURF (Speeded Up Robust Features) and ORB (Oriented Fast and Rotated Brief). Each of these algorithms aim detecting a set of features based on different common characteristics. The most useful one in the GVS implementation is the ORB detector.

A feature detector is composed by two components:

- The locator identifies points on the frame that are stable under transformations like translation, or shift, scale, increase or decrease in size, and rotation. It finds the (x, y) coordinates of such points. In particular, the locator used by the ORB detector is called Fast;
- The locator only defines where the points of interest are. The descriptor, instead, encodes the appearance of the point so that we can tell one feature point from the other. The descriptor evaluated at a feature point is simply an array of numbers. Ideally, the same physical point in two images should have the same descriptor. ORB uses a modified version of the feature descriptor called Brisk.

At this point the detector has all the information: the position of each feature point and the relation between points in consecutive frames.

Once at least four points are detected in both frames it is possible to calculate the homography H of the two images. This concept is mathematically described as follows (4.12) and it describes the rotation matrix between the two samples acquired.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \tag{4.12}$$

Usually, this approach is used to restore rotated images but in this case, retrieving the homography gives enough information to define the angle of rotation between the first and second frame. The measurement is relative to the previous frame and the sum of the angle at the previous step and the one calculated must be performed to have the final absolute vehicle direction with respect to ground.

In the GVS implementation Fig.4.12, after frames acquisition, there is no need to filter or enchant the images from the moment the ORB detector can work with the image as it is, in the RGB domain. After matching key points of features are stored in the vectors, the angle defined by each of them is computed. At the end, all angles are averaged to retrieve the direction of movement of the car.

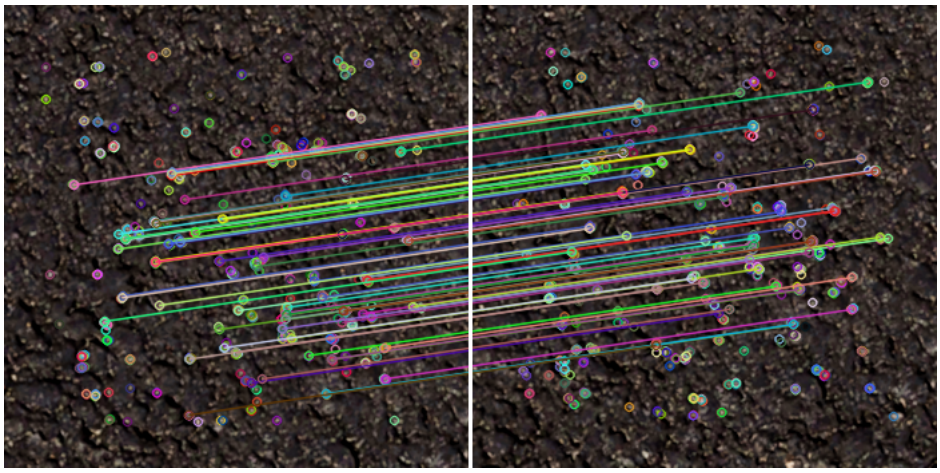

Figure 4.11: ORB detector frame difference pixels matches in GVS application.

The steps required are way less than the ones of the manual frame difference method. In this case, not only computational time decreases considerably but also accuracy and precision are higher. In the following plot it is possible to see the overall algorithm performances.

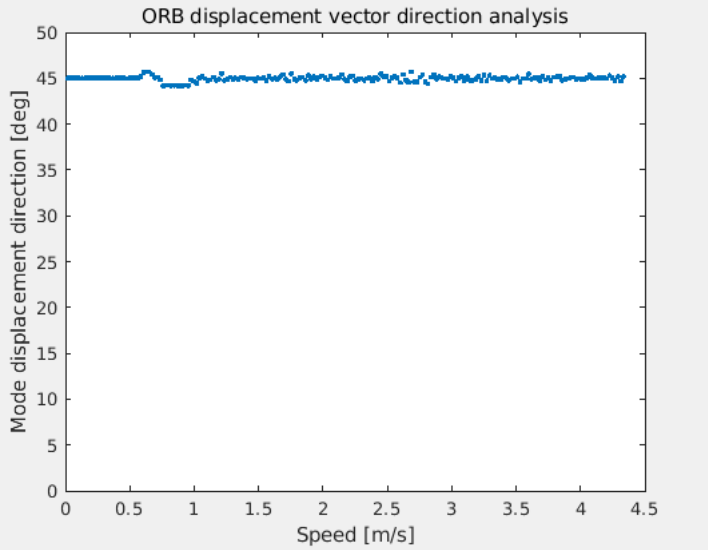

Figure 4.12: Frame difference ORB direction analysis at constant 45 deg orientation.

Increasing speed over the 4.5 m/s blur effect deteriorates the analysis. Anyway, this approach, thanks to the ORB descriptor, is a more robust method for the choice of the kind of feature of interest to detect. In the following list Tab.4.3, the characteristics of a constant 45 deg

test are exposed.

Mean	44.97 deg
Standard deviation	0.278 deg
Mode	45 deg
Min value	44.14 deg
Max value	45.71 deg

4.4. Histogram of Gradients method

To overcome the deterioration of the static analysis due to the motion blur effect at speed higher than 15 Km/h, the concept has to move to another method, based on just one frame, thanks to the fact the time information is fully contained in the image. For this reason HoG is implemented in the GVS design.

First of all the blurred frame is acquired with a camera exposition time of 3 ms Fig.4.13. The concept behind this implementation is that moving the camera with respect to the ground produces blur in the same direction of the moving car.

Figure 4.13: Frame used for HoG analysis.

As for the static analysis cases, the frame color scale is converted to HSV for feature enchantment. Here, just the saturation channel is used for the computation from the moment the features of interest are related to saturation intensities Fig.4.14. Thanks to this choice, it is possible to work on just one third of the information contained in the frame, speeding up the overall computational time.

The Gaussian low pass filter is used in order to delete undesired outliers. From the moment the features of interest are, for sure, stretched, they are not represented by points. The filter, smoothing the picture, helps to connect small elements increasing the edges detection consistency.

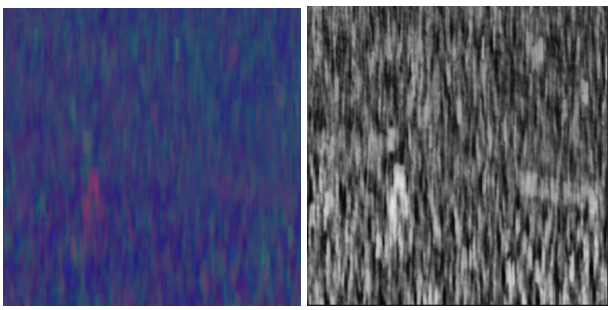

Figure 4.14: Frame in HSV scale on the left and saturation intensities on the right.

The main direction in the image is represented by the most common orientation of edges in

the frame. To detect edges, the gradient, which contains the first derivative information, is analyzed. Instead of a Canny detector, here the Sobel filter is used on the blurred frame. Also in the Canny case, edge detection is performed but, in order to have more control on each parameter of the implementation, the Sobel filter is used directly on the image to find the gradients in x and y directions. This step generates two frames related to the first derivative on x and y of the frame representing the change of intensity between pixels, as known as edges. The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives. Defining A as the source image, G_x and G_y two frames which at each point contain the horizontal and vertical derivative approximations respectively, the computations performed are the following (4.13).

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * A, G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \tag{4.13}$$

In the representation Fig.4.15 is possible to see the case in which the car is moving straight and the motion blur is more evident in the y direction. Moreover, the Sobel filter binaries the image, decreasing the computational cost.

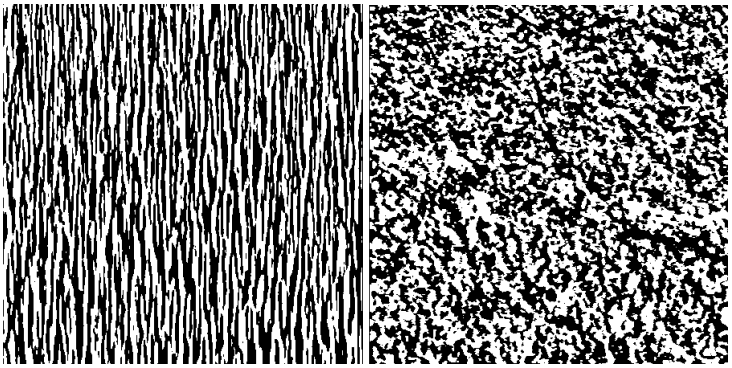

Figure 4.15: Frames filtered by Sobel on x (left) and y (right) directions.

To retrieve the angle direction defined by the gradients, it is necessary to recompose the two images, related to Cartesian coordinates, in one single frame thanks to a coordinate transformation returning two polar images, one for the magnitude G of the pixel gradient and one for the its direction Θ (4.14), (4.15).

$$G = \sqrt{{G_x}^2 + {G_y}^2} \tag{4.14}$$

$$\Theta = atan2(G_y, G_x) \tag{4.15}$$

Once the polar coordinates matrices are retrieved, the histogram of gradients is computed. Defining the number of bins, in this case one for each degree from 0 to 180, and storing the weighted sum of each angle first derivative magnitude in each of them, the Gaussian distribution appears in the plot Fig.4.16.

The value of the bin with maximum magnitude in the Gaussian distribution is the desired direction of the velocity vector. For the case of image orientation at 45 degrees, the data 45 and 135 degrees are describing the same situation due to the specularity of the gradient orientation.

The following plot shows the HoG algorithm overall performances Fig.4.17;

At low speed, under 20 Km/h, the algorithm works with very low performance. This is due to

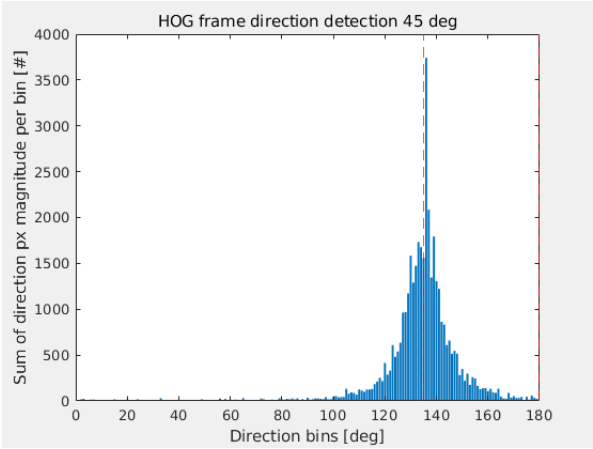

Figure 4.16: Frame HoG analysis for 45 degrees image orientation.

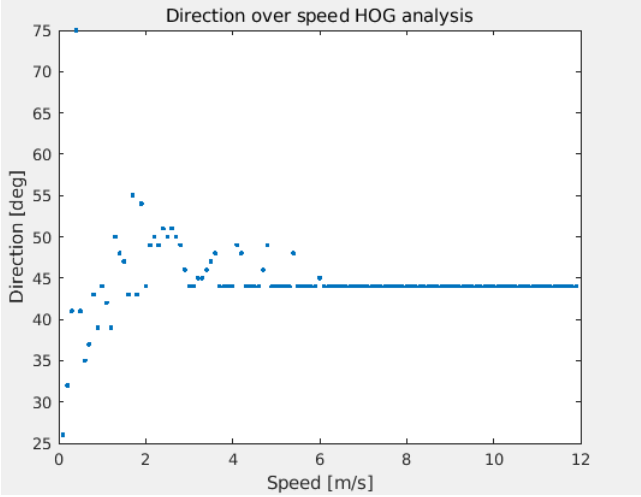

Figure 4.17: HoG acquisition for 45 degrees image orientation.

the fact the blur is not present at low speed and the gradient cannot be computed. Outside the working range, the first derivative is retrieved on small dots representing the asphalt, generating an almost flat histogram. The characteristics of this algorithm, taken as example the constant configuration at 45 degrees, and just considering the defined working range, are listed in the following table Tab.4.4:

Mean	44.36 deg
Standard deviation	1.1 deg
Mode	44 deg
Min value	44 deg
Max value	49 deg

Due to the high computational cost required by the application to generate the histogram and trying to improve accuracy, different tests Fig.4.18 are performed at different frame sizes and aspect ratios. Decreasing the area of computation, also the accuracy of the algorithm changes. Taking as reference the center of the image and making the area of analysis smaller, the histogram becomes flatter, the standard deviation increases and the mean value decreases due to too few numbers of samples taken into account. This condition is also known as under-fitting. Increasing the area, instead, means increasing computational time.

To try to avoid these problems the following approaches are tested. In Fig.4.18 the tests performed are shown. For each of these tests, different areas of analysis are taken into account.

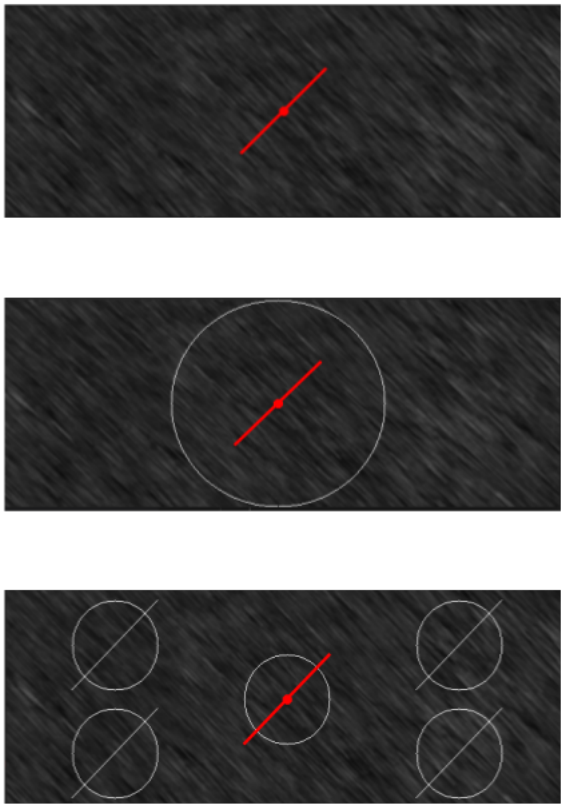

Figure 4.18: HoG optimization tests for 45 degrees image orientation.

Using small areas centered in multiple frame points and averaging the results of each of them, it is possible to increase just a bit the computational time improving accuracy. Due to possible inequality in the image illumination and asphalt texture, some of the point areas return very different values making the average not reliable. This approach should take into account a way to define which the optimal areas of analysis is, for example using the intensity parameter.

Returning to the concept of one single area of study but improving accuracy, the approach is to find the optimal point in which blur is highlighted and compute the histogram on the area around it. As a key feature to define the optimal point, the brightness level is used. Due to the fact that a high level of gray in asphalt mixture means more probability to have a near low level pixel, the software searches for maximum pixel values and computes the histogram on this zone. Also in this case the result is not good enough due to the fact that usually the level of gray is higher but not very different from the overall texture intensity.

After the analysis of each method and its variants, the following table Tab.4.4 reports the result of the performances of each of them where R is the circle radius in which the analysis is performed.

Method	Mean MSE
Averaging all image	± 1.156 deg
Averaging circle area with $R = 100$ px	± 2.625 deg
Averaging circle area with $R = 50$ px	± 3.438 deg
Averaging circle area with $R = maximumR$ px	± 1.375 deg
Averaging 4 circle areas with $R = 50$ px	± 2.195 deg
Averaging 5 circle areas with $R = 50$ px	± 0.963 deg

After all these tests, the best result is using the center frame point. In fact, this is the position right under the source of light and so the one with the best illumination condition. Analyzing the circle area around this point gives the condition in which the trade-off between computational time and mean MSE (Mean Squared Error) is the best.

The biggest issue of this algorithm, as mentioned, is due to the high computational cost needed to build the HoG. This characteristic invalidates the possibility to use this approach in a real-time system as a racing car is.

4.5. Fast Fourier Transform method

Due to the high computational cost of the HoG approach, another method is tested in order to retrieve the direction of the velocity vector. Basing the analysis on the FFT of blurred frames, the concept is moved from the space domain to the frequency one analyzing magnitude and phase of the pictures [22]. Also in this case, just one sample per step is required and motion blurred features must be present in the scene.

The DFT (Discrete Fourier Transform) is obtained by decomposing a sequence of values into components of different frequencies. Due to the fact that DFT can be improved in computational time, the FFT, instead, is used in practical applications. An FFT rapidly computes such transformations by factorizing the DFT matrix into a product of sparse, mostly zero, factors. As a result, it manages to reduce the complexity of computing the DFT from $O(N^2)$ to $N \log(N)$, where N is the data size. The difference in computational speed can be very high. In the presence of round-off error, many FFT algorithms are much more accurate than evaluating the DFT definition directly or indirectly [13].

The DFT is the sampled Fourier Transform and therefore does not contain all frequencies forming an image but only a set of samples which is large enough to fully describe the spatial domain frame. The number of frequencies corresponds to the number of pixels in the spatial domain image, so the sample in the spatial and Fourier domain are of the same size. For a square image of size $N \times N$, the two-dimensional DFT is given by (4.16).

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) e^{-i2\pi(\frac{ki}{N} + \frac{lj}{N})} \tag{4.16}$$

The Fourier Transform produces a complex number valued output image which can be displayed with two matrices, either with the real and imaginary part or with magnitude and phase [20]. In image processing, often only the magnitude of the Fourier Transform is displayed as it contains most of the information of the geometric structure of the spatial domain. However, to re-transform the Fourier image into the correct spatial domain after processing, both magnitude and phase of the Fourier image must be preserved. In the following pictures it is possible to better understand what this kind of information is Fig.4.19.

The Fourier domain image has a much greater range than the image in the spatial domain. Hence, to be sufficiently accurate, its values are usually calculated and stored in float values.

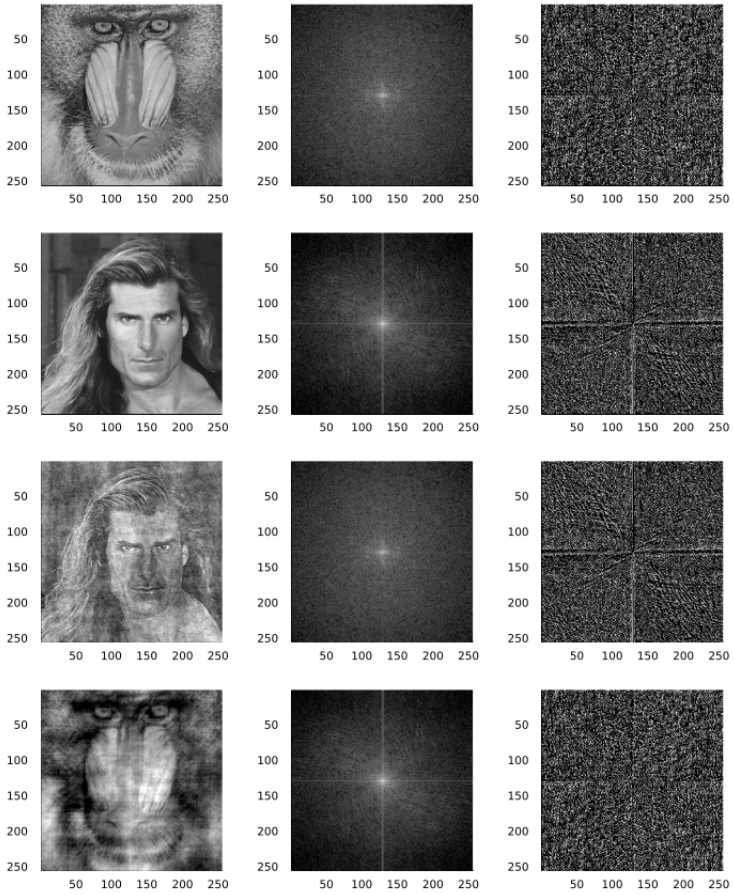

Figure 4.19: Magnitude and phase after Fourier transformation of example images.

FFT is usually used for computer vision analysis due to its low computational costs: converting an image to FFT domain, processing the frame and converting it back to space domain is in general faster than making the same application in the spatial domain directly. In fact, one of the most important properties of the Fourier domain is the property to compute convolutions as simple multiplications [2]. Let g, s, $h \in L^1(\mathbb{R})$, in space domain, be functions that satisfy $\hat{g}$, $\hat{s}$, $\hat{h}$ $\in L^1(\mathbb{R})$ in Fourier domain. Then the convolution $g(x)$ = $(s * h)(x)$ in spatial space corresponds to a multiplication in Fourier space (4.17):

$$\hat{g}(\epsilon) = \hat{s}(\epsilon)\hat{h}(\epsilon) \tag{4.17}$$

This property can be applied to almost every filter in computer vision techniques which is convolved with the image. Unfortunately, from the moment most used software functions are developed in the spatial domain, these algorithms must be usually implemented manually as in the GVS case.

From what mentioned before, also rotation information is stored in the frequency domain, in particular in the amplitude spectrum of the FFT. Usually, these features are not desirable because having an algorithm invariant with respect to rotation can also manage frames acquired at different angles. Anyway, in the GVS case, this information is what makes the system work and so it is maintained and enchanted.

In general, for an image rotation with transformation matrix R and its inverse $R^{-1} = R^T$ it is possible to calculate the Fourier transform of $f(Rr)$ as following (4.18), (4.19), (4.20) and (4.21). The determinant of the Jacobian is assumed to be 1.

$$(\mathbb{F}f(Rr))(k) = \int_{\mathbb{R}^2} f(Rr)e^{i2\pi k^T r}dr \text{ substitute } \tilde{r} = Rr \tag{4.18}$$

$$(\mathbb{F}f(Rr))(k) = \int_{\mathbb{R}^2} f(r)e^{i2\pi k^T R^T \tilde{r}}d\tilde{r} \tag{4.19}$$

$$(\mathbb{F}f(Rr))(k) = \int_{\mathbb{R}^2} f(r)e^{i2\pi (Rk)^T \tilde{r}}d\tilde{r} \tag{4.20}$$

$$(\mathbb{F}f(Rr))(k) = (\mathbb{F}f(r))(Rk) \tag{4.21}$$

Consequently, a rotation in image space corresponds to a rotation in the Fourier space as it is illustrated in the following picture Fig.4.20.

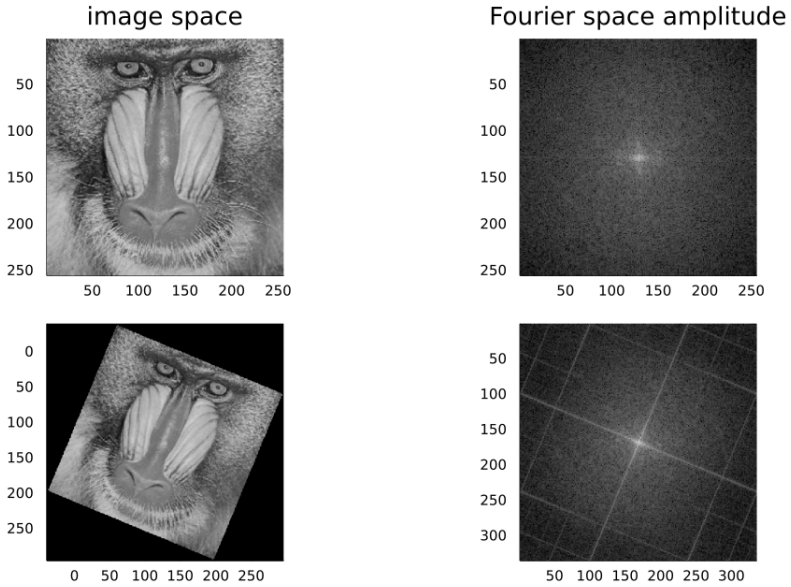

Figure 4.20: Fourier space amplitude of an image rotated in space domain.

The rotation of the image produces features that can be detected in the Fourier amplitude domain. The explanation to this fenomena is that the rotation invariance only holds in the continuous setting. The edges appear due to the sharp change of intensity present in the rotated image shown on the left. These artifacts can be mitigated by cropping the rotated image, anyway, the rotation invariance does not hold in the discrete setting. Moreover, in the GVS case, these artifacts are present due to evident edges in a main direction spread all other the frame. From these considerations, the concept of using the FFT for the detection of the direction of velocity vector finds a reason to be implemented. From what stated it is also true that, if the amount of blur increases, this property is stronger, resulting in a better overall analysis. This is the perfect situation to develop a high speed orientation detector.

In the software implementation, first of all, the frame is acquired and its FFT is computed. In this case just the magnitude information is used due to the fact it is not necessary to use the IFFT (Inverse Fast Fourier Transform). In the following image Fig.4.21, the car is moving straight, in fact, the motion blur effect is present and the features are stretched along the vertical direction. The main information in the FFT is centered and horizontal. This is due to the fact the FFT is describing the edges direction of each pixel.

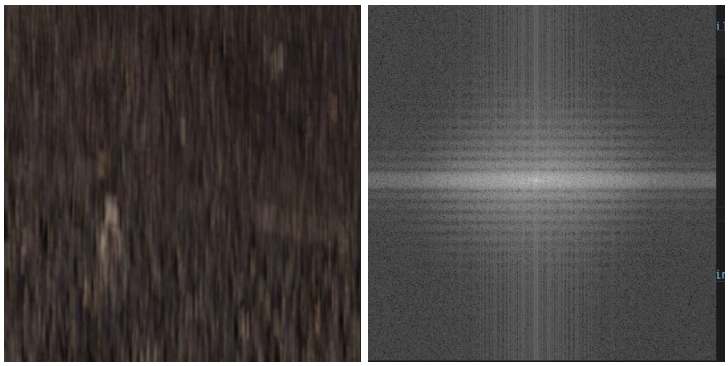

Figure 4.21: Original blurred frame (left) and its FFT magnitude (right).

Ideal high pass filter is applied to enchant the high frequencies strongly related to the gradient direction. This filter is defined as follow (4.22) with $\hat{h}_{LP}$ described by (4.23) and α the parameter used to adjust the amount of the sharpness that is added to the original one.

$$\hat{h}_{HP}(u,v) = 1 + \alpha(1 - \hat{h}_{LP}(u,v)) \tag{4.22}$$

In the following pictures it is possible to understand the relation between the mathematical model in frequency space Fig.4.22 and the result in the GVS case Fig.4.23 with a cut off frequency of 0.5 Hz.

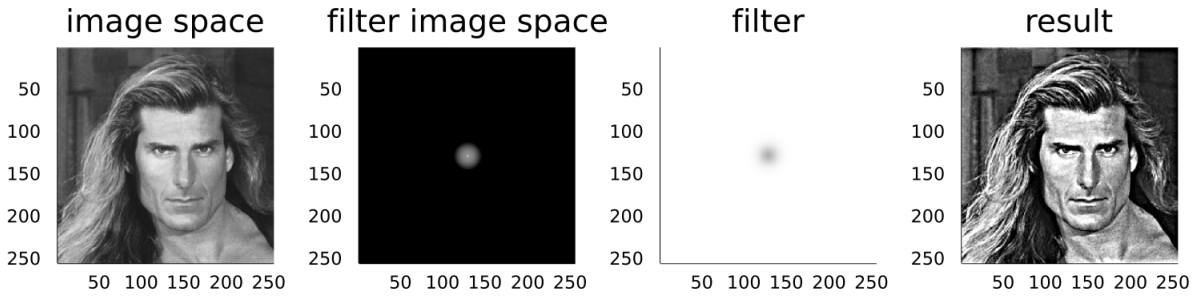

Figure 4.22: Frequency domain application of high pass filter.

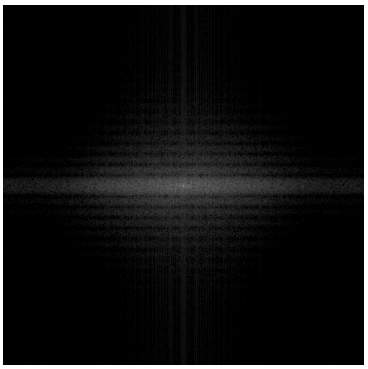

Figure 4.23: High pass filtered frame in the GVS application.

Also an ideal low pass filter is used to smooth the image and so also the outliers that can be present in the scene (4.23).

$$\hat{h}_{LP}(u,v) = \begin{cases} 1 \text{ if } ||(u,v)^T||_\infty < D \\ 0 \text{ otherwise} \end{cases} \tag{4.23}$$

In the following pictures, the relation between mathematical model, frequency space images Fig.4.24 and the result in the GVS case Fig.4.25, with a cut off frequency of 15 Hz is shown;

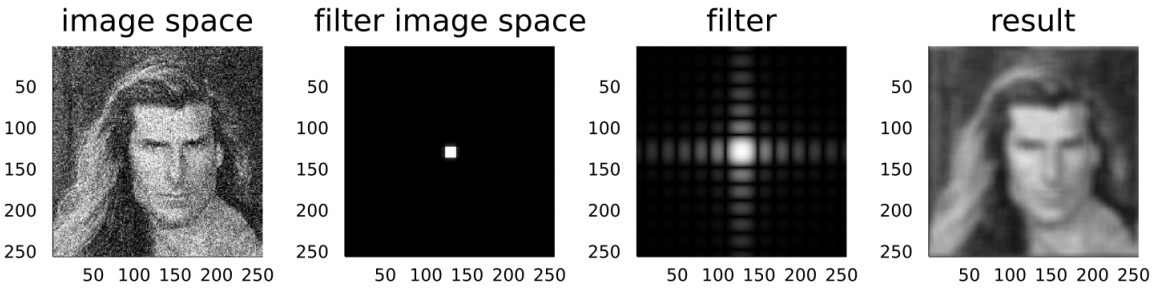

Figure 4.24: Frequency domain application of high pass filter.

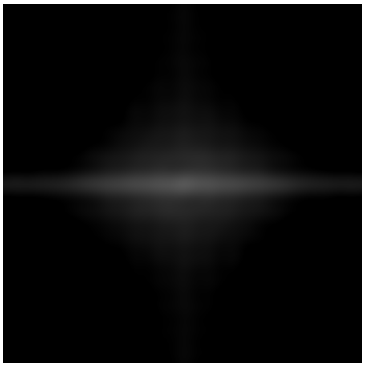

Figure 4.25: Low pass filtered frame in the GVS application.

High pass and low pass filters are used to highlight just the features of interest [10], [29]. In general, the meaningful information is the one around the center of the frame from the moment the motion blur is equally distributed in the sample. In a more practical view, the previous filters are used to reduce the image information to just one line passing through the center of the FFT. This is not fundamental but it speeds up the overall computational time from the moment a lot of pixels are near 0 intensity.

At this point, the image is ready to analyze the information about the direction. Around the center of the frame, for each column, its pixel with maximum intensity is detected and its position is stored in a vector. Connecting the center of the image with the point found, the segment containing maximum FFT intensities is defined Fig.4.26. This line, remembering what stated previously in the FFT theory, is the one related to the orientation transformation matrix and so to the perpendicular to the frame blur direction, considering the $2D$ reference frame with x pointing to the right and y up.

The interpolation of maximum intensities positions is not used because it will increase the computational time a lot more than accuracy.

The next picture Fig.4.27 shows the results of a sine test performed using the FFT method.

The most evident fact is the proof of what the initial concept states. Due to the fact the blur features are not present in the low speed range, it is not possible to retrieve the direction of movement. Moreover, the plot highlights the fact that the possible values are discretized. In

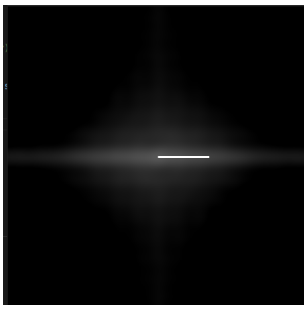

Figure 4.26: Maximum intensity line related to velocity vector direction.

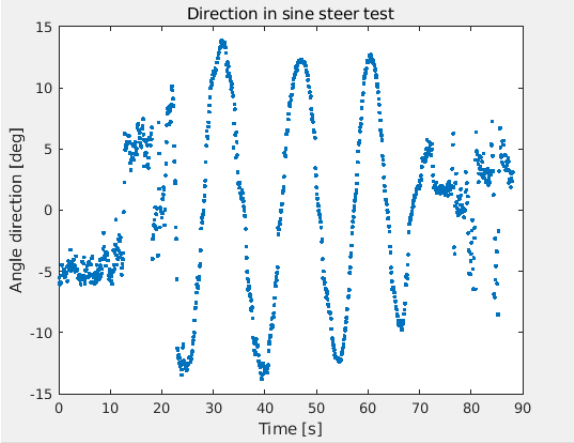

Figure 4.27: FFT analysis result performed on a sine steer test. The point of analysis is distant 75 px from the center.

fact, the slope of the line related to the orientation is based on FFT points. These elements are intrinsically defined by the number of pixels of the sample.

Anyway, it is possible to consider using points at different distances from the center in order to have smaller discretization steps on the final angle. Increasing the distance, means also increasing the area acquired by the frame. Moreover, these positions store low information, increasing the possibility of errors from the moment the relevant data are the ones around the frame center.

As reported in the plot Fig.4.28, increasing the angle direction also increases the error. Considering the working range limit in the case in which the car's absolute direction is 20 degrees, a reasonable value for a racing car, the maximum error is under ± 1 degree in the worst case. This behavior is due to the computation of the angle which is based on pixel positions. The algorithm is very precise when the direction is perpendicular to y axes and decreases going to the 90 degrees configuration. To avoid this problem it is possible to compute the point of interest basing the analysis also on rows and composing the two results found along the x and y axis. Anyway, due to the fact the direction of movement of racing cars is pretty small around the vehicle x axes, in the final GVS prototype application, introducing also the analysis along the other direction is not improving the result considerably. The choice is to use just the approach described to not double the computational time.

Finally, the performances of the FFT algorithm are very good in terms of accuracy, precision and computational cost. This statement is true only in the working range defined at the beginning of this thesis. Also the FFT algorithm begins to lose precision at very high speed, not considered in the working range of GVS systems, due to image saturation.

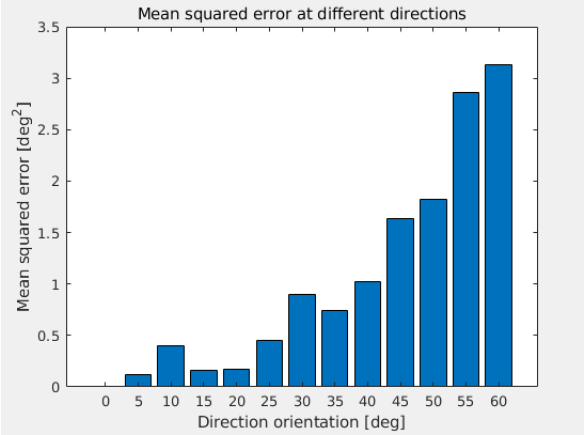

Figure 4.28: Comparison of mean MSE at different directions.

5

Velocity speed analysis

The goal of the GVS application is focused on velocity vector direction measure. Also the concept behind the speed analysis is taken into account.

This section introduces the future improvement of the GVS system describing the possible approaches to retrieve velocity vector magnitude. The major issue is to define which features of the frame are related to vehicle speed. The static approaches implemented fully contain the information about this quantity but, at higher speed, the HoG and FFT methods developed are not directly related to speed.

The manual frame difference and the ORB descriptor approaches work, also in the speed case, just up to a certain threshold. Over a this velocity magnitude value, the frame contains too blurred features deteriorating the overall analysis. For this reason, other methods must be introduced for speed analysis at high speed.

It is possible to split the concepts in static and dynamic exposure time based algorithms.

The approaches implementation can start from the ones used for the direction analysis but they must be updated and upgraded to retrieve a different kind of information.

The described algorithms are also partially implemented in C++ programming language to retrieve basic knowledge of their possibilities but they are not at the release stage. In fact, this is not the goal of the project for the first application prototype. Moreover, they are necessary to generate the first plots to understand the behavior of the model ruling the speed analysis.

In the following paragraphs the pros and cons of each possible method of implementation are described.

5.1. Frame difference and ORB descriptor

The manual and the ORB descriptor frame difference algorithms, as defined in the direction analysis chapter, can work also for the speed analysis. The previous description focused on the angle defined by the displacement vectors of each pair of pixels in consecutive frames. Their magnitude, instead, contains the information about the rate of movement of the vehicle in a certain period. If the exposition time remains the same, the more the frames are shifted between each other the faster the car is. Fixing this period as described in the FD section, at 3 ms, it is possible to relate the magnitude of the displacement $||x(t)||$ to the vehicle speed (5.1) in m/s.

$$Speed = \frac{Displacement}{Time} = \frac{||x(t)||}{ExposureTime} = \frac{||x(t)||}{0.003} = ||v(t)|| \tag{5.1}$$

Moreover, this information related to the angle calculated by the GVS system, retrieved by the direction analysis, can be used to define the two components of $||v(t)||$: $||v_x(t)||$ in x direction and $||v_y(t)||$ in y direction.

As in the direction case, the biggest issue is related to the amount of blur contained in the images. Must be remembered that these algorithms need a history of one sample frame to work.

Switching to the implementation of the FD and ORB descriptor speed analysis, the magnitude is retrieved after the necessary pre-processing and features enchantment explained previously.

Briefly summarizing the steps needed to perform manual FD algorithm, first of all the two images are acquired and transformed from RGB to HSV domain. Here, thresholds are applied in order to retrieve two binary images. Erosion and dilation structural elements are convolved with the images to perform the closing of the frames. Finally, the key points positions are computed thanks to the Canny edge detector.

In the ORB descriptor case, instead, the center points are computed using a locator and a descriptor weighting and detecting the features of interest.

The ORB and manual FD methods differ in the way key points are retrieved but, at the end, they measure the same kind of output information which can be used for direction and speed measurement.

At this point the only upgrade necessary is to calculate, along with the angles, also the displacement vectors magnitude. The overall amplitude is filtered averaging the speed described by each pair of pixels in consecutive frames (5.2) with N the number of displacement vectors computed by the GVS.

$$|x(t)| = \sum_{n=1}^{N} \frac{||x_n(t)||}{N} \tag{5.2}$$

In the next picture it is possible to see how the speed ORB descriptor algorithm performs Fig.5.1.

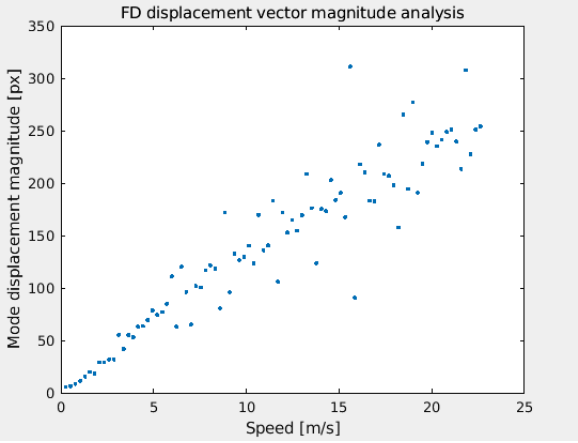

Figure 5.1: Frame difference speed analysis results at 45 degree constant direction.

As for the direction case, the loss of precision is present when the blur in the frame becomes more evident. As mentioned, at least another approach is needed to cover the high speed range.

5.2. Histogram of Gradients

During the implementation of the HoG method for the analysis of the direction of the velocity vector, also the speed is studied.

From the moment the HoG is an histogram, it is possible to calculate its mean and standard deviation. In particular, if the speed is higher, and so the amount of blur, also the Gaussian distribution changes. Increasing the speed, means measuring more blur in the image and so, as seen in the direction application, a more sharp distribution. This aspect is reflected in the standard deviation and the precision of the HoG algorithm. Measuring the performance of the implementation and its precision in particular, it is possible to retrieve the information about the velocity vector magnitude of the vehicle.

In the plot Fig.5.2, the expected behavior matches suitable results only in the middle of the operative range. At low speed the model is not accurate at all due to the low amount of blur. In the very high speed range, instead, the blur saturates the image and increasing the motion blur does not change the analyzed frame characteristics related to the standard deviation.

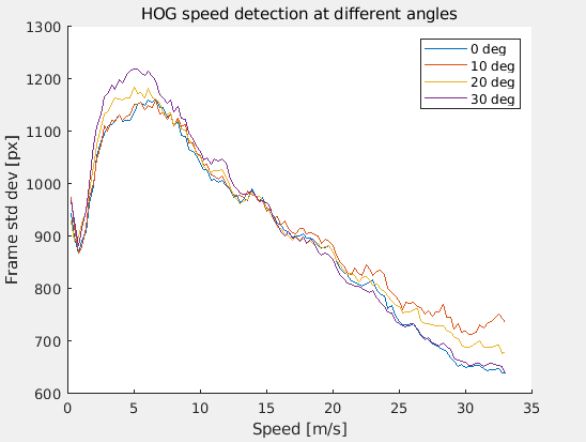

Figure 5.2: HoG speed analysis at different constant directions.

From the moment the next approach can cover all the operative range, this concept is not further investigated but it is, in principle, possible to use the ORB descriptor and the HoG approaches together as a complete solution to the magnitude velocity measure. Another issue comes from the fact that the HoG method is based on histogram and so the considerations about computational time are the same as the one of the direction case. The system cannot match the real-time requirement and a faster method can be used to fix this condition.

5.3. Laplacian

Due to the issues related to the previously described approaches, it is necessary, also for the speed analysis, to find another more suitable method. It turns out that this new algorithm can cover all the speed working range giving the possibility of using the speed computed by the GVS itself as reference for the switching threshold of the direction analysis. In this first prototype, the speed, which is possible to be measured from sensors already mounted on the electric vehicle, can be provided by devices external to the GVS.

Starting from the concept of gradient, it is possible to understand which is the most interesting information to detect in the speed analysis.

The first derivative information of each pixel, used in the HoG method, can just be related to the direction of the features, or the edges of an image, identifying the slope of the changes of

intensities in pixel values. To shift the concept of derivative to the amount of blur, not just to its direction, the concept is to use the Laplacian filter [14], [3].

The Laplacian filter is based on the second derivative of the image and detects, not how much, but how fast the gradient is changing. In practice, the filter generates a matrix collecting the information about the fastness of changing of the first derivative. The Laplacian is defined as following (5.3):

$$\Delta f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y} \tag{5.3}$$

In the GVS application, the implementation is performed in the space domain but, as future improvement, to speed up the computational time, it is possible to apply the Laplacian filter in the Fourier domain. In this case the equation holding is the following (5.4):

$$\hat{h}_{Laplacian}(u, v) = -4\pi^2(u^2 + v^2) \tag{5.4}$$

The next pictures are an example of this kind of computation in the Fourier domain Fig.5.3. This filter can also be used as a sharpening image filter.

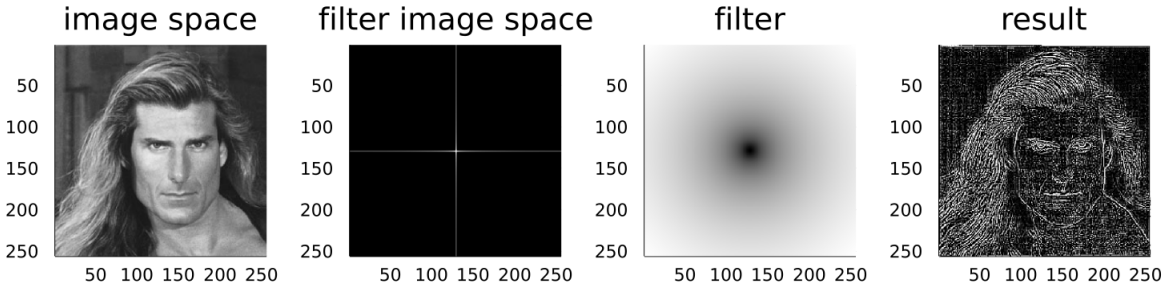

Figure 5.3: Example of Laplacian filter use in the Fourier domain.

An important property of the Laplacian is that it is invariant under all Euclidean transformations: rotations and translations. In two dimensions this property is described as follows (5.5):

$$\begin{aligned} &\Delta(f(x\cos\theta - y\sin\theta + a, x\cos\theta + y\sin\theta + b)) = \\ &= (\Delta f)(x\cos\theta - y\sin\theta + a, x\cos\theta + y\sin\theta + b) \end{aligned} \tag{5.5}$$

The invariance of the Laplacian implies that the speed analysis is not affected by frame rotation. This approach gives more robustness to the overall system and separates the velocity vector direction implementation from the one used to retrieve the speed of the vehicle.

Thanks to LP, it is possible to calculate the mean and standard deviation of the image pixel intensities representing how fast the pixel values are changing in the blurred features. From the moment the distribution is describing the rate of change of each pixel value, the more the speed is higher and the more the intensities, due to blur, go to the average value of the overall frame.

This statement also suggests that, at high speed, the values are all closer to the mean value and the standard deviation is smaller. On the other hand, if the blur is not enough, the pixel intensities will cover all the color scale range and the standard deviation will be higher. In the space domain this concept is related to the amount of the gray level in the image.

The more the frame is blurred and the more the frame is uniform and the pixel values mixed between themselves so the standard deviation is higher. At the end, it is possible to say that the standard deviation is related to the amount of blur of the image through the second derivative of

the frame pixel values.

In order to perform the analysis, first of all, erosion and dilation, or closing, on the original frame is performed to enchant features of interest Fig.5.4.

Figure 5.4: On the left, Laplace speed analysis original frame. On the right the blurred frame after morphology transformation.

At this point, the Laplacian filter is applied in the space domain Fig.5.5 to retrieve the second derivative information.

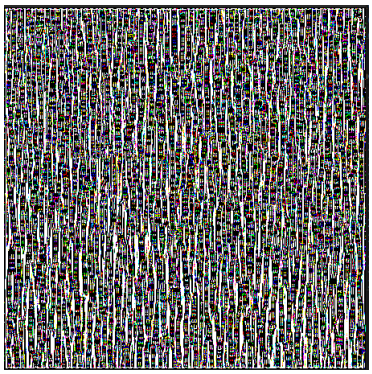

Figure 5.5: Frame after Laplacian filter is applied.

Once the matrix is computed, the standard deviation of the frame is calculated and, thanks to the inverse model, the speed can be retrieved.

Plotting the standard deviation Fig.5.6 of a linear acceleration test it is possible to see that the curve becomes almost flat and is not possible to discriminate the speed information at very high speed.

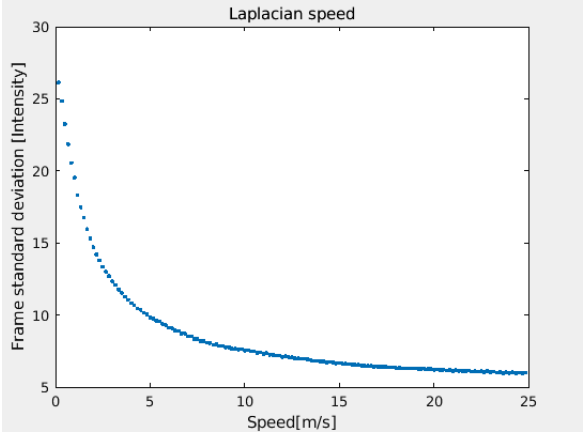

Figure 5.6: Laplacian speed analysis.

Differently to the case of the direction analysis, where the orientation is directly proportional to the real angle, here, a more complex model is necessary. Thanks to Matlab analysis, it is possible to fit a model to be inverted, splitting the curve into exponential sub-models Fig.5.7. The choice of the exponential form is due to the fact it is simpler to be inverted than a high grade polynomial. The switching between the sub-models can be handled internally in the software after the computation of the standard deviation value.

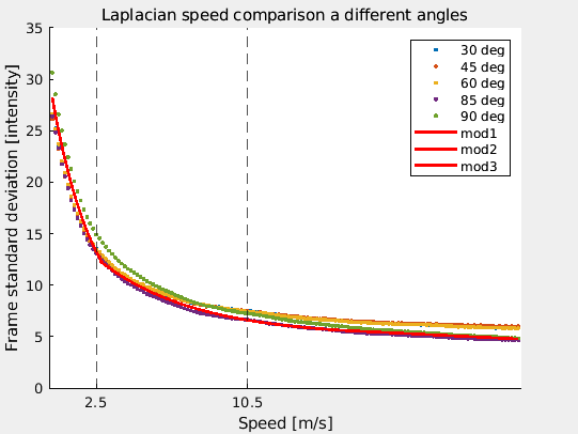

Figure 5.7: Laplacian speed models generation.

What is important to define is how the software shifts between one sub-model and the next or the previous one. The switching points are defined as the limits of the inverted functions, starting from the boundaries of the direct sub-models. In this way, when the application standard deviation output is greater or smaller than a certain threshold, the algorithm should change the function to use for the speed measurement. This is an approach which finds its implementation offline, while the final application can use the inverted final model in a real-time environment.

The Laplacian method can have good performances in the speed analysis but should be tested more in the development phase. Anyway, the results are good and the curve behavior reflects what reality and expectations are. The plot Fig.5.7 shows that the amount of blur is not linearly distributed with respect to the speed.

Moreover, no calibration is needed due to the fact standard deviation, and not mean, is used. The standard deviation is in fact measuring the spreadness of data, related to speed and blur, but not to intensities in absolute values. For these reasons, changing asphalt mixture does not change the speed analysis result.

Using, instead, a dynamic shutter opening time, can move the complexity of the algorithm to the development of an estimator to change the exposure period [8]. In this case, the approach can result in an overall simpler and single model. Once the speed is going too near to the boundary of the model, the exposure time can be increased or decreased to give the possibility to use the same function, just shifted, in all the speed range.

Due to the fact, at the moment, static exposure time is used in the direction analysis, the dynamic method is not further investigated for the first prototype but must be remembered that it can be a reasonable solution to the speed analysis.

6

Overall GVS system

The previous chapters treated all the components of the GVS system separately. Not all of them are used in the final overall application to retrieve the velocity vector direction and magnitude.

In the low speed range, the FD and the ORB algorithms are based on the same concept. Magnitude and direction of the velocity vector are related to the displacement vector between two consecutive frames. Moreover, the two methods can be used to retrieve the same information in the same speed range. Looking at the output results obtained, the ORB descriptor frame difference approach is kept for the analysis of the vehicle direction at low speed.

At high speed, as mentioned, the HoG analysis is discarded due to the high computational time. Even if the output quality is good, the real-time requirement can not be matched. For this reason the FFT algorithm is adopted in the final implementation.

As explained, the ORB and FFT methods are used to retrieve the direction of the velocity vector in the relative operative working ranges. From the moment using a static exposure time it is not possible to detect blur in the image, the gradient is not present at low speed, while the effect generates noise at very high speed. For this reason it is important to remember to use the GVS system in the defined speed working range.

To understand better how the GVS system works in the real overall application, the following procedural graph can be helpful Fig.6.1. First of all, the frame must be acquired by the camera sensor. As default parameters, the ones defined in the section related to the camera model are used and suggested in case the system is replicated.

The sample is then stored for the next step in case the frame difference needs to be performed. To satisfy the memory requirement of the algorithm, the first frame at 0 m/s speed is acquired as reference for the following steps.

The speed analysis should be performed or another method to input the magnitude quantity should be used. Speed is computed before direction because it is also used as a threshold parameter to switch between ORB and FFT approaches. Moreover, Laplacian is rotation invariant so the magnitude of the velocity vector can be calculated before direction.

In particular, in the case of the Laplacian analysis, there is a very small range of uncertainty near 0 m/s speed. Anyway, it is possible to assume that, from the moment this situation is present just when the frame does not present blur at all, the velocity is describing the real 0 m/s condition. Thanks to this assumption just one method can be used, the Laplacian one, for the speed quantity measure.

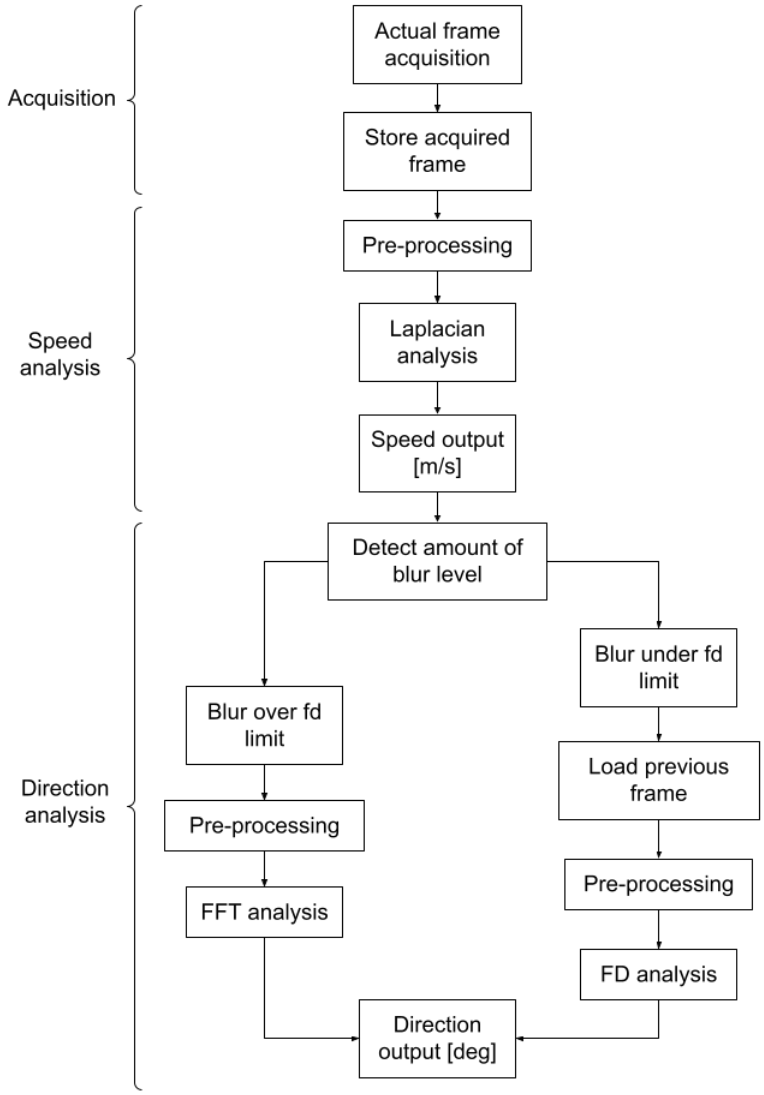

Figure 6.1: Procedure performed by GVS system at each step of computation.

In the Laplacian filter, the frame is processed in order to enchant the features of interest and to smooth the noise effects. Here, the goal is to retrieve the standard deviation of the image after the filter is applied. This information is related to how long the blur features are and so also to the speed. The standard deviation in pixels must be converted to velocity vector amplitude through the inverted models.

After the speed analysis is performed, the direction measurement can start with the choice of which of the two possible algorithms has to be used. If the speed is high enough to detect the blur in the sample, the FFT method is used, otherwise, the frame difference one is adopted. In the frame difference case it is necessary to load the previous frame from the moment the method is based on features displacement vectors. After pre-processing, the ORB descriptor

method can be used.

In the case the blur is enough to perform the FFT analysis, instead, it is possible to take just a single frame, converted in frequency domain, into account. First of all, also here pre-processing is necessary. Then, the FFT analysis returns the direction vector basing the analysis on the spectrum magnitude.

In both the previously mentioned cases the output direction is given in degree units.

From the moment the application must run in a real-time application, the driverless car of E-agle Trento Racing Team, it is important to measure also the computational time required by each algorithm. The acquisition time, as defined previously, is set at 3 ms. Moreover, the following list Tab.6 contains all the periods of time between the first instruction of each method and the moment in which the output of the algorithm is available.

FD computational time	8.020 ms
ORB computational time	6.210 ms
HoG computational time	351.375 ms
FFT computational time	5.383 ms
Laplacian computational time	5.408 ms

As it is possible to see from the list, the computational time required by the HoG is very high and it is not compatible with the real-time requirement. In fact, if the sample rate of the camera is 60 FpS, the time between each frame is (6.1):

$$AcquisitionRate = \frac{1000}{60} = 16.667ms \tag{6.1}$$

This means that the overall computational time should be lower than this period in order to not lose the frames acquired or to decrease the acquisition rate. There are different contributions to overall application (6.2) from the moment Laplacian, ORB and FFT methods are used consequently. In the worst case, using the ORB frame difference approach:

$$\begin{aligned} OverallTime = acquisition + Laplacian + ORB = \\ = 3.000ms + 5.408ms + 6.210ms = 14.618ms \end{aligned} \tag{6.2}$$

The requirement about the computational time is met but it should be safer to decrease the sample rate to 50 FpS. At this point of development of the GVS application, the 60 FpS sample rate is long enough to perform the analysis in real-time but improving the image quality, and so needing less pre-processing, the overall system efficiency can be improved.

The computational times defined in this chapter are collected using a Lenovo Thinkpad T440p [16] running Ubuntu OS. The CPU (Central Processing Unit) is an Intel® Core™ i5-4330M (FSB up to 3.50 GHz, 3 MB, L3, 1600 MHz). The storage used is a Samsung 512 GB SSD (Solid State Drive). The RAM (Random Access Memory) is 16 GB max7 / PC3-12800 1600 MHz DDR3L composed of two 8 GB slots. Nowadays this can be considered a low cost and low computational power setup. Moreover, the hardware already mounted in the electric autonomous racing vehicle of E-agle Trento Racing Team is more powerful and also includes specific and dedicated hardware for graphical computations.

7

Conclusion

The goal of this thesis is to design and implement a new sensor for the acquisition of the velocity vector direction.

Starting from the requirements definition, different approaches are defined in order to find the best solution for the first prototype of the GVS system. The fundamental aspects to be investigated are the accuracy, precision and computational time. Overall external complexity should be low to make the final user interface as simple as possible.

Analyzing the low speed range, two frame difference methods are described. Thanks to the knowledge achieved implementing the manual FD method, it is possible to describe two main limitations:

- The system lose accuracy and precision at high speed due to the motion blur effect;
- Pre-processing is necessary to prepare the two frames for the analysis increasing the computational time.

Even if the first point is related to an intrinsic characteristic of the system acquisition, the second one can be improved by using the ORB descriptor. In this case the key points feature can be computed without pre-processing decreasing the overall computational time by ≈ 3 ms.

In the ORB descriptor frame difference analysis case the results are accurate and precise but these properties are lost at higher speed Fig.7.1.

To cover the entire working speed range defined in the requirements section, from 0 to 130 Km/h, it is necessary to define a new method based on motion blur analysis. In fact, setting the exposure time at 3 ms, the features of interest related to the velocity vector are stretched and they can be measured in direction and magnitude.

The first approach adopted uses the HoG algorithm. In this case, the performance can match the desired characteristic just in case the bins size decreases enough. This quantity is directly related to the accuracy of the GVS. Unfortunately, optimizing this attribute increases considerably the computational time, moving the system out from the real-time working condition.

In order to fix this issue, the HoG concept is converted to the FFT one. In The Fourier Domain efficiency increases and the filtered frame processing can produce an output matching the desired accuracy and precision Fig.7.2.

Finally, using the frame difference, based on ORB detector, and the FFT methods together, it is possible to implement a ground velocity sensor working on all the speed range defined.

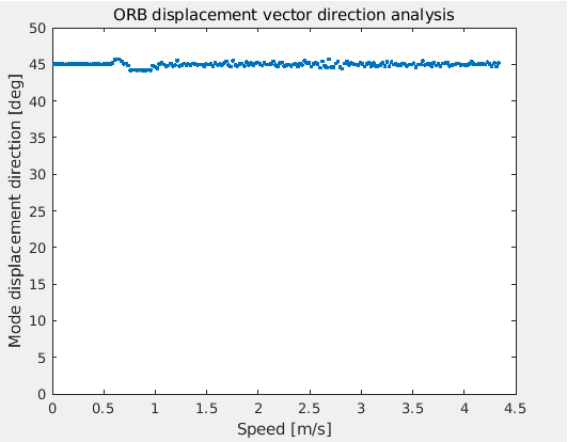

Figure 7.1: ORB descriptor frame difference analysis in constant 45 deg direction test.

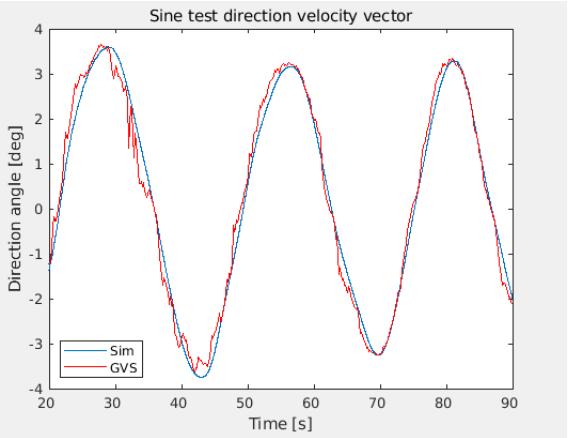

Figure 7.2: FFT analysis in sine test.

From the moment two models are used for low and high speed, the magnitude of the velocity vector must be used to switch between the two approaches. The speed measure is not fundamental in this project from the moment different kinds of sensors are available to be used as input to the GVS direction pipeline.

In this thesis, the speed analysis is just introduced and left as an improvement in a future thesis that completes the GVS system. The Laplacian method should be investigated in more detail. Thanks to the invariance property, this approach can be used before the direction analysis and independently from it. Adopting this concept also increases the overall internal complexity of the system from the moment it requires the definition of sub-models which transform the standard deviation of the frame into the speed quantity in m/s units.

Due to the issues described in the simulator data acquisition section, the tests are restricted and should be further investigated to describe the overall complete characteristics of the GVS sensor. The Monte Carlo analysis and tests at different speeds and conditions should be performed to achieve this knowledge.

This thesis aims at implementing the necessary algorithms to let the GVS system working in the E-agle Trento Racing Team electric and autonomous vehicles. For this reason the project is focused on the implementation of the algorithm and not on the overall tests which are scheduled for the next months with a new thesis project.

Finally, in this project, it is possible to understand which are the requirements necessary to implement a working ground velocity sensor prototype. What is treated in these chapters is fully

implemented and tested in C++ code. This programming language is chosen in order to make the import on embedded systems easier and faster.

The GVS system can retrieve the velocity vector direction, which nowadays requires several hardware and sensor fusion techniques, with just one device. The Ground Velocity Sensor finds a solution to the problem related to the high complexity of the approaches used in the vehicle dynamics literature. The accuracy and precision are matching the constraints set by the FS championship vehicles. Finally, the outputs of the GVS user are the direction and magnitude of the vehicle velocity vector centered in the car CoM.

References

[1] Borrowlenses. *Shutter Speed Chart and Tips on How to Master It*. URL: https://www.borrowlenses.com/blog/shutter-speed-chart/ (visited on 12/06/2016).

[2] Paul Bourke. “Image Filtering in the Frequency Domain”. In: (1998).

[3] Daniel Leventhal Adapted from Brian Curless. “Image processing”. In: *CSE 457* (2011).

[4] Nicola Conci. “Camera Geometry and Multi-View”. In: *CV and MM course* (2019).

[5] Nicola Conci. “Images and Videos”. In: *CV and MM course* (2019).

[6] Nicola Conci. “Local Feature Extraction”. In: *CV and MM course* (2019).

[7] Nicola Conci. “Models”. In: *CV and MM course* (2019).

[8] Felix Heide Emmanuel Onzon Fahim Mannan. “Neural Auto-Exposure for High-Dynamic Range Object Detection”. In: *Princeton University, Algolux* (2021).

[9] “End-to-End Velocity Estimation For Autonomous Racing”. In: *IEEE robotics and automation letters* (2020).

[10] Francesca Pizzorni Ferrarese. “Image Enhancement: filtering in the frequency domain”. In: ().

[11] “Formula Student Rules 2022”. In: *FSG* (2021).

[12] Gary Bishop Greg Welch. “An Introduction to the Kalman Filter”. In: *Department of Computer Science University of North Carolina at Chapel Hill* (2007).

[13] *Introduction to Fourier transform image processing*. URL: https://www.cs.unm.edu/~brayer/vision/fourier.html (visited on 10/09/2021).

[14] Ramesh Jain, Rangachar Kasturi, and Brian Schunck. *Machine Vision*. Jan. 1995. ISBN: 978-0-07-032018-5.

[15] Attila Kun. *Exposure ISO, Aperture, and Shutter Speed Explained*. URL: https://www.exposureguide.com/author/attila-kun/ (visited on 09/13/2021).

[16] *Lenovo Thinkpad T440p specifications*. URL: https://support.lenovo.com/sg/it/solutions/pd100280-detailed-specifications-thinkpad-t440p (visited on 2022).

[17] Sergio Rapuano Luigi Coco. “Accurate Speed Measurement Methodologies for Formula One Cars”. In: *Instrumentation and Measurement Technology Conference - IMTC* (2007).

[18] *MatLab Documentation*. URL: https://it.mathworks.com/help/matlab/ (visited on 2021).

[19] Soohyun Kim Minyoung Lee Kyung-Soo Kim. “Measuring Vehicle Velocity in Real Time Using Modulated Motion Blur of Camera Image Data”. In: *IEEE transaction on vehicular technology* (2017).

[20] M. Strzelecki P. Strumiłło. “Fourier transform of images”. In: (2021-09-11).

[21] Konrad Scheffler M. Sc. Prof. Dr. Ing. Tobias Knopp. “Image Processing - Color Image Processing”. In: *TUHH CV course* (2021).

[22] Konrad Scheffler M. Sc. Prof. Dr. Ing. Tobias Knopp. “Image Processing - Filtering in Frequency Domain”. In: *TUHH CV course* (2021).

[23] Konrad Scheffler M. Sc. Prof. Dr. Ing. Tobias Knopp. “Image processing - Image fundamentals”. In: *TUHH CV course* (2021).

[24] R. Reddy, Chiluka Nagaraju, and I Raja Sekhar Reddy. “Canny Scale Edge Detection”. In: *IJETT* (Jan. 2016). DOI: 10.14445/22315381/IJETT-ICGTETM-N3/ICGTETM-P121.

[25] Ethan Rublee et al. “ORB: an efficient alternative to SIFT or SURF”. In: *Proceedings of the IEEE International Conference on Computer Vision* (Nov. 2011). DOI: 10.1109/ICCV.2011.6126544.

[26] Jai Puneet Singh and Simarpreet Singh. “Motion Blur: An Introduction”. In: *2347-6710* 2 (Jan. 2013), p. 293.

[27] *Types of ADAS Sensors in Use Today*. URL: https://dewesoft.com/daq/types-of-adas-sensors (visited on 08/30/2021).

[28] *UnrealEngine Manual*. URL: https://docs.unrealengine.com/5.0/en-US/ (visited on 2021).

[29] Raquel Urtasun. “Computer Vision - Filtering”. In: *TTI Chicago* (2013).

A

GVS Source Code

Following the main sections of code implemented in the GVS C++ code application are presented. For more details and the complete project refer to this GitHub repository:
https://github.com/MatteoSpadetto/race_dv_velocity.

```cpp
using namespace cv;
using namespace std;
using namespace cv::xfeatures2d;

int main(int argc, char const *argv[])
{
    cout << std::setprecision(3) << std::fixed; ///< Set output at 3 dec
    float theta_main;                           ///< Velocity vector direction output
    float speed_main;                           ///< Velocity vector magnitude output

    ofstream file_csv;
    file_csv.open("../gvs/data_csv/data_out.csv"); ///< Storing output in csv

    for (int frame_id = START_FRAME; frame_id <= END_FRAME; frame_id++) ///< For each frame collected
    {
        String path_a = "../../test_sine_3_10_60fps_matlab/frame_" + to_string(frame_id) + ".png";     ///< Compose frame path A
        String path_b = "../../test_sine_3_10_60fps_matlab/frame_" + to_string(frame_id - 1) + ".png"; ///< Compose frame path B

        Mat frame_a = imread(path_a, IMREAD_COLOR); ///< Load frame A
        Mat frame_b = imread(path_b, IMREAD_COLOR); ///< Load frame B

        speed_main = gvs_lap(frame_a); ///< Compute GVS speed with Laplacian method

        if (speed_main > SPEED_TH) ///< Thresholding speed for direction analysis
        {
            float theta_main = gvs_orb(frame_a, frame_b); ///< Compute GVS direction with ORB method
            cout << "[FD]" << endl;
        }
        else
        {
            float theta_main = gvs_fft(frame_a);  ///< Compute GVS direction with FFT method
            cout << "[FFT]" << endl;
        }
        cout << "Frame: " << frame_id << " --- Theta: " << theta_main << " --- Std_dev: " << speed_main << endl; ///< GVS output
        file_csv << frame_id - START_FRAME << "," << theta_main << "," << speed_main << endl; ///< GVS cvs output
        waitKey(20);
    }
    file_csv.close(); ///< Close csv output file
    return 0;
}
```

Figure A.1: GVS code of gvs_main.cpp.

```cpp
float gvs_fd(cv::Mat frame_a, cv::Mat frame_b)
{
    Mat img_th_a, img_bw_a, img_th_b, img_bw_b;
    cv::cvtColor(frame_a, img_bw_a, COLOR_RGB2HSV); ///< Convert frame A to HSV
    cv::cvtColor(frame_b, img_bw_b, COLOR_RGB2HSV); ///< Convert frame A to HSVi
    vector<Mat> channels_a, channels_b;
    split(img_bw_a, channels_a);
    threshold(channels_a[1], img_th_a, 0, 255, THRESH_BINARY); ///< Binarise the image channel 1
    split(img_bw_b, channels_b);
    threshold(channels_b[1], img_th_b, 0, 255, THRESH_BINARY); ///< Binarise the image channel 1

    /// Erode and dilate ///
    size_t elem_size = 1;
    Mat struct_elem = getStructuringElement(MORPH_ELLIPSE, Size(2 * elem_size + 1, 2 * elem_size + 1), Point(elem_size, elem_size)); // Setting dilation
    dilate(img_th_a, img_th_a, struct_elem);                                                                                        // Dilate A
    erode(img_th_a, img_th_a, struct_elem);                                                                                         // Erode A
    dilate(img_th_b, img_th_b, struct_elem);                                                                                        // Dilate B
    erode(img_th_b, img_th_b, struct_elem);                                                                                         // Erode B

    /// Find contours ///
    Mat canny_output_a, canny_output_b;
    Canny(img_th_a, canny_output_a, THRESH, THRESH * 3); ///< Apply Canny to frame A
    Canny(img_th_b, canny_output_b, THRESH, THRESH * 3); ///< Apply Canny to frame B
    vector<vector<Point>> contours_a, contours_b;
    findContours(canny_output_a, contours_a, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE); ///< Find contours of A
    findContours(canny_output_b, contours_b, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE); ///< Find contours of B

    /// Geyt contour centers ///
    std::vector<cv::Point> centers_a, centers_b;
    for (int i = 0; i < contours_a.size(); i++)
    {
        if (contourArea(contours_a[i]) >= AREA) ///< Take into account features of a certain area
        {
            cv::Moments M = cv::moments(contours_a[i]);
            cv::Point center_a(M.m10 / M.m00, M.m01 / M.m00); ///< Get center
            centers_a.push_back(center_a);
        }
    }
    for (int i = 0; i < contours_b.size(); i++)
    {
        if (contourArea(contours_b[i]) >= AREA) ///< Take into account features of a certain area
        {
            cv::Moments M = cv::moments(contours_b[i]);
            cv::Point center_b(M.m10 / M.m00, M.m01 / M.m00); ///< Get center
            centers_b.push_back(center_b);
        }
    }
```

Figure A.2: GVS code of gvs_fd.cpp.

```cpp
float gvs_orb(cv::Mat frame_a, cv::Mat frame_b)
{
    Mat frame_rot, homog;
    // Convert images to grayscale
    Mat frame_a_g, frame_b_g;
    cvtColor(frame_a, frame_a_g, cv::COLOR_BGR2GRAY); ///< Convert frame A to gray scale
    cvtColor(frame_b, frame_b_g, cv::COLOR_BGR2GRAY); ///< Convert frame B to gray scale

    // Variables to store keypoints and descriptors
    std::vector<KeyPoint> keypoints_a, keypoints_b; ///< Vectors to store frames A and B keypoints
    Mat descriptors_a, descriptors_b;               ///< ORB descriptors of frames A and B

    // Detect ORB features and compute descriptors.
    Ptr<Feature2D> orb = ORB::create(MAX_FEATURES);
    orb->detectAndCompute(frame_a_g, Mat(), keypoints_a, descriptors_a); ///< Populate ORB A
    orb->detectAndCompute(frame_b_g, Mat(), keypoints_b, descriptors_b); ///< Populate ORB B

    // Match features
    std::vector<DMatch> matches;
    Ptr<DescriptorMatcher> matcher = DescriptorMatcher::create("BruteForce-Hamming");
    matcher->match(descriptors_a, descriptors_b, matches, Mat()); ///< Matching keypoints of frames A and B

    // Sort matches by score
    std::sort(matches.begin(), matches.end()); ///< Sort matches

    // Remove not so good matches
    const int numGoodMatches = matches.size() * GOOD_MATCH_PERCENT;
    matches.erase(matches.begin() + numGoodMatches, matches.end()); ///< Erase bad matches

    // Prepare vectors for analysis
    std::vector<Point2f> points_a, points_b;
    for (size_t i = 0; i < matches.size(); i++)
    {
        points_a.push_back(keypoints_a[matches[i].queryIdx].pt); ///< Query descriptor index
        points_b.push_back(keypoints_b[matches[i].trainIdx].pt); ///< Train descriptor index
    }

    float slope_out = slope(points_a, points_b); ///< Compute slope direction
    return slope_out;
}
```

Figure A.3: GVS code of gvs_orb.cpp.

```cpp
float gvs_hog(cv::Mat frame)
{
    cvtColor(frame, frame, COLOR_RGB2HSV); ///< Convert to HSV

    GaussianBlur(frame, frame, Size(5, 5), 0); ///< Gaussian blur

    vector<Mat> channels;
    split(frame, channels); ///< Split frame in 3 channels

    cv::Ptr<cv::CLAHE> clahe = cv::createCLAHE();
    clahe->setClipLimit(4);
    clahe->apply(channels[2], channels[2]); ///< Clahe

    Mat gx, gy, mag_mat, ang_mat;
    Sobel(channels[2], gx, CV_32F, 1, 0);     ///< Apply sobel x
    Sobel(channels[2], gy, CV_32F, 0, 1);     ///< Apply sobel y
    cartToPolar(gx, gy, mag_mat, ang_mat, 1); ///< Convert to cartesian coordinates

    vector<int> bins;                                          ///< Vector describing each bin
    Point test_p = Point((ang_mat.cols / 2), (ang_mat.rows / 2)); ///< Set center point of analysis
    bins = compute_hog(ang_mat, mag_mat, test_p); ///< Compute HOG

    Scalar hog_mean, hog_std_dev;
    meanStdDev(bins, hog_mean, hog_std_dev); ///< Compute mean and std_dev of HOG

    return hog_mean[0];
}
```

Figure A.4: GVS code of gvs_hog.cpp.

```cpp
float gvs_fft(cv::Mat &frame)
{
    cv::cvtColor(frame, frame, COLOR_RGBA2GRAY); ///< Go to gray scale

    Mat magI = compute_fft(frame); ///< Compute magnitude of fft

    magI = ideal_high_pass_filter(magI, 4); ///< HP filter
    magI = ideal_low_pass_filter(magI, 5);  ///< LP filter

    cv::Mat mask = cv::Mat::zeros(magI.size(), magI.type()); ///< Use a mask to not consider outer region
    cv::circle(mask, cv::Point(mask.cols / 2, mask.rows / 2), 100, cv::Scalar(255, 255, 255), -1, 8, 0);
    cv::Mat magI_crop = cv::Mat::zeros(magI.size(), magI.type()); ///< Apply the mask
    magI_crop.convertTo(magI_crop, CV_8UC4);
    mask.convertTo(mask, CV_8UC4);
    magI.copyTo(magI_crop, mask);

    return get_orientation(magI_crop); ///< Get slope from weighted mean of FFT pixels directions

}
```

Figure A.5: GVS code of gvs_fft.cpp.

```cpp
float gvs_lap(cv::Mat frame)
{
    Mat frame_lap; ///< Laplacian out frame
    Laplacian(frame, frame_lap, CV_64F, 3); ///< Compute Laplacian of original frame

    Scalar lap_mean; ///< Laplacian frame mean
    Scalar lap_std_dev; ///< Laplacian frame std_dev
    meanStdDev(frame_lap, lap_mean, lap_std_dev, Mat()); ///< Compute Laplacian frame mean and std_dev
    return lap_std_dev[0];
}
```

Figure A.6: GVS code of gvs_lap.cpp.