CRANFIELD UNIVERSITY

NELSON FERNÁNDEZ PROVECHO

UGV Velocity Estimation from a Monocular Camera

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
Autonomous Vehicle Dynamics and Control

MSc Thesis
Academic Year: 2016 - 2017

Supervisor: Rafal Zbikowski
August 2017

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
Autonomous Vehicle Dynamics and Control

MSc Thesis

Academic Year 2016 - 2017

NELSON FERNÁNDEZ PROVECHO

UGV Velocity Estimation from a Monocular Camera

Supervisor: Rafal Zbikowski
August 2017

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Science

ABSTRACT

The work presented in this thesis focuses on the use of a forward-looking monocular camera to estimate the velocity of a ground vehicle and check the accuracy of its measurements. The proposed method processes the recorded video with a 2D-to-2D-based visual odometry approach to obtain an estimation of the camera pose of one frame with respect to the previous one using the information of salient corners on both images. However, the translation vector of this estimation is of unit length so a method to obtain the absolute scale has been also developed. Points on the road from two consecutive images are triangulated to obtain a *virtual road* plane to then measure the distance from it to the *virtual camera*. With the knowledge of the actual height of the sensor, a scale factor is computed and, together with the time between frames, the velocity is obtained. These raw measurements are noisy and also have unreal velocity peaks produced by the presence of other vehicles on the road, among other factors. To resolve this, a limit on the maximum allowed acceleration is set and a smoothing step is done. The influence of this threshold has been studied and demonstrated to be important when analysing the final results. Good estimations are achieved using the proposed methodology, showing that it is effective, giving accurate results, especially when the velocities are not too big and enough features are on the road. Additionally, it has been tested for different spatial and temporal resolutions, showing that their effect on the results is not very big, although, in general terms, better results have been obtained for the highest resolutions.

Keywords:

Visual Odometry, egomotion, essential matrix, motion estimation, RANSAC, five-point algorithm, computer vision

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

CCD	Charge-Coupled Device
CMOS	Complementary Metal-Oxide-Semiconductor
FAST	Features from Accelerated Segment Test corner detector
FHD	Full High Definition (1920x1080 pixels)
fps	Frames Per Second
GPS	Global Positioning System
HD	High Definition (960x540 pixels)
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
KLT	*Kanade-Lucas-Tomasi* tracker
RANSAC	Random Sample Consensus
RGB	Red-Green-Blue additive colour model
ROI	Region Of Interest
SD	Standard Definition (640x360 pixels)
SFM	Structure From Motion
SNR	Signal-to-Noise Ratio
UGV	Unmanned Ground Vehicle
VO	Visual Odometry

1 INTRODUCTION

1.1 Context

The work presented in this thesis has been done as a part of the MSc Autonomous Vehicle Dynamics and Control in Cranfield University and is sponsored by RACELOGIC, an English company based in Buckingham and dedicated to design, manufacture and sell systems to measure, analyse, display and simulate data from moving vehicles, especially racing ones. It also develops its own software for vehicle testing and motorsport to analyse the recorded telemetry.

As the thesis title suggests, the main aim of the project is to measure the velocity of a UGV, or any other ground vehicle (a regular car has been utilised to gather the necessary data), using a forward-looking monocular camera placed in the dashboard or, as it was the case in this work, onto the roof of the car. The idea is to check whether these sensors can give an accurate estimate of the velocity of a vehicle to be used or not in future researches as a complementary sensor in a GPS-IMU system (Inertial Navigation System - INS). INS is a mature technology that works well, but it has some limitations. GPS antennas do not always have direct line of sight with GPS satellites so their measurements may be affected by multipath signals or even no measurements are done. IMUs can fill those gaps but they have the problem that their measurements drift with time leading to inaccurate readings after a while. It is in this context where the camera system wants to be utilised in a future to bound these errors and obtain a better estimation of the actual vehicle velocity.

The data recorded by the camera is processed using visual odometry techniques, which allows for computing the relative motion of the camera and, therefore, of the vehicle, using the motion extracted from video frames. This is not the first time velocity has been tried to be measured or estimated using a monocular camera but, as it will be discussed later in section 2.2.3, it implies several challenges mainly caused by camera uncertainty sources, the lack of texture of the road and also by the fact that using a single camera does not give

depth information so measurements can only be recovered up to an unknown scale factor. In fact, an important part of this work is measuring or estimating that factor to get a proper measurement of the velocity.

Current state of the art techniques for velocity estimation with a camera are mainly based on optical flow approaches, that is, analysing the movement of every pixel of each frame (or of a part of the frame) to derive the camera velocity. However, these methodologies have given results that are noisy or that work only under controlled circumstances and slow speeds. That is because optical flow approaches greatly rely in the texture of the captured images, especially the road/ground texture. In general, roads do not have much texture and when increasing the speed of the vehicle the displacements of the pixels from one frame to the next one are bigger and bigger so the optical flow computation is more likely to fail.

In this thesis, a different approach has been followed. Instead of using optical flow techniques, the velocity has been estimated using salient features (corners) that are present in consecutive frames, so the camera pose relative to the previous one can be obtained and then scaled using a scale factor calculated using a known distance on the scene: the height of the camera.

1.2 Aims and objectives

As commented above, the main objective of this thesis is estimating the velocity of a ground vehicle using a forward-looking monocular camera. However, it is not the only objective of the present work. Below the main goals of the thesis are listed:

- Build an algorithm that allows for estimating the relative motion of the camera from one pose to another.
- Build an algorithm that allows for computing the scale factor necessary to know an estimation of the actual movement of the camera.
- Compute de vehicle velocity with the knowledge of the frame rate and the motion of the camera.

- Analyse the influence of the feature tracking thresholds in the velocity estimation results.
- Study the influence of spatial resolution in the velocity estimation results.
- Study the influence of temporal resolution in the velocity estimation results.

1.3 Thesis' structure

The rest of the report is structured as follows:

1. **INTRODUCTION**
2. **LITERATURE REVIEW**: description of the research done to be able to do the actual work of building and testing an algorithm for velocity estimation.
   - **2.1. Digital cameras and image sensors**: brief introduction to how a camera sensor works, including information about **CCD sensors**, **CMOS sensors**, **Noise on imaging sensors** and **External uncertainty sources**. Here, **Camera modelling** is also explained.
   - **2.2. Visual Odometry**: the topic of visual odometry is introduced here, that is, the estimation of the egomotion of a body using on-board cameras. A brief historical introduction is done in **History**, followed by the types of existing VO depending on the utilised number of cameras, in **Stereo and Monocular VO**, a review of some works that have faced **Velocity estimation using visual odometry** problems, and finally a classification of some **Motion estimation techniques using feature-based methods**, which is the approach followed in this work.
3. **UTILISED HARDWARE TO GATHER THE DATA**: This chapter presents the hardware that was utilised to record the video and the synchronised GPS data as well as the camera modelling parameters.
   - **3.1. Hardware**: brief description of the utilised hardware.
   - **3.2. Camera modelling**: procedure followed to obtain the intrinsic camera matrix and the radial distortion coefficients.
4. **ALGORITHM OVERVIEW**: in this chapter, the utilised algorithm to estimate the velocity is explained.

4.1. **Pre-processing steps**: some pre-processing steps must be done before starting with the VO algorithm itself.

4.2. **Motion estimation: 2D-to-2D**: explanation of the way the camera pose can be estimated with respect to the previous one using a 2D-to-2D method. To do so, some steps must be follow: first **Feature detection: FAST algorithm for corner detection**, then **Tracking the corner features and redetection** and finally the **Estimation of the essential matrix and *extraction* of *R* and t** from it.

4.3. **Computing the scale factor**: due to with a monocular camera the absolute scale of its translation cannot be computed directly with the above algorithm, here a way to estimate a scale factor is detailed. It consists in calculating the distance from the virtual camera to the virtual road plane by performing a **Road feature detection, tracking and triangulation** to then **Fit a plane to the point cloud and compute the scale factor** based on the knowledge of the actual camera height.

5. **ANALYSIS OF THE RESULTS**: In this chapter, the explained algorithm in Chapter 4 is analysed in terms of accuracy of its output measurements.

5.1. **Description of the recorded data**: brief description of the utilised video and trajectory followed by the car, as well as its subdivision in three sections to be independently analysed.

5.2. **Smoothing the raw VO output**: explanation of the procedure utilised to smooth the raw noisy output of the VO algorithm. It consists in two steps: using a **Moving average filter** and establishing an **Acceleration limit**.

5.3. **Influence of the frame feature tracking threshold**: in this section, the influence of the defined threshold to redetect corners in the whole frame is analysed.

5.4. **Influence of the road feature tracking threshold**: in this section, the influence of the defined threshold to redetect corners in the road part is analysed.

5.5. **Influence of temporal and spatial resolution**: in this section, the influence of both spatial and temporal resolutions is analysed.

6. SUMMARY AND CONCLUSIONS

6.1. Summary: summary of the thesis work.

6.2. Conclusions: obtained conclusions of the analysis.

6.3. Further study: further study that can be carried out.

1.4 Thesis' contributions

The main contributions of this work are in the technical part in terms of software development. Although the majority of that software is based in some already existing functions, the fact of putting everything together and, more important, making it work with good results is the key point in this work. Thus, the main contributions can be summarised as follows:

- Development of the VO MATLAB software to estimate the camera movement from frame to frame (section 4.2).
- Development of the MATLAB software to obtain a scaling factor and get a proper estimation of the actual translation vector of the camera from frame to frame (section 4.3).
- Getting good results of the estimated velocities with the developed software after smoothing the raw measurements, and studying the influence of the threshold that limits the acceleration of the algorithm raw output (section 5.2).
- Analysing the influence of the redetection feature tracking thresholds for both the whole frame and only the road, finding that their effect is almost non-existent (sections 5.3 and 5.4).
- Analysing the influence of both the spatial and the temporal resolution in the results (section 5.5).
- Find the limitations of the adopted approach as well as the range of applicability (Chapter 5).

2 LITERATURE REVIEW

This chapter is a summary of the research done before beginning the actual work for velocity estimation. First of all, a brief introduction to the different imaging sensors is done in section 2.1 also including their main sources of noise and uncertainty as well as how a camera can be modelled to project 3D world points on to the image sensor 2D coordinate system. Then, in section 2.2, visual odometry is presented and explained, some works about velocity estimation using a camera are also introduced, and the motion estimation techniques using feature-based methods are expounded.

The information collected in section 2.1 is mostly used as an introduction to the topic of computer vision. However, sub-section 2.1.5 presents the way a camera can be modelled and the found information about this topic is utilised later on in section 3.2.

Section 2.2 introduces the topic of visual odometry and contains information about different techniques and the way some of them have been applied to the task of velocity estimation. All this knowledge has been useful to develop the contents of chapter 4.

2.1 Digital cameras and image sensors

Digital cameras are those that utilise digital sensors to capture images instead of a photographic film. The images are then stored in the memory of the device. Nowadays two main technologies of image sensors exist: CCD and CMOS sensors.

2.1.1 CCD sensors

The first kind of sensors, the CCD (charge-coupled device) sensors, are integrated circuits utilised in digital cameras to record an image. They have lots of individual semiconductor capacitors that can transform photons into electron charges thanks to the photoelectric effect.

Sensor are located behind the camera's lens, which focus the scene in front of it onto the sensor. When light falls on it, the capacitors accumulate electric charge

proportionally to the number of photos hitting that element, that is, proportionally to the light intensity. Then this charge is transferred line by line to its neighbour and the last one into an amplifier, converting that charge into a voltage. This way, CCD sensors works by shifting the content of each array of pixels to the next one obtaining at the end a sequence of voltages that are sampled, digitalised and stored in a memory.

2.1.2 CMOS sensors

This kind of camera sensors are based in the *complementary metal-oxide-semiconductor* (CMOS) technology, which allows integrating more functions into a same sensor, such as luminosity control or an Analog-to-Digital Converter.

These sensors are also based on the photoelectric effect. Each one has numerous photodiodes, one per pixel, that generates an electrical current that varies depending on the light intensity they receive. The main difference between this technology and the CCD is that CMOS sensors have an active amplifier for every photodiode that converts the charge into a voltage.

The main advantages of CMOS sensors compared with CCD is that in general the former is cheaper and utilised less energy. CMOS sensors have also a better control of blooming and read more pixels simultaneously. However, they also have some drawbacks, such as the rolling shutter effect when capturing high speed movements caused by the fact that CMOS sensors read a line at a time.

2.1.3 Noise on imaging sensors

A way to measure the noise in imaging sensors is to measure the Signal-To Noise-Ratio (SNR). It is the relative magnitude of the signal compared to the uncertainty in that signal on a per-pixel basis, that is, it is the ratio of the measured signal to the overall measured noise (frame-to-frame) at that pixel [1]. Having this into account, in imaging sensors the objective is not to reduce the noise, but to extract a signal from the noise [2] or, what is the same, to have a greater value of SNR. Hence, although having a lower signal will provide less noise (of the type that increases with signal), it does not mean the information

obtained from that image is better than other with a higher signal and, therefore, more noise if the latter has a better SNR than the former. The main sources of noise in image systems are the ones listed below [1], [2]:

- Photon noise: it is produced due to the natural variation of the incident photon flux into the sensor and increases with the intensity of the signal (see Figure 2-1 b). The number of photoelectrons gathered by each pixel have a Poisson distribution.
- Dark noise: it is caused by the thermally generated electrons in the sensor, additionally to those generated by the absorption of photons. The number of electrons generated per unit of time depends on the temperature of the sensor. That is why this kind of noise is also known as *thermal noise*. It also follows a Poisson distribution. Cooling the sensor can reduce the dark noise.

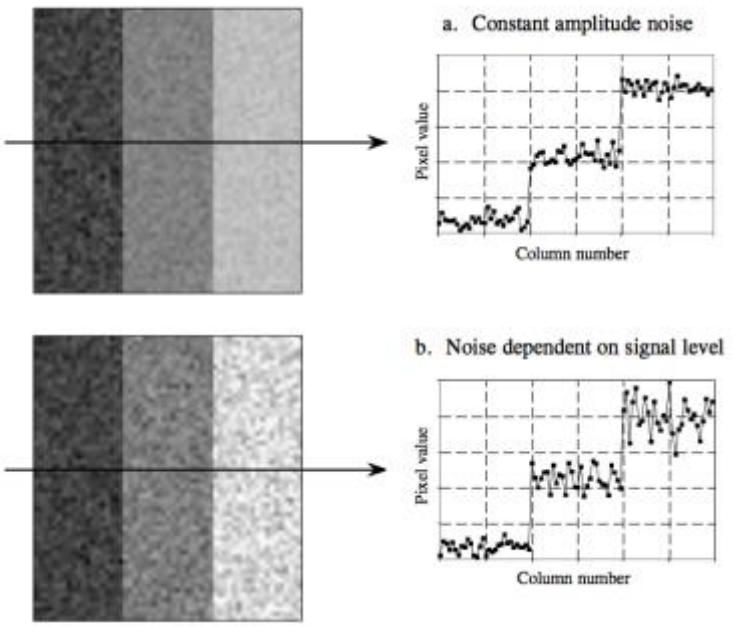

Figure 2-1 Image noise. In a) the amplitude of the noise remains constant as the signal level changes. In b) the amplitude of the noise increases as the square-root of the signal level.

- Read noise: this source of noise is due to electronic sources, representing the error introduced when quantifying the electronic signal on the sensor [1]. The main contribution to this kind of noise arises from

the amplifier and its amplitude does not depend on the light intensity (see Figure 2-1 a). Read noise can be reduced by a good design of the electronics.

2.1.4 External uncertainty sources

Apart from the inherent noise of the camera sensors, of internal nature, there also are other external uncertainty sources that does not depend on the camera hardware itself but on the recording conditions:

- Vibrations: sudden and fast movements of the camera may alter the readings from the video sequence giving unexpected results due to those undesired movements.
- Scene illumination: regular camera sensors capture visible light so to record a scene properly, there must be enough illumination. Sudden changes in illumination may affect the readings of the sensors and make, for example, some inter-frame tracking methods to have a worse performance [3].
- Contrast: this characteristic is very important especially if some salient features are needed. Images must have enough contrast to distinguish them from other regions of the picture, such as the background. Cameras have a limited dynamic range what leads to lose information in dark areas or in oversaturated backgrounds.
- Shadows: this is another element that can influence the quality of the results of a computer vision application, especially if shadows come from moving objects, that is, if shadows are moving. The reason why, is that they may be confused with moving objects.

2.1.5 Camera modelling

To model the geometry of the camera, the perspective camera model and, in particular, the pinhole camera projection system is a well-known model and the most utilised for perspective cameras.

In the pinhole camera model the image plane (grey rectangle in Figure 2-2) is assumed to be in front of the optical centre *C* at a distance *f*, which is the focal

length of the camera, and perpendicular to the optical axis. In addition, the optical centre is also the origin of the 3D camera coordinate system.

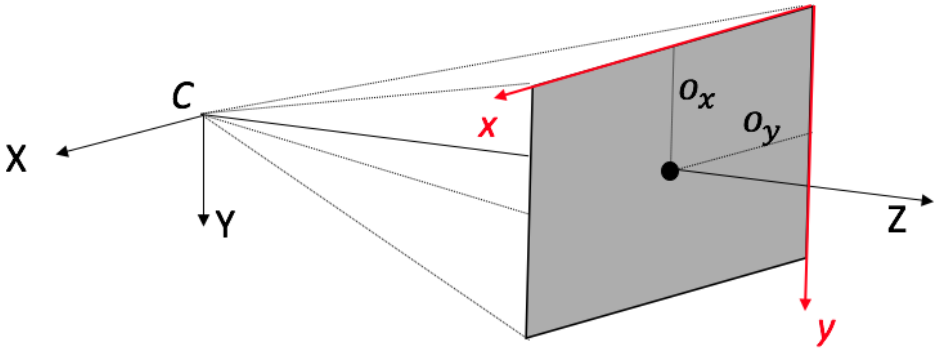

Figure 2-2 Pinhole camera model with reference systems. In red, the image plane reference system, and in black, the 3D camera coordinate system.

Using this model is easy to obtain the projection of any 3D world point into the image plane, taking into account that they are subjected to rigid body motion (see [4] for more details), using the next equation

$$\underline{\boldsymbol{x}} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad \textbf{(2-1)}$$

where $\underline{\boldsymbol{x}}$ is the homogeneous coordinates of a point in the image plane, r_{ij} and t_k are the coefficients of the rotation-translation matrix, and *X, Y, Z* are the homogeneous coordinates of the 3D world point.

But another transformation is required to obtain a projection of continuous points into a discrete reference system: the sensor coordinate system, measured in pixels. Following the reference [4], the next projection can be obtained:

$$\underline{\boldsymbol{u}} = \begin{pmatrix} \alpha & s_\theta & o_x \\ 0 & \beta & o_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad \textbf{(2-2)}$$

$$\underline{\boldsymbol{u}} = \boldsymbol{K} \cdot \boldsymbol{W} \cdot \boldsymbol{X}$$

where α and β are the focal lengths in x and y directions, respectively, measured in pixels, s_θ is the skewness of the image plane (usually ~ 0), o_x and o_y are the location of the image centre in pixels, and $\underline{\boldsymbol{u}}$ is the sensor homogeneous coordinates. $\boldsymbol{K}$ is the intrinsic camera matrix and contains the necessary information about the camera inner parameters to perform a complete perspective imaging transformation using (2-2); and $\boldsymbol{W}$ is the matrix that contains the extrinsic parameters of the projection.

The pinhole camera model described so far does not take into account any kind of distortion so it is not valid for a regular camera (supposing its lenses distorts the final image) unless the images are corrected. Lens distortion is usually a complex physical phenomenon that can be modelled with enough accuracy as a single-variable polynomial function [4]. The main types of lens distortion are pincushion and barrel, both depicted in Figure 2-3.

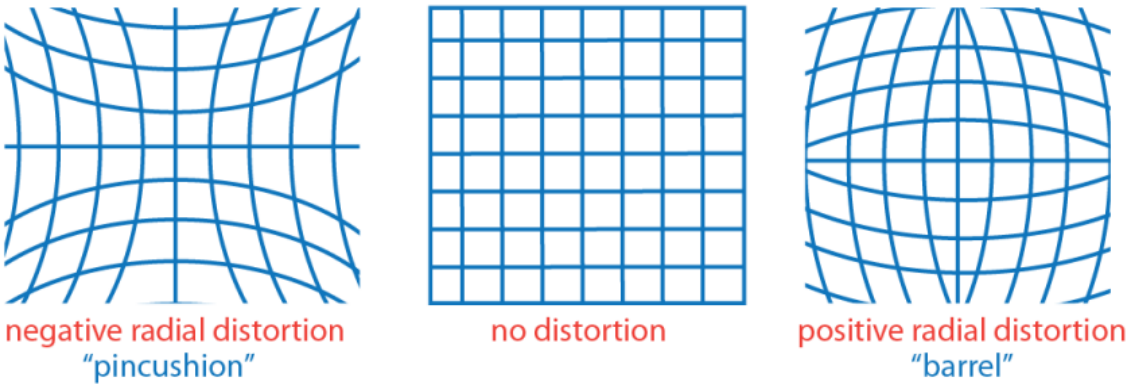

Figure 2-3 Types of radial distortion compared with a non-distorted image [5].

Lens distortion affects the projection before the image-to-sensor transformation defined by the intrinsic camera matrix is applied [4]. Hence, a way to correct it is modifying the image plane coordinates before multiplying them by the intrinsic camera matrix. To do so, a radial distortion model is utilised, because it is the most common model used for correcting lens distortions:

$$x_{distorted} = x(1 + k_1 \cdot r^2 + k_2 \cdot r^4)$$
$$y_{distorted} = y(1 + k_1 \cdot r^2 + k_2 \cdot r^4) \quad \textbf{(2-3)}$$

where

$$r^2 = x^2 + y^2$$

2.2 Visual Odometry

According to [6], visual odometry (VO) can be defined as “the process of estimating the egomotion of an agent using only the input of a single or multiple cameras attached to it”. This term was given for its similarity to wheel odometry, which consists in the estimation of the motion of a vehicle by integrating the number of times the wheels spin. Nister was the one that helped to use this terminology thanks to its famous paper about VO [7]. Hence, once the egomotion of the vehicle has been recovered using cameras, it is also possible to obtain its velocity if the time between the images is known. Even with the knowledge of only the relative motion of the camera between two consecutive frames, a velocity estimation can be obtained, without the need of computing the whole vehicle trajectory.

VO works in a similar way than wheel odometry but using images instead. The idea is to estimate the pose of the vehicle, in an incremental way, by studying the changes on the images, taken with the on-board cameras, produced by the movement of the agent. In order to do that, the imagery gathered by the cameras must be good enough, that is, it must be enough static scene sufficiently illuminated and with sufficient texture [6]. The reason for this is to be able to extract apparent motion. An additional requirement to make VO work properly is to have enough scene overlap from one image to another so common features can be identified in both.

2.2.1 History

VO is a particular case of what in computer vision is known as *structure from motion* (SFM), which consists in recovering the relative camera pose and 3D structure from a set of images [6]. SFM is a technique utilised since the 1980s in works such as [8] and [9], and is more general that VO as it solves the problem of reconstructing in three dimensions both the camera poses and the structure [6]. VO, however, focuses only in the 3D motion estimation of the camera.

The first researches in the field of visual odometry were made for planetary rovers also in the 1980s and 1990s, such as the one of Moravec [10] and the one of Lacroix et al [11]. In the former, for example, a computer utilises the information gathered with a *slider stereo*[1] to plan an obstacle avoidance path to a certain destination. The system was reliable in the short term but it needs time to process the data, moving only one meter every ten minutes [10]. This work presented the baseline for VO, whose main ideas are still in use nowadays.

2.2.2 Stereo and Monocular VO

Stereo VO is the one that uses two cameras to compute the relative 3D position of the detected features directly by triangulation and then derive the relative motion [6]. Hence, the methodologies utilised by Moravec in [10] belongs to this group due to the pictures were taken from different perspectives, sliding the camera, but being the vehicle in the same place.

In general, in stereo VO works the relative motion is computed in a 3D-to-3D basis, triangulating points for every stereo pair. Nister et al [7] improved those implementations changing some parts. Instead of tracking the features, they detect them independently in every frame and search for matches between them. They also set the problem as 3D-to-2D to estimate the relative motion of the camera. And finally, they also included RANSAC (Random Sample Consensus) outlier rejection algorithm in the motion estimation step [6].

On the other hand, it is the monocular VO. In this case only one camera is utilised. Its main disadvantages are that only bearing information is available and motion can only be recovered up to a scale factor [6]. The absolute scale can be determined by fusing it with other sensors, such as wheel odometry or IMUs, or by knowing the size of an element in the scene. The importance of monocular VO is that stereo VO degenerates to a monocular case when distances are much bigger than the one utilised as the baseline [6], typically the

[1] A *slider stereo* is a term utilised in [5] to refer to a single camera moving on a rail.

distance between both cameras. Hence, in this situation monocular methods are needed.

According to [6], monocular VO works can be classified into three categories:

- Feature-based methods. The main idea in works of this group, such as in [7], [12], is to detect and track salient features from frame to frame. The RANSAC method along with the five-point minimal solver, introduced in [13], are utilised in these and many other publications for outlier rejection and also for 3D-to-2D camera-pose estimation.
- Appearance-based methods. The intensity information of all the pixels, or of a subregion of the image, is utilised to estimate the motion. Into this group, works that utilised optical flow are included, such as [14], [15] or [16]. The last one utilised optical flow only to estimate the velocity of a car.
- Hybrid methods. These approaches are a combination of the other two.

One of the main drawbacks of appearance-based methods is that they are not robust to occlusions [6] and also the computation of the optical flow over an entire image is computationally expensive [6], [15]. In general, they are less accurate than feature-based methods [6]. However, they have been utilised widely in monocular VO works because they are easier to implement than feature-based methods when compared with the stereo camera case [6]. But feature-based methods usually are faster and more accurate, being the most utilised approach in VO.

2.2.3 Velocity estimation using visual odometry

The main aim of this project is to make an estimation of the actual velocity of a car using a monocular camera. This is why, in this subsection, some works that do this are presented.

To begin with, in [17] the authors present an algorithm to measure the velocity of on-road cars using a monocular camera detecting the relative motion of the road in front of the vehicle with respect to itself (Figure 2-4). But some assumptions and restrictions were made. It was developed to work only on

straight roads on a flat terrain and the road must have some distinctive landmarks (lane markers, edges of the road, etc.) in order to identify the road. The authors utilised three methods to compute the relative motion, two of them from the feature-based methods (feature matching and feature tracking) and the other one based on optical flow. The latter was the one that had the worst performance, mainly because of the lack of texture of the road. However, the final results with the other approaches were not very good either as they were noisy and did not follow well the actual velocity of the car.

Figure 2-4 Representation of the ROI within the image [17].

The authors of [18] utilised a method based on optical flow techniques to estimate de velocity of a UGV with a camera located underneath its body pointing towards the ground, as shown in Figure 2-5. It was validated over several kinds of terrain surfaces, having good results when compared with actual velocities. However, the tested speeds were quite small (around 0.2 km/h) to be useful in a real car moving much faster. One of the main parameters they had into account was the distance between the camera projection centre and the ground, since it greatly influenced the final results.

Finally, a special mention is given to the work presented in [16], since it is the past year's thesis about this topic. It utilised a fusion algorithm of data from IMU readings and a forward-looking monocular camera to estimate the velocity of a car. The latter data was processed using VO and, in particular, using the Farnebäck optical flow algorithm in order to extract the velocity of the vehicle by

estimating the velocity of each pixel and then filtering out the outliers. The proposed method worked well when the outputs of both sensors were combined, but the results obtained only with VO were quite noisy, although they follow a similar trend as the ground truth.

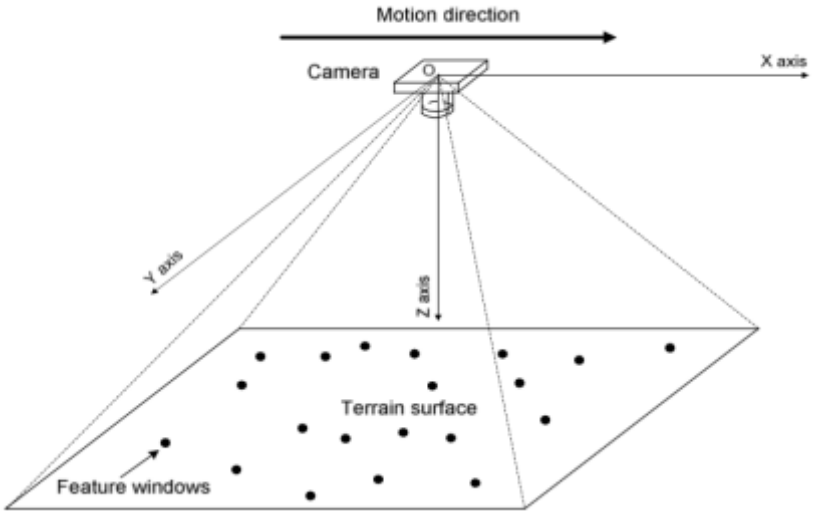

Figure 2-5 Camera and terrain surface [18].

The aim of this thesis is to obtain an estimation of the velocity of a vehicle, but a different approach has been utilised. Instead of using optical flow as the works presented above, the methodology has consisted in using a feature-based method to not compute the movement of every single pixel, but of some salient features of the scene. Then, the movement of the camera can be estimated and, knowing the frame rate at which each image has been taken, the velocity can be computed.

2.2.4 Motion estimation techniques using feature-based methods

In a VO system, the motion estimation of the camera and, hence, of the vehicle (if the camera is rigidly mounted), using an image-pair is the most important step as it gives an estimate of how the camera has moved in the 3D world. Once this is known, its full trajectory can be obtained by concatenating every single movement, although, for velocity estimation only the individual camera movements are necessary.

In order to recover the camera pose of one frame with respect to the camera pose of the previous one using feature-based methods, different techniques exist:

- 2D-to-2D: in this case the corresponding features of both frames are specified in 2D image coordinates.
- 3D-to-2D: the features extracted from the previous frame are in 3D and the ones of the current frame are the reprojections of those 3D points into the current image. In the monocular case, at least three different images from different camera poses are necessary because the 3D structure is obtained by triangulating points from two adjacent views and then matched to 2D image features in a third view [6].
- 3D-to-3D: using this methodology implies both the features from one time and the previous one are in 3D. This is only possible in the stereo case [6].

Features can be lines or points but, in general, in VO point features are usually utilised due to the fact that are more common in unstructured scenes [6] and also because their position in the image can be measured precisely [19]. They can be either blobs (image patterns that differ in terms of colour, intensity… from its neighbourhood) or corners (point at the intersection of two or more edges).

In order to detect those salient point features, different feature detectors exist, each of them with different properties and ways to find the features. But in general, the important properties they should have to be considered as good feature detectors, according to [19] are:

- **Localization accuracy**.
- **Repeatability**, so a large number of features can also be detected in next frames.
- **Computational efficiency**.
- **Robustness** to noise, compression artefacts, etc.
- **Distinctiveness**, to accurately match features between frames.

- **Invariance** to geometric and photometric changes.

In general, corner detectors, such as FAST, Harris or Moravec, are faster than blob detectors (SIFT or SURF, for instance) whereas the latter are more distinctive. However, the choice of one or another depends on several factors such as the time and computational constraints or the environment in which the VO algorithm is working. For example, in [19] is pointed out that for urban environments SIFT does not take into account corners and they are very abundant on those scenarios.

From now on, when referring to *features* in this report, it will be referring to point features (and, as it will be explained in section 4.2.1, to corner features), unless it is clarified another kind of feature has been used.

2.3 Conclusions of the literature review

Estimation of velocity using a monocular camera requires the use of image processing and computer vision techniques to be applied to the direct output of that device. However, as any other sensor, images sensors have some noise and uncertainty sources that may affect the quality of the imagery in terms of extract valuable information from it. Hence, in order to be able to process the video to obtain a measure of the velocity of a vehicle, it is necessary to have a scene sufficiently illuminated and static with enough texture.

The camera must be modelled to allow for knowing the equivalence of the position of a point in the real 3D world to its projection in the image sensor plane in 2D. To do so, the pinhole camera model is a good option because it is a well-known camera model and lens distortion can be corrected without changing the model itself.

VO allows for estimating the egomotion of a vehicle and is the way to proceed to obtain the velocity of a car, as it is the case in this work. There are different techniques of VO but the ones that have been already utilised for velocity estimation are based on optical flow. However, these methodologies have given results that are noisy or that work only under controlled circumstances and slow speeds, mainly because they rely in in the texture of the captured images,

especially the road/ground texture, and they need small motion between frames to compute correctly that flow.

For this reason, a different approach has been utilised in this work. The followed methodology has consisted in using a feature-based VO method to compute only some salient features on the scene and then estimate the motion of the camera and measure how much it has been moved per unit time.

3 UTILISED HARDWARE TO GATHER THE DATA

This chapter presents in the first place the hardware that was utilised to record the video and the synchronised GPS data. In section 3.1 the utilised hardware is presented and described and in section 3.2 the intrinsic camera matrix and the coefficients necessary to undistort the images are obtained.

3.1 Hardware

The utilised camera in this project has been the one that comes with the video data logger VBOX Video HD2 of RACELOGIC, shown in Figure 3-1. This system has two individual wide-angle cameras (only one has been used), both with the same technical specifications, and a GPS logging unit, which in this project has been utilised only to obtain the GPS velocity. Both data, video and GPS velocity, are time-synchronised.

The main technical characteristics for this system are displayed in Table 3-1 and Table 3-2. Finally, it is important to mention that the camera was firmly attached onto the roof of the car at a height of 1.46 m above the ground, since this distance is necessary to compute a scale factor (explained in 4.3).

Clarify that the data utilised as ground truth is the one provided by the GPS unit. The measured velocity is only from the GPS (it is not fused with any other measurement device) and its accuracy would probably be less than the one specified on Table 3-2 since the test section where the data was gathered has trees, buildings and other elements that may affect the measurements.

Figure 3-1 VBOX Video HD2 video data logger of RACELOGIC Ltd [20].

Table 3-1 Technical specifications of the camera [20].

Resolution	1920 x 1080 pixels
Frame rate	30 fps
Focal length	1.97 mm
Pixel size	2.8 μm x 2.8 μm

Table 3-2 Technical specifications of the GPS logger [20].

Frequency of data logging	10 Hz
Accuracy	0.1 km/h
Resolution	0.01 km/h

3.2 Camera modelling parameters

Both the intrinsic parameters and the radial distortion coefficients can be computed easily using already developed MATLAB functions, that is, doing a calibration process taking some especial pictures of a known pattern. However, the idea in this project is to not depend on external calibration processes as much as possible. That is why the unknown parameters has not been obtained this way but using the known technical specifications of the camera provided by RACELOGIC.

3.2.1 Intrinsic camera matrix

Using the data in Table 3-1 all coefficients of the intrinsic matrix (see eq. (2-2)) can be obtained. α and β, the focal lengths in pixels, are considered to be the same. The can be obtained as follows

$$\alpha = \beta = \frac{f}{Pixel\ size} = \frac{1.97\ mm}{2.8 \cdot 10^{-3}\ mm/pixel} = 703.57\ pixels \quad \textbf{(3-1)}$$

As the skewness is usually zero or very small, it has been assumed zero.

Finally, the position of the image centre in pixels has been set as half of both image dimensions:

$$o_x = \frac{Horizontal\ length}{2} = 960\ pixels$$
$$o_y = \frac{Verticla\ length}{2} = 540\ pixels \quad \textbf{(3-2)}$$

Hence, the intrinsic camera matrix is

$$\boldsymbol{K} = \begin{bmatrix} 703.57 & 0 & 960 \\ 0 & 703.57 & 540 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3-3)}$$

However, it is important to emphasise that this intrinsic matrix is only valid when using the original video resolution, that is, Full HD. If the spatial resolution is downsampled, both the focal lengths and the position of the image centre will vary. The latter is quite straightforward to be calculated. If a scale factor *s* is defined (being *s* a value between 0 and 1), the position of the image centre will be

$$o'_x = o_x \cdot s$$
$$o'_y = o_y \cdot s \quad \textbf{(3-4)}$$

The same applies to the focal length since they form similar triangles (see Figure 3-2):

$$f' = f \cdot s \quad \textbf{(3-5)}$$

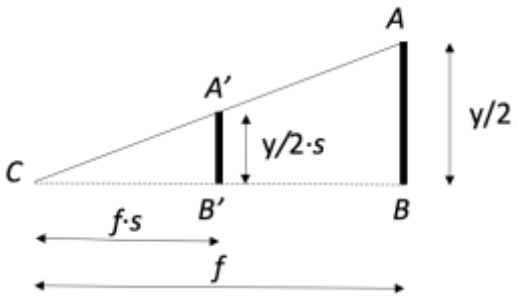

Figure 3-2 2D geometry of the focal length and image plane when the size of each frame is downsampled to a lower spatial resolution.

3.2.2 Radial distortion coefficients

As commented in the description of the camera, it has a wide-angle lens that allows to capture more scene but at the cost of a big image distortion, as shown in Figure 3-3.

Figure 3-3 Frame from the first seconds of the video. A big *barrel* distortion caused by the wide-angle lens can be appreciated.

In this case it is a radial distortion, which occurs when light rays bend more near the edges of a lens than they do at its optical centre [5], and, in particular, it is *barrel distortion* (see Figure 2-3), typical from wide-angle lenses [4].

Obtaining the radial distortion coefficients has been made by trial and error, thanks to the first minute of video where the car is located in a car park with car park markings that are straight lines, as shown in the satellite image of Figure 3-4. Because of the wide-angle lens, they appear to be curved in the footage (Figure 3-3 and Figure 3-5 left), so the parameters k_1 and k_2 from equation (2-3) have been chosen to straighten those lines.

To do so, first of all the intrinsic camera matrix was defined in MATLAB using the command *cameraParameters*, since it is needed to use the command *undistortImage*. Two arbitrary numbers were also included in this MATLAB object as values for k_1 and k_2 based on values from other wide-angle cameras from [21].

Figure 3-4 Satellite image of the car park with its car park markings where the motion of the vehicle starts in the video. (Photo courtesy of Apple Maps © 2012-2016 Apple Inc.)

Figure 3-5 Comparison of the original footage (in greyscale) with the images distorted (left images) and the same frames after the lens distortion correction (right images). The red straight lines have been added to these frames to make easier the comparative.

Once the object was created, k_1 and k_2 were varied until a good result was obtained using *undistortImage*, shown in the right images in Figure 3-5. The obtained values are the next ones:

$$k_1 = -0.156$$

$$k_2 = 0.013$$

3.3 Conclusions

Synchronised GPS and video data has been provided by RACELOGIC at different rates, 10 and 30 Hz respectively, that must be resampled in order to have a measurement of GPS for each frame of the video.

A very important part of every computer vision problem is calibrating the camera to know the equivalence of the position of a point in the real 3D world to its position in the image sensor plane in 2D. To do so, the parameters that define the intrinsic camera matrix and the radial distortion model have not been obtained using a standard calibration processes but getting them from the hardware specifications and, in the case of the distortion model, by trial and error of different values until a good correction was obtained.

4 ALGORITHM OVERVIEW

As explained in the section 2.2 *Visual Odometry* of the literature review, a camera can be used to estimate de egomotion of a vehicle and also its velocity. From the different feature-based methods that exists for VO (mentioned in section 2.2.4), in this project the 2D-to-2D approach has been selected due to the next reasons:

- The available video data is from only one camera (monocular scheme). Thus, the 3D-to-3D is immediately discarded.
- As it is mentioned in [6], the 2D-to-2D approach is better than the 3D-to-2D case, mainly because the former does not need to triangulate points from the whole scene as the latter does. This triangulation step is the main challenge of this method because consistency and accuracy must be maintained for every set of triangulated points as well as the creation of 3D-to-2D matches for at least three consecutive views [6]. However, as it will be explained later on, a triangulation step has been included in the final 2D-to-2D algorithm, but only to compute the road plane in order to obtain the absolute scale of each movement and, hence, a proper estimation of the velocity.

In Figure 4-1 a diagram of the main part of the algorithm is shown. First of all, the left branch is explained (section 4.2) and corresponds to the VO itself where features are detected in one frame to then be tracked to the next frames to compute the relative pose between them. When the number of features falls below a threshold, a redetection step is executed (it is not shown in the diagram). The right branch corresponds to the scale factor computation to obtain a real measurement of the displacement of the camera and is explained in section 4.3. But before those two sub-chapters, the pre-processing steps that have been done are presented in section 4.1.

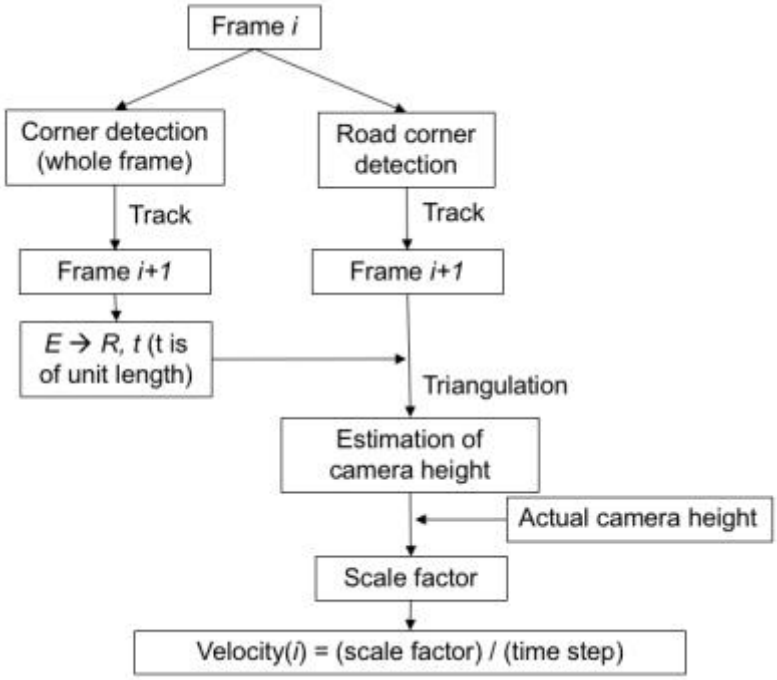

Figure 4-1 Diagram of the core of the utilised VO algorithm.

4.1 Pre-processing steps

Before running the VO algorithm, some previous steps need to be done:

1. Convert the original RGB video frames (image *a* in Figure 4-2) to greyscale (image *b* in Figure 4-2), because the feature detector and tracker that will be utilised later work only with greyscale images.
2. Correct lens distortion of each video frame (image *c* in Figure 4-2), using the radial distortion coefficients obtained in the section 3.2.2, using the MATLAB command *undistortImage*.
3. Rotate the image (image *d* in Figure 4-2). As it can be seen in the zoom of image *c*, the lamppost, which is a completely vertical object in an urban environment and is located in the centre of the image, is not vertical. This means that the camera is tilted a certain angle around its longitudinal axis resulting in a video that is rotated. That is why this step is necessary, to eliminate this rotation (that will cause the fitted plane, explained in section 4.3.2, to have an extra rotation and difficult its computation). The angle of rotation is 7º anticlockwise.
4. Downsample the spatial resolution if necessary.

5. Downsample the temporal resolution (frame rate) if necessary.

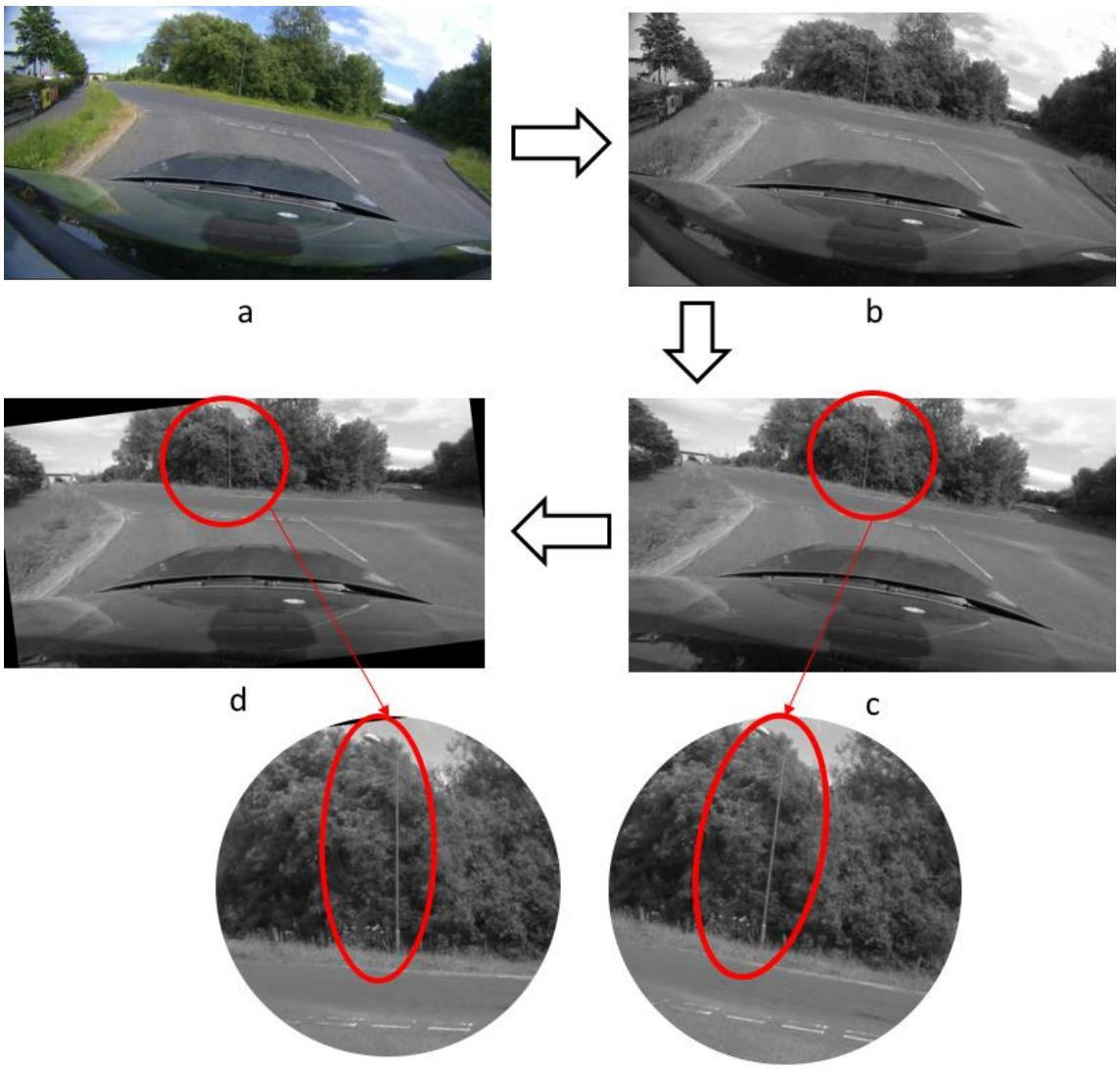

Figure 4-2 Pre-processing steps: a) Original frame (RGB); b) Greyscale frame; c) Undistorted frame; d) Rotated frame. The two bottom zooms highlight the inclination of the lamppost in image c) and the rotation correction in image d).

4.2 Motion estimation: 2D-to-2D

In this section, the left branch of the diagram of Figure 4-1 is explained step by step:

1. Detect corner features on frame *i* (section 4.2.1).
2. Track features to frame *i+1* (section 4.2.2).
3. Compute essential matrix for that frame pair (section 4.2.3).

4. Extract the relative camera pose from the essential matrix (section 4.2.3).
5. If number of features:
    a. < threshold, go to step 1
    b. > threshold, go to step 2

4.2.1 Feature detection: FAST algorithm for corner detection

In section 2.2.4 a brief introduction to some point detectors was made. Here, one of them is chosen and explained.

Table 4-1 Comparison of feature detectors: properties and performance [19].

	Corner Detector	Blob Detector	Rotation Invariant	Scale Invariant	Affine Invariant	Repeatability	Localization Accuracy	Robustness	Efficiency
Haris	x		x			+++	+++	++	++
Shi-Tomasi	x		x			+++	+++	++	++
FAST	x		x	x		++	++	++	++++
SIFT		x	x	x	x	+++	++	+++	+
SURF		x	x	x	x	+++	++	++	++
CENSURE		x	x	x	x	+++	++	+++	+++

As mentioned in that section, point detectors can be either for detecting corners of blobs. The former is faster whereas the latter are more distinctive, that is, they are accurately matched better between frames. Additionally, as it can be seen in Table 4-1, where some point detectors are compared, blob detectors have more repeatability and are more robust. However, point features will not be redetected in each frame but they will be tracked from frame to frame (see section 4.2.2), so properties such as repeatability are not very important in this case. This, together with the fact that blob detectors are less accurate in terms of localization, allows to discard blob detectors. Then, regarding the corner detectors, the FAST algorithm has been chosen mainly because it is scale invariant and it is the most efficient one (although the final VO algorithm is not thought to work in real time, it will help to obtain results quickly). Moreover, the FAST corner detector has been used in VO applications such as in [22], [23]

and [24] with good results and, as it will be shown later on in this report, the obtained velocities using FAST corner detector in the developed algorithm are close to the ground truth.

4.2.1.1 FAST corner detector

The way FAST (Features from Accelerated Segment Test) corner detector works is quite simple. Basically, what it does is analyse the intensities of sixteen pixels around a corner candidate *p* (shown in Figure 4-3). First of all, the algorithm selects possible corner candidates and rejects those that are not. To do so, the intensities of four of those pixels, 1, 5, 9 and 13, are compared with the one of *p*. If three of them are brighter than *p* plus a threshold, or darker than *p* minus a threshold it will be considered as a candidate [25]. Then, for the candidate points, if the intensities of 12 contiguous pixels of the circle are above or below the one of *p* by some threshold, then the candidate point *p* will be considered as a corner [25].

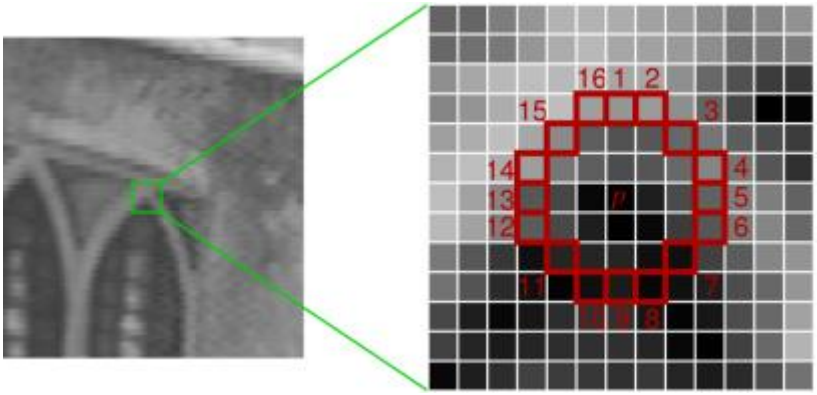

Figure 4-3 Example of the 12-point segment test corner detection used in FAST. The red squares represent the pixels utilised in the corner detection, being *p* the centre of the possible corner [25].

In MATLAB, this is implemented with the function *detectFASTFeatures* included in the Computer Vision System Toolbox. It receives as input an image, in this case each video frame, and also some other arguments to be defined:

- *MinQuality* – this parameter defines the minimum quality of the corners. It "represents a fraction of the maximum corner metric value in the image" [26].

- *MinContrast* – It is utilised to specify the minimum difference between the intensity of the corner and its surroundings [26].
- *ROI* – Region of interest where corners must be detected.

The first two arguments have a strong influence in the number of features detected. For both of them, the greater they are (maximum 1) the less corners are found on an image. That is why their values have not been chosen to be very big. In fact, a value for *MinContrast* as low as 0.2 (with *MinQuality* by default, that is, 0.1) returns much less features than 0.1, 247 compared with 2364 corners (see Figure 4-4 and Figure 4-5 as example). Moreover, for values equal or bigger than 0.3, the VO algorithm failed because at some points of the video there were not enough points to reliably estimate the essential matrix (explained in section 4.2.3). That is why ***MinQuality*** has been left with its default value, **0.1**, and ***MinContrast*** as **0.05**, to have a good quantity of corners to estimate the essential matrix.

Then, another thing that has been made is something that is known as *bucketing*, which consists in dividing the image in several individual zones creating a grid, using the *ROI* parameter, to detect the corners separately in each zone. This is done to obtain a more uniform distribution of the features, because using the feature detector directly over the entire image would probably give zones with plenty of corners whereas other areas would have almost no features. Additionally, *ROI*s have been defined only on the top half of each frame to avoid detecting features on the bonnet that would give erroneous readings due to reflections.

Finally, from the found corners, only the 70 with the strongest metric, their quality, from each bucket are utilised for the next step. This is done using the MATLAB command *cornerPoints.selectStrongest* where *cornerPoints* is the output of *detectFASTFeatures*. So, because in the *bucketing* part a grid of 10x5 (wide x high) buckets is utilised, the maximum number of features per frame is be 3500 corners.

Figure 4-4 Detected corners (red crosses) in a frame of the video with the FAST feature detector configured as follows: Minimum quality = 0.1, Minimum Contrast = 0.2 and ROI = upper half of the image. Total detected corners = 247.

Figure 4-5 Detected corners (red crosses) in a frame of the video with the FAST feature detector configured as follows: Minimum quality = 0.1, Minimum Contrast = 0.2 and ROI = upper half of the image. Total detected corners = 2364.

4.2.2 Tracking the corner features and redetection

Once a set of corners has been detected in one frame, the next step is to know their correspondences in the next image. To do so, two different approached can be used: feature matching and feature tracking.

The first approach consists in detect salient features in each frame to then find correspondences between the two sets by comparing all feature descriptors of one image with all of the other. Sometimes one feature from the second frame may have two or more good matches with features of the first one. To solve this problem mutual consistency check is utilised.

Tracking features, on the other hand, works differently. Corners are only detected in the first frame and then correspondences in the next frames are searched based on the previous ones. This approach works well in VO applications when the motion from one image to the next one is not very big [19], as it is the case in this project. The recorded video has a frame rate of 30 fps, which means that at the highest speed the car was driven during the recording, 75.38 km/h, the distance travelled between frames is only 0.69 m. Hence, for slower velocities this distance is shorter. If the temporal resolution is downsampled to 15 fps, for example, the maximum distance between frames would be 1.39 m. It may appear a big distance, but it is what the camera moves and not what the features will move in the images. Points closer to the camera will move a bigger distance than those situated far away. However, as it will be seen in *5 ANALYSIS OF THE RESULTS*, the results of the part of the video where those high speeds are reached are not very good. Nevertheless, on the rest of the footage it has work reasonably well.

In addition, this approach has been demonstrated to be valid in VO in the implementation done in [22] with a video of only 5 fps. Then in [27] a comparative between feature matching and tracking for vision-based navigation was done, revealing that the tracker gives better results than other feature matching options in aspects as ratio of successfully localized features and smaller reprojection errors. Hence, the utilised methodology in this project will be tracking the features detected on one frame.

In particular, the tracker utilised in [22] and [27] was the known as *Kanade-Lucas-Tomasi* (KLT) tracker, which is also mentioned in [19] as a good option for tracking features since it applies an affine-distortion model to them in order to take into account the changes in their appearance over long image sequences. KLT algorithm uses spatial intensity information to lead the search for the position (of a feature) that gives the best match. It examines fewer potential matches between the images than other trackers, what makes it a fast option when compared with other methods.

The KLT feature-tracking algorithm is included in the Computer Vision System Toolbox of MATLAB in the form of a system object called *vision.PointTracker* [28]. It works as follows:

1. Create the point tracker object using *vision.PointTracker*. This function also allows to include more properties about the point tracker that are briefly explained in section 4.2.2.1.
2. Initialise the point tracker object using the command *initialize.* To do so, a video frame is necessary. It will be the first one of the video to be processed (or another one if the number of tracked features drops below a certain threshold, explained in section 4.2.2.2). It is also necessary the initial location of the point features (extracted from the mentioned frame with the FAST corner detector).
3. Use the *step* method to track the previously initialised points to successive frames. The inputs to this function must be the point tracker object and the next frame. As outputs, the tracked points will be obtained together with a logical array, *point_validity*, containing the information of which of the tracked points have been reliably tracked. Due to several reasons, a point can be lost, such as it goes outside the frame, it is occluded or its maximum bidirectional error is greater than a threshold, for instance.
4. Repeat 3 until the number of tracked features drops below a certain threshold. Then, go to 2.

4.2.2.1 Properties of the KLT tracker MATLAB function

The properties that can be set for the point tracker are the next ones:

- *NumPyramidLevels* – This parameter represents the number of pyramid levels. The MATLAB implementation of KLT uses image pyramids, which are reduced resolution copies of the actual frame that allows the algorithm to track points at multiple levels of resolution, beginning with the lowest one [28]. The bigger the number of levels, the larger displacements could be tracked but at the cost of more computation resources. It has been left as default, that is, with a value of 3.
- *MaxBidirectionalError* – Threshold for a forward-backward error. Its value represents the distance from the original position of a point in frame *i-1* to its final location after tracking backwards from frame *i* to the frame *i-1* (see Figure 4-6). It has been fixed to 1 pixel.
- *BlockSize* – It defines the size of the area, called neighbourhood, where the spatial gradient matrix is computed. It has been left as default, that is, with a value of [31 31].
- *MaxIterations* – Maximum number of search iterations for each point. It has been left as default, that is, with a value of 30 since, according to [28], the algorithm typically converges within 10 iterations.

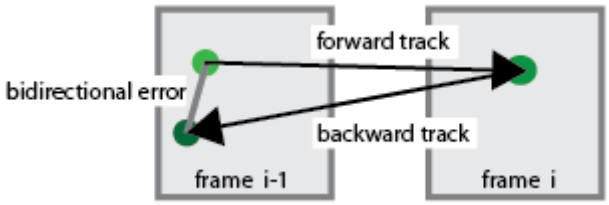

Figure 4-6 Diagram showing the definition of bidirectional error [28].

4.2.2.2 Feature redetection

As mentioned above, when corners are tracked along video frames, they may disappear from one frame to another, especially if the camera is moving. Thus, there will be a moment in which all of them would disappear and no camera pose or velocity could be estimated. This is why they need to be redetected from time to time to maintain a minimum number of corners, that is, a threshold

is needed, which will be called *frame feature tracking threshold* (*frame threshold* for short). In section 5.3 the influence of this parameter is analysed and, based on the results, a frame feature threshold is chosen.

4.2.3 Estimation of the essential matrix and extraction of *R* and *t* from it

Let's call $[I|0]$ and $[R|t]$ to the extrinsic camera matrices of two consecutive frames where the former is the first view (placed at the origin and with a rotation matrix equal to the identity) and the latter the second view (where *R* is the rotation matrix and *t* the translation vector, both with respect to the first view). The camera matrices of those two images will be

$$P_1 = K[I|0]$$
$$P_2 = K[R|t] \quad \textbf{(4-1)}$$

Knowing this, the essential matrix describes the geometric relations between two adjacent frames taken with a calibrated camera up to an unknown scale factor for the translation vector [6]:

$$E \backsimeq \hat{t}_k R_k \quad \textbf{(4-2)}$$

where $\backsimeq$ means the equivalence is true up to a scale factor and

$$\hat{t}_k = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \quad \textbf{(4-3)}$$

Because the camera is calibrated, an important characteristic of 2D-to-2D motion estimation, the epipolar constraint (depicted in Figure 4-7), can be expressed as [6], [13]

$$p'^T E p = 0 \quad \textbf{(4-4)}$$

Where *p* is the location of a feature in one image and *p'* is the location of the same feature in another frame. Hence, knowing the 2D-to-2D correspondences between features of two different frames and using the above constraint the essential matrix can be obtained [6].

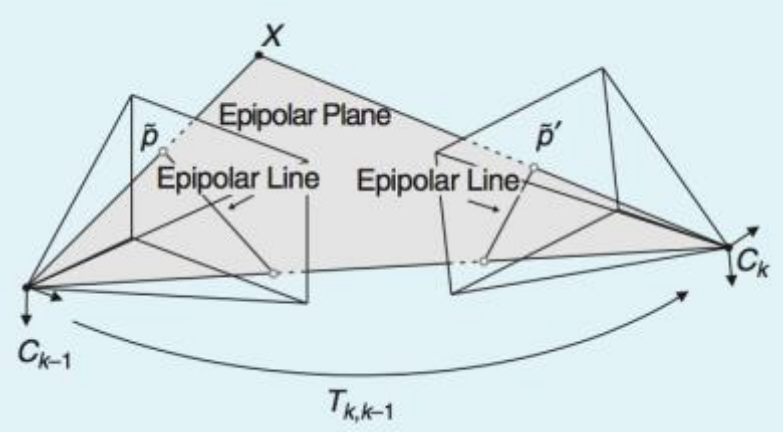

Figure 4-7 Illustration of the epipolar constraint [6].

There is an algorithm, called the *five-point algorithm*, that solve this problem. The implementation proposed by Nister [13] is the most utilised when there are outliers in the problem and is already implemented on MATLAB with the function *estimateEssentialMatrix*. Presence of outliers is due to several factors such as wrong matching/tracking of the features, occlusion of some of them, changes in the scene properties (illumination, contrast, etc.), etc. In order to overcome this problem, together with the five-point algorithm, the outlier removal known as RANSAC is utilised [13], also implemented in the function *estimateEssentialMatrix*.

RANSAC is an iterative method utilised to obtain the parameters of a mathematical model from a group of observations, the image corners in this case, with the presence of outliers. What RANSAC does is selecting random subsets of observations to then compute the model hypothesis with them [19]. After that, the hypotheses are checked with the rest of the points until one of them shows a high level of consistency with the model. Outliers will be those points that do not fit the model [19].

The *estimateEssentialMatrix* function needs as inputs the position, on both frames, of the points that have been successfully tracked from one frame to another, and the camera parameters.

Once the essential matrix has been computed, both the rotation matrix *R* and the translation vector *t* can be extracted:

$$R = UW^T V^T \quad \textbf{(4-5)}$$

$$\hat{t} = UWSU^T \quad \textbf{(4-6)}$$

where U, S and V^T come from the singular value decomposition (SVD) of *E* and *W* is

$$W \equiv \begin{bmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

To determine the ambiguities of this problem, it is necessary to assume the first camera matrix as $[I|0]$ and a translation vector *t* of unit length [13]. Four different solutions are obtained and the correct one can be found by triangulating one point that must satisfy the cheirality constraint, that is, the point should be in front of both cameras [13]. In MATLAB, the recovery of *R* and *t* is implemented in the function *relativeCameraPose*, which needs as inputs the essential matrix, the camera parameters and the inlier points that were utilised to compute the essential matrix (these are returned by the function *estimateEssentialMatrix*).

4.3 Computing the scale factor

This section explains the right branch of the diagram of Figure 4-1, which depict the core of the developed VO algorithm. A more detailed diagram of only this branch is shown in Figure 4-9.

The 2D-to-2D motion estimation step gives the relative pose of the camera but with the limitation that the translation vector is always of unit length. In order to obtain an absolute scale of that movement from one frame to another and, therefore, of the car, some additional information is needed. In this scenario, GPS data is available but it does not make sense to use it as it is the same data utilised as ground truth. On the other hand, IMU data is also provided. However, it has not been used for scaling the VO results because the translation vector obtained with VO is always of the same length. Hence, to scale it an estimation or measurement of the real translation is needed, but using the IMU

measurements, or of other sensor, will lead to have VO translation vectors of the same magnitude as the IMU readings. Therefore, the only alternative to obtain the scale is using a known distance that can be recovered from the footage. One option could be using the road markings but there are not always and their size can vary from one place to another. Hence, the height of the camera has been utilised. This distance is always the same, it is constant.

So, once the movement of the camera has been estimated, it is necessary to compute the road plane (named as *virtual road*) to obtain the distance from it to the *virtual camera* position and compute the scale factor using the actual height. To do it, the next hypothesis has been used:

- The actual road plane has been considered flat. In the analysed scenario, this assumption is quite reasonable as there are not big slopes (the difference from the highest and the lowest points is only of about 10 m, being the total length of the trajectory 1.7 km).
- The camera is tilted around its *X* axis a certain angle θ (see Figure 4-8 left) that is considered constant (angle between the longitudinal camera axis, *Z*, and the road) since the camera is firmly attached to the vehicle.

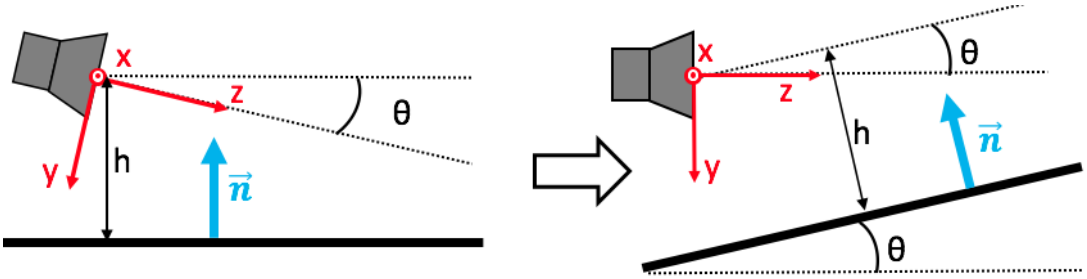

Figure 4-8 Diagram of the geometry between the camera and the road plane where $\vec{n}$ is the normal to the road and θ the camera inclination angle. On the left, actual configuration (considering the road as a horizontal surface). On the right, result from road plane estimation (*virtual road*).

Due to the second assumption, the *virtual road* will have a certain slope and will not be horizontal, as shown in Figure 4-8 right, because when the camera pose is estimated the camera is assumed to be in horizontal position in its reference system.

The strategy followed to estimate the height of the camera and, therefore, the scale factor, is depicted in the diagram of Figure 4-9. Basically, using two consecutive camera poses given by the VO part, a set of 3D road points are obtained by triangulating the 2D corners of the two associated frames. Then a plane is fitted to those points, applying the restriction of the inclination angle θ, and the distance to the *virtual camera* is calculated. Knowing it and the actual height, the scale factor and, consequently, the velocity, can be obtained.

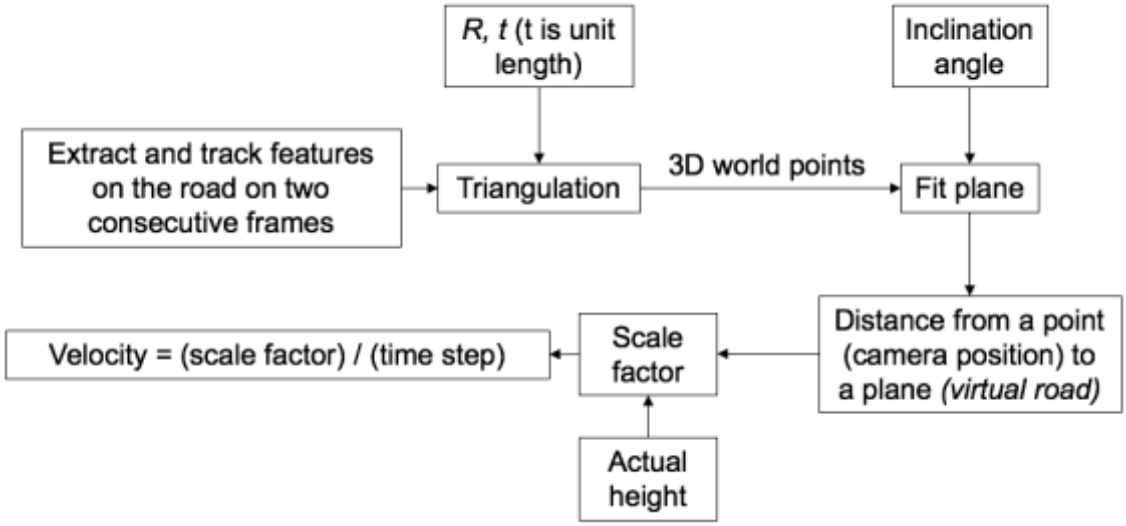

Figure 4-9 Diagram of the process of estimation of the height of the camera in order to obtain a scale factor that allows to compute an estimation of the velocity.

But some problems may arise from this approach:

- There are not always salient features on the road to be detected and tracked.
- Road points are detected with FAST corner detector by defining one or more ROIs that cover only the road but, since the ROIs that can be defined in MATLAB are rectangles, they may cover parts of the frame that is not only the road itself, but also its margins (see Figure 4-10). This can be good and bad at the same time. On the one hand, it allows to get features just from the boundary although there are not any on the road. But on the other hand, on the sides of the road may be other objects,

such as vegetation, people, etc., that are not at the same height as the road, introducing an error in the estimation.

- Sometimes, other cars are driving in the other lane, occluding the road. Additionally, points of those vehicles may be counted as road points when they are inside the corner detector ROIs.

4.3.1 Road feature detection, tracking and triangulation

As it is shown in Figure 4-10, two different ROIs have been defined in order to find features inside them, using the same methodology as in section 4.2.1.1 part but with a different value for *MinContrast*, 0.04 instead of 0.05 (to detect weaker corners). ROIs have been set up as a percentage of the image height and width to cover the road from side to side knowing the car is driving on its the left-hand side. However, not all of the road in front of the car has been included. Since the points from farther areas move just a few number of pixels from two consecutive frames originating errors in the triangulation step, they have been excluded. Hence only nearest zones of the road to the car have been included to look for corners.

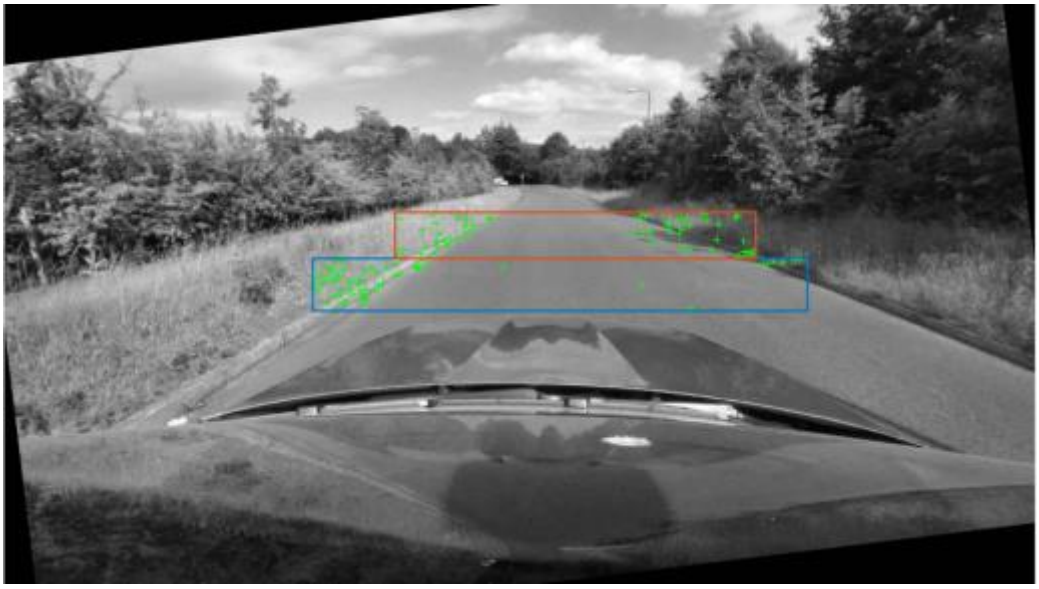

Figure 4-10 Frame from de video with two defined ROIs utilised to extract features (green crosses) only in that area.

Once the features have been computed on one frame, they are tracked along the next frames until they fall below a certain threshold, named *road feature tracking threshold* (*road threshold* for short), similarly as in subsection 4.2.2.2. The influence of this threshold is studied in section 5.4.

After the tracking part, the triangulation of the 2D points is done. To do so, the MATLAB function *triangulate*, included in the Computer Vision System Toolbox, is utilised. It needs as inputs two sets (from two images) of matched or tracked points, the extrinsic matrix of the camera in the first position and the extrinsic matrix of the camera in the second position. Since the interest of this step is to obtain a plane to get a distance and not to obtain a 3D position of every point in a global reference system, the first camera matrix is set to $[I|0]$ and the second one is

$$W = [R|t] \tag{4-7}$$

where

$$\begin{aligned} R &= R_{pose}^{T} \\ t &= -t_{pose} \cdot R \end{aligned} \tag{4-8}$$

and R_{pose} and t_{pose} are the rotation matrix and translation vector in camera coordinates relative to the previous pose obtained from the motion estimation step. Hence, the obtained 3D points will be in a reference system centred in the position of the first camera pose.

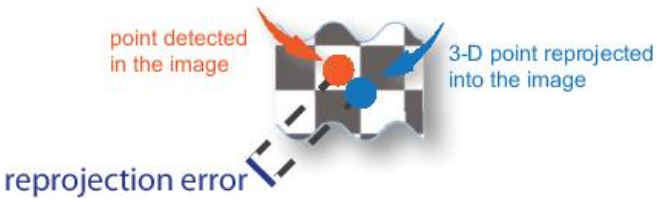

Figure 4-11 Illustration of the reprojection error [29]

Points with a reprojection error bigger than 0.5 (see Figure 4-11), those which are 50 units[2] away from the camera (in 3D coordinates) and points triangulated behind the sensor are discarded. The reason why is to use only those points with bigger possibilities of being correctly triangulated.

4.3.2 Fit a plane to the point cloud and compute the scale factor

Once all the road points have been triangulated for an image pair, a plane can be fitted since all of them should ideally be contained in the plane defined by the road. To do so, the MATLAB function *pcfitplane*, included in the Computer Vision System Toolbox, is used. Four input arguments can be given to the function:

- The 3D point cloud obtained when triangulating the road points.
- *referenceVector*: normal vector to the plane. Knowing the inclination angle is $\theta = -16.31º$, the value of this parameter has been fixed to a vector contained in the YZ plane of the camera reference system (see Figure 4-8) and with an angle of $-16.31º + 90º$, that is, $\vec{n} = (0 \quad -1 \quad -0.2927)$.
- *maxDistance*: parameter that indicates the maximum distance from an inlier point to the plane. Fixed to a value of 0.5 units.
- *maxAngularDistance*: it is the maximum absolute angular distance between the reference vector and the normal vector of the fitted plane. Its value has been chosen to be 0.5º to give a certain degree of freedom because if not, to many plane fitting operations fail.

In Figure 4-12, an example of a resulting plane is shown. There, it is possible to see how the cloud of points follows a plane of an inclination very similar to the one of the actual camera inclination (of the fitted plane). However, there may be some situations where the point cloud has a different slope because of some external disturbances such as points on cars, on vegetation or, even the plane is not computed because not enough inliers are found to find a plane that meets

[2] Note that in this context 1 unit is the distance between the two camera poses.

the specified requirements. In the last case, the result is that for that time there is no high estimation.

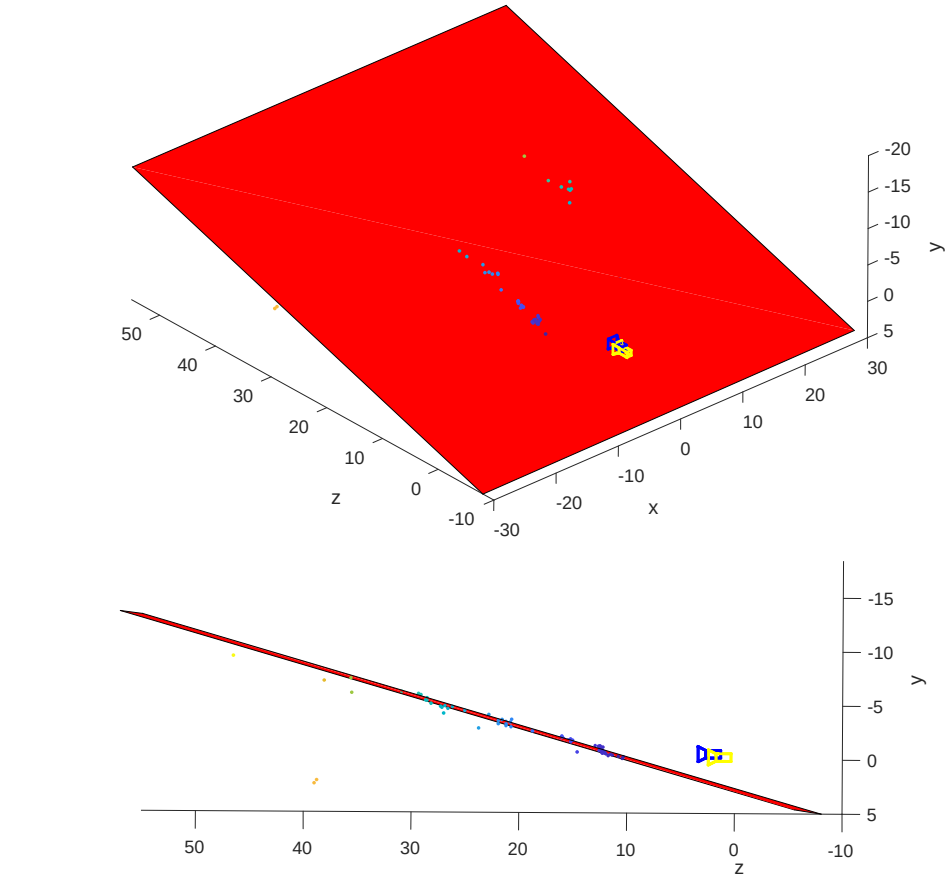

Figure 4-12 Example of a plane fitted to a cloud of points corresponding to the 2D road points (in green) of the bottom image). Units are expressed as a multiple of the separations between the two camera poses (1 unit).

Once the plane has been computed, obtaining the scale factor is straightforward. The distance from a point (camera position) to a plane (*virtual road*) is calculated

$$d = \frac{abs(A \cdot x_{cam} + B \cdot y_{cam} + C \cdot z_{cam} + D)}{\sqrt{A^2 + B^2 + C^2}} \quad \textbf{(4-9)}$$

where *A, B, C* and *D* are the coefficients of the implicit equation of the plane ($A \cdot x + B \cdot y + C \cdot z + D = 0$) and x_{cam}, y_{cam} and z_{cam} are the coordinates of the camera position.

With this, the scale factor is

$$scale\ fator = \frac{h_{actual}}{d} \quad \textbf{(4-10)}$$

where h_{actual} is the actual camera height, $h_{actual} = 1.46\,m$. And because the length of the translation vector is 1 unit, the velocity can be computed using directly the scale factor

$$v_{estimatd} = \frac{scale\ factor}{t_{step}} \quad \textbf{(4-11)}$$

where t_{step} is the time step between frames.

4.4 Conclusions

From the existing feature-based VO methods, the 2D-to-2D approach is the one that has been chosen to address the problem of velocity estimation. However, it has a problem: all the translation vectors are of unit length. That is why, together with the VO algorithm, another step is done to obtain a scale factor knowing that the height of the camera is constant.

So, after correcting the video frames (lens distortion and frame rotation), the corners of the first frame are extracted using the FAST feature detector and then are tracked to the following images using the KLT tracker. Since the number of corners will decrease with new incoming images and, eventually, disappear, a redetection step is done when they fall below a certain threshold.

At the same time, a road feature detection and tracking is done to find salient points that are only on the road and its margins.

From the corner correspondences between two consecutive frames the relative camera pose is obtained by firstly estimating the essential matrix (using the five-point algorithm together with RANSAC to discard outliers), but the computed translation vector is of unit length. Then, knowing the relative position between the two camera poses, the 3D position of the road points is obtained by triangulation and a plane is fitted to them that will represent the *virtual road*. The distance from that plane to the camera position is computed and the scale factor is obtained and utilised to rescale the unit length translation vector of the camera movement and to estimate the velocity of the vehicle between the two correspondent frames.

5 ANALYSIS OF THE RESULTS

In this chapter, the explained algorithm in Chapter 4 is analysed in terms of accuracy of its output measurements. A brief explanation of the recorded data is presented in section 5.1. Then, in section 5.2, how to get a smoother velocity profile than the raw output is explained. After that, the algorithm is tested for four different frame tracking thresholds (section 5.3) and four road tracking thresholds (section 5.4) to see their influence in the final results and select one to continue with the next analysis. Finally, the algorithm is run for nine combinations of spatial and temporal resolution to see their influence on the estimations (section 5.5).

It is worth to mention that the obtained results may not be the optimal solution as many parameters contribute to the final estimation and many different combinations exist. Additionally, the objective of this thesis is not to obtain an optimal tool to estimate the velocity but to prove it is possible to get good results.

Moreover, all the studies, until section 5.4 included, have been done using a smaller spatial and temporal resolutions of the video data (960x540 pixels and 15 fps) in order to not lose much time, since analysing 1 minute of video at these resolutions takes around 15 minutes.

5.1 Description of the recorded data

The available data has been recorded by RACELOGIC and consists in a 5-minute video with the corresponding synchronised GPS data. In this section, a brief description of the video is given, utilising some of the GPS data to help in that task.

First of all, Figure 5-1 shows the path the vehicle followed while gathering the data. As it can be seen, it is characterised by a long quasi-straight road, where the car was driven twice (blue and yellow lines), and a shorter path with six 90° curves (orange line in Figure 5-1 and Figure 5-2). This allows to split the whole trajectory in three individual and shorter sections:

- Section 1 (from second 80 to 153[3]): section characterised by being a quasi-straight road with some traffic on the opposite lane and velocities between 21 and 42 km/h.
- Section 2 (from second 160 to 220[3]): section characterised by having six sharp curves (see Figure 5-2 for a more detailed view) and several static vehicles close to the car on both sides of the road between curves 3 and 4, and by velocities between 8 and 27 km/h. This section is the only one in which part of the road has road markings. The times at which the car is turning are:
    - Curve 1: 162 – 166 s.
    - Curve 2: 168 – 171 s.
    - Curve 3: 180 – 185 s.
    - Curve 4: 191 – 197 s.
    - Curve 5: 203 – 207 s.
    - Curve 6: 217 – 221 s.
- Section 3 (from second 234 to 284[3]): section characterised by being a quasi-straight road (the same one as section 1 but in the opposite direction) with the especial characteristic of reaching the highest speed of the dataset, 75.38 km/h. That is why this video section will be referred as the *high-speed section*. Velocities are between 20 and 75.38 km/h.

In addition to these three sections, there is more footage but is not relevant to the analysis. Before the first section the car is stopped during almost one minute and then it leaves a car park (footage utilised in section 3.2.2 to obtain the radial distortion coefficients).

Finally, the GPS velocity has had to be resampled to match the video footage frequency from 10 to 30 Hz.

[3] The time intervals utilised here, and in the following pages and plots of this chapter, refer to the video time.

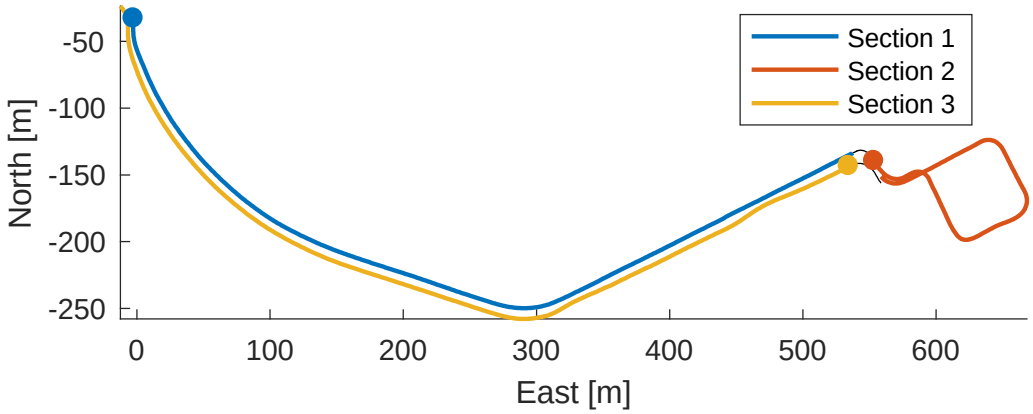

Figure 5-1 Trajectory followed by the car in the three sections in which the video data has been divided. Dots indicate the initial position of the car in each section.

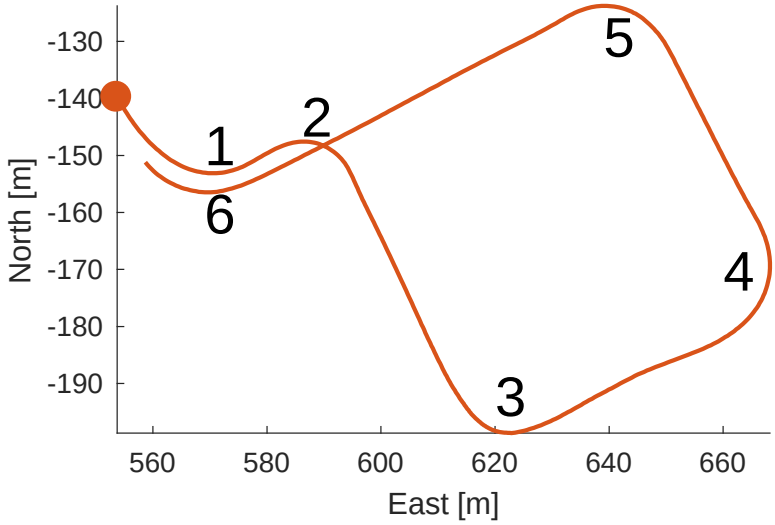

Figure 5-2 Detail of the track of video section 3 with its six curves numbered in order of appearance.

5.2 Smoothing the raw VO output

The velocities estimated directly from the algorithm presented in Chapter 4 (from now on *raw velocities*/*output*/*results*) are quite noisy, as shown in Figure 5-3 for the three video sections, but follows a trend very close to the ground truth. The origin of this noise is probably that the *virtual plane* estimations are not always good, depending on the quality of the detected road corners and their tracked positions. That is because the road, in general, has few features and the road detector has been set up with a low contrast level so the found

points may not be very good, and therefore, they may be wrongly tracked sometimes. Additionally, the raw outputs have velocity peaks that are unreal because of the huge accelerations that should occur to reach them. The utilised configuration to obtain the mentioned figure, and the ones utilised in section 5.2, are the ones obtained after selecting both tracking thresholds in sections 5.3 and 5.4 and at 15 fps and 960x540 pixels. They are used here because this is a previous step and to show how the raw VO velocities must be processed to obtain a better estimation than the raw output itself. To do so, two tasks are done after obtaining the raw results:

- Use a moving average filter to smooth the raw output eliminating the noise.
- Establish an acceleration limit to discard those raw measurements that are much bigger than the previous ones, that is, those with large accelerations.

5.2.1 Moving average filter

In order to eliminate, or at least reduce, the noise of the raw velocity, a moving average filter has been utilised. The size of the filtering window is important. Small windows are good to capture sudden and rapid changes in the velocity but can also give wrong results if those changes are due to noisy data. On the other hand, bigger windows help to eliminate the latter but at the cost of not following well real changes in the velocity. That is why a window length of 1 second forwards and 1 second backwards, that is, 20, 30, and 60 samples in total for 10, 15 and 30 fps respectively, has been chosen and, unless the contrary is specified, it is the window size utilized from now on. As it is shown in section 5.2.2, after setting an acceleration limit to remove the velocity peaks, the selected window size works well.

The results after moving averaging the raw output, shown in Figure 5-3 in black, are much better than he noisy raw velocity, although there are still some velocity peaks, but not as big as before, and also some points where the smoothed velocity differs from the actual one. In the next subsections, these discrepancies are explained.

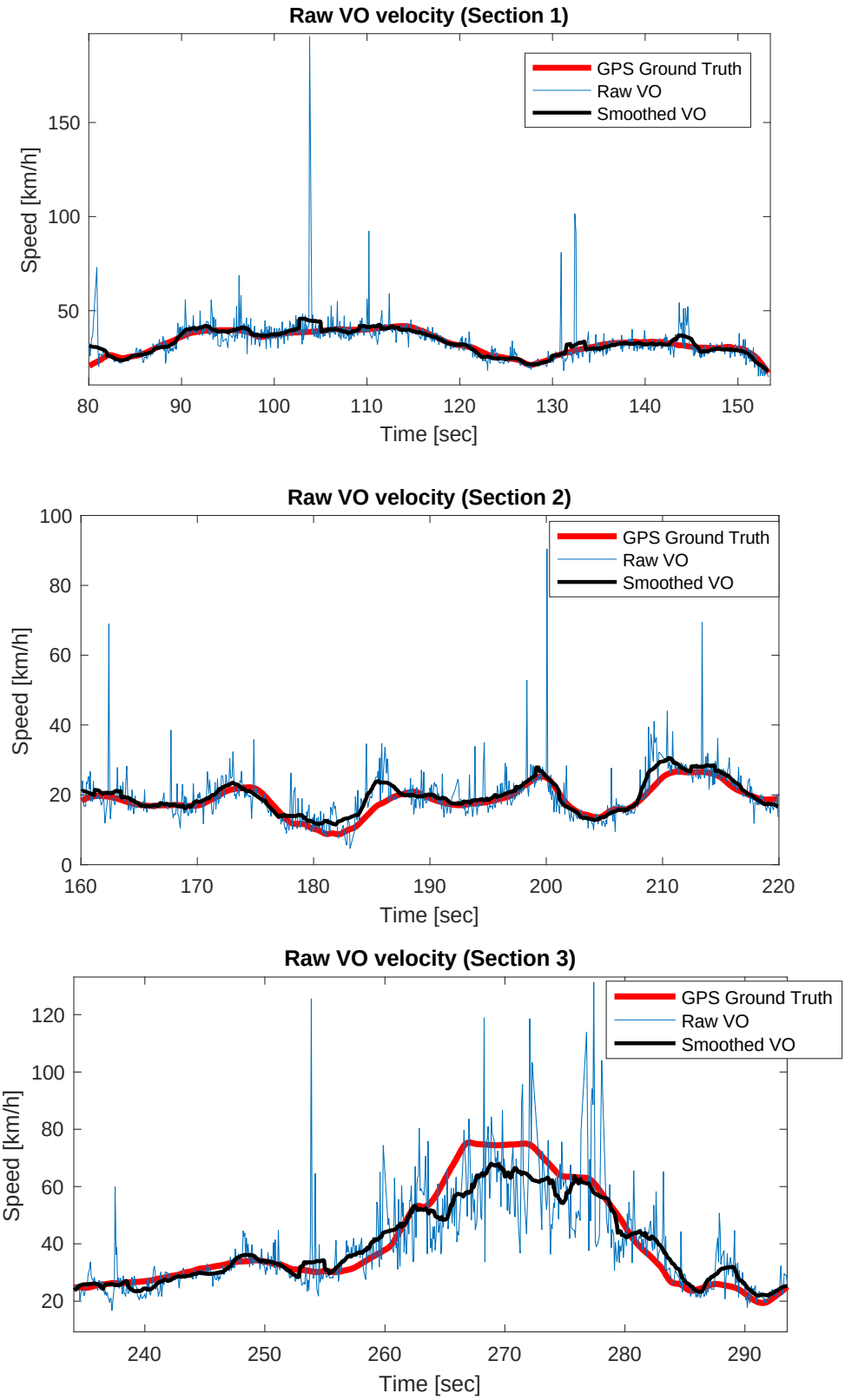

Figure 5-3 Comparison of the raw VO velocity with its smoothed version and the GPS ground truth in the three video sections. Resolutions: 960x540 pixels and 15 fps.

5.2.1.1 Discrepancies in video section 1

In Figure 5-4 the main discrepancies that appear after smoothing the raw velocity of video section 1 are highlighted. It is remarkable that most of them are due to other vehicles driving on the same road but, in this case, only on the opposite lane. These peaks happen because, when a car is inside the ROIs defined to detect road features, some of them will be on that car and they are above the actual road plane (see Figure 5-5 as an example). This, together with the fact that cars are moving (they introduce an error in the triangulation of those points) leads to compute a wrong plane that is above the real one and, therefore, the distance to the camera is smaller, resulting in a bigger scale factor and, consequently, a bigger estimated velocity. Additionally, the covered road has no road markings and just few shadows so the majority of the detected corners actually correspond to its margins where there is some vegetation that also increase the estimated height of the plane (see Figure 5-5). Then, there is another part where the lack of accuracy is due to another factor. Between the seconds 90 and 95, orange circle in Figure 5-4, there is an area with not many road features and where most of them are only on the right-hand side of the road, as shown in Figure 5-6.

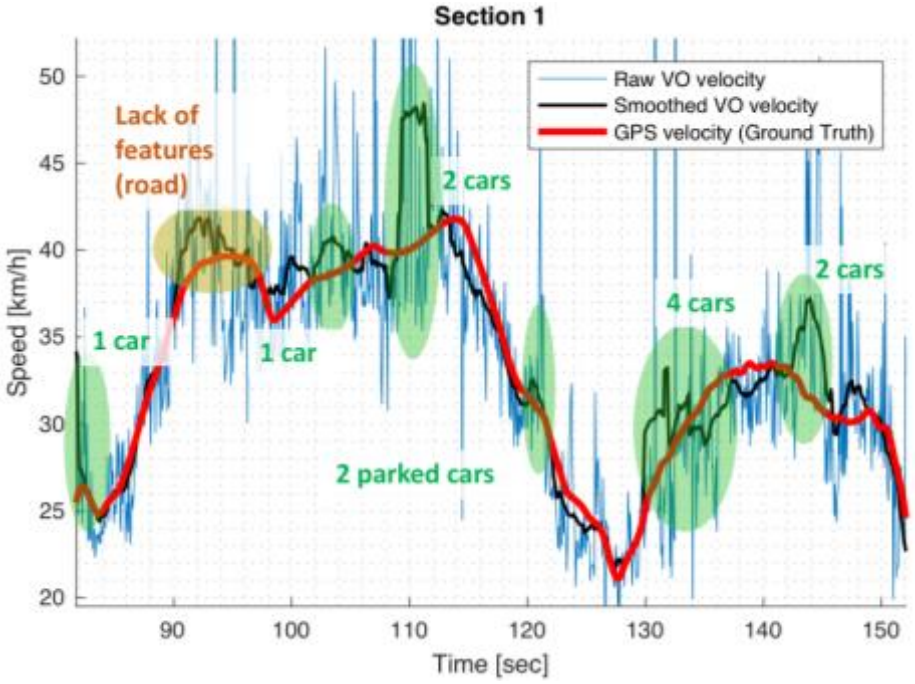

Figure 5-4 Velocity profile of section 1 after smoothing the raw VO output. In green, parts of the video where other cars appear on the opposite lane or parked

are highlighted. In orange, a zone where there are few corners on the road is also highlighted. Resolutions: 960x540 pixels and 15 fps.

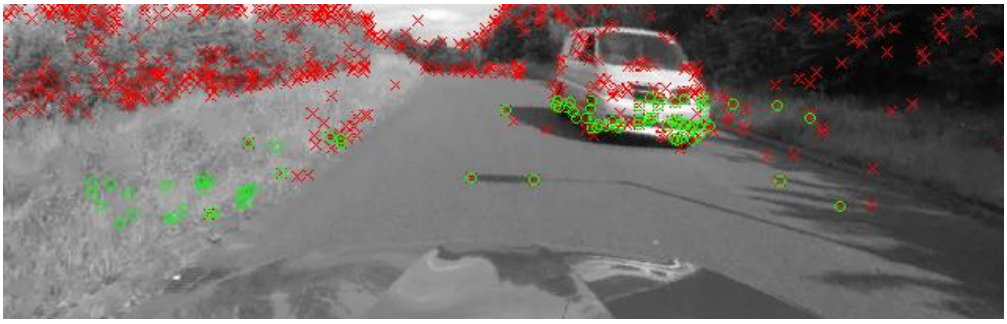

Figure 5-5 Part of a frame corresponding to the part highlighted in green in the second 104 in Figure 5-4 where road corners are represented with green circles. In this specific example, the majority of the road points are not on the road but on the car and on the lateral vegetation.

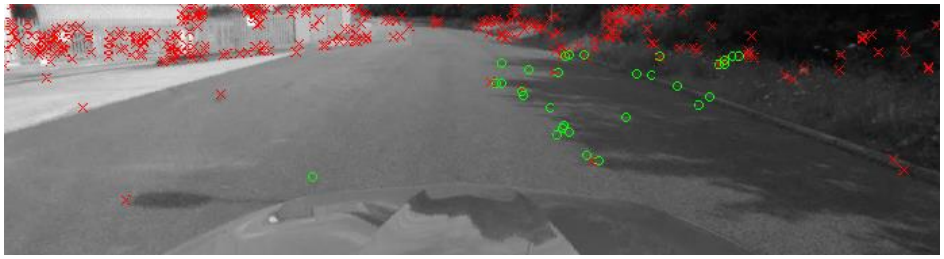

Figure 5-6 Part of a frame corresponding to the part highlighted in orange in Figure 5-4 (video time: 90 – 95 sec) where road corners are represented in green.

5.2.1.2 Discrepancies in video section 2

In the section 2 of the video, characterised by having six 90º curves, similar problems to the ones of section 1 occur. Again, other vehicles cause wrong estimations, as shown in Figure 5-7. In the first and third curves, from 162 to 166 seconds and from 180 to 185, other cars appear just in front of the camera, producing a similar effect as those described for video section 1. The main difference between these two curves is that in the latter the other vehicles are parked on both sides and very close to the camera, as shown in Figure 5-8.

Finally, the last remarkable point of this part of the video where the measurements fail is between 208 and 213 seconds. In this occasion, there is a big bush on the left-hand side, shown in Figure 5-9, that again makes most of the road points to not be on the road.

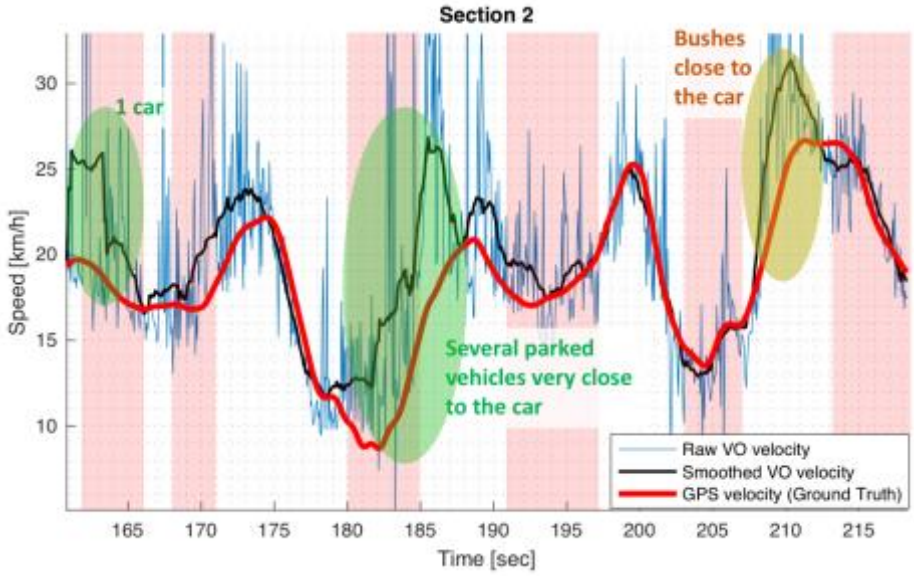

Figure 5-7 Velocity profile of section 2 after smoothing the raw VO output. In green, parts of the video where other cars appear on the opposite lane or parked are highlighted. In orange, a zone where there are bushes very close to the car is also highlighted. Finally, red rectangles point out parts of the video where the car was turning. Resolutions: 960x540 pixels and 15 fps.

Figure 5-8 Part of a frame corresponding to the part highlighted in green in Figure 5-7 at video time 180 sec, where road corners are represented in green.

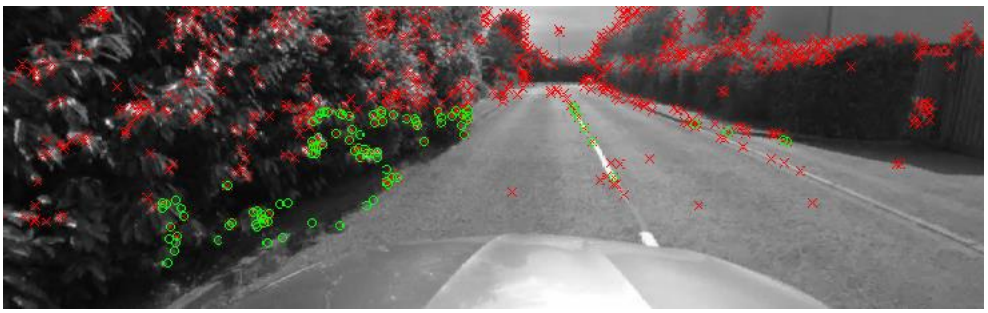

Figure 5-9 Part of a frame corresponding to the part highlighted in orange in Figure 5-7 (video time: 208 and 213 sec) where road corners are represented in green.

5.2.1.3 Discrepancies in video section 3

Finally, in video section 3, similar things happen, highlighted in Figure 5-10. In this case, the most remarkable dissimilarity between the smoothed and the actual velocities is the part from 265 to 277 sec where the car is moving at more than 50 km/h. This is the fastest section of the video and the measurements are very noisy and apparently fail because of the speed. It is the same track than from 90 sec to 112 of video section 1 (Figure 5-4), but returning. In section 1 the raw measurements are also noisy (including the effect of other cars) but not as much as in this situation (where there are no cars in this period of time).

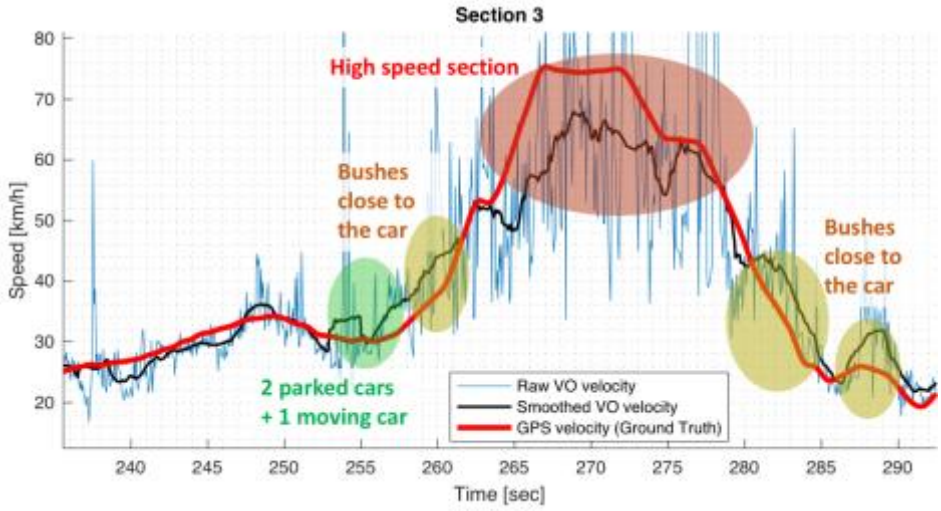

Figure 5-10 Velocity profile of section 3 after smoothing the raw VO output. In green, parts of the video where other cars appear. In orange, zones where there are bushes very close to the car are also highlighted. Finally, in red, a section where the car is driving fast. Resolutions: 960x540 pixels and 15 fps.

5.2.2 Acceleration limit

The raw measurements are processed before doing the smoothing in order to remove those peaks as much as possible. To do so, the selected technique consists in fixing an acceleration limit and discard those speeds that exceed the limiting speed change when compared with the previous valid velocity.

Figure 5-11 shows the results after applying this limit and smoothing the raw data for five different acceleration thresholds: 10, 20, 30, 40 and 50 km/h/s. The first one could be a normal acceleration for a regular car as it is the case; the

next two, and even the fourth, could be the ones of a sport car; 50 km/h/s, however, would be too much for a car (from 0 to 100 km/h in just 2 sec). Nonetheless, the velocities to which these limits are applied are the ones of the raw VO output, that is, their strong accelerations are due to its noisy nature. Hence, although those thresholds may be not very realistic, are just utilised to eliminate the biggest accelerations and keep those that follow a trend.

In general, all of the them give similar velocity profiles, as might be expected since all of them use the same raw output. However, some works better than others:

- 40 and 50 km/h/s limits do not work well in points where, as shown in Figure 5-4, Figure 5-7 and Figure 5-10, there are discrepancies with the ground truth: in seconds 92, 145 (section 1), 186, 211 (section 2) and from 265 to 280 (section 3). Nevertheless, the rest of the differences have been greatly removed.
- The 10 km/k/s limit also works well but at certain points it behaves worse than the others, since it does not follow well the actual velocity profile: seconds 92, 128 (section 1), 186, 203 (section 2) and, again, from 265 to 280 (section 3).
- Finally, the 20 and 30 km/k/s limits are very similar. However, the first one follows better the actual velocity although both of them also misestimate the actual velocity from 265 to 280 (section 3).

Having a look at Figure 5-12, where the maximum and mean errors and standard deviations[4] are plotted against the acceleration limits, it is possible to see that effectively the 20 km/h/s threshold has the minimum maximum absolute and relative errors, especially in the first two sections and also has one of the smallest maximum and mean standard deviations.

However, all of them work well, having errors below 3 km/h in the first section and below 6 km/h in the second. In terms of mean and RMS errors all the limits

[4] Standard deviations are computed using the method of moving standard deviation, utilising the same window size as for the moving average filter.

are small indicating that, as shown in Figure 5-11, they follow well the ground truth velocity and the measurements are accurate (mean errors very close to 1 km/h in sections 1 and 2). A more detailed view of the evolution of the absolute and relative errors is shown in Figure A-1 and Figure A-2 of the Appendix A.

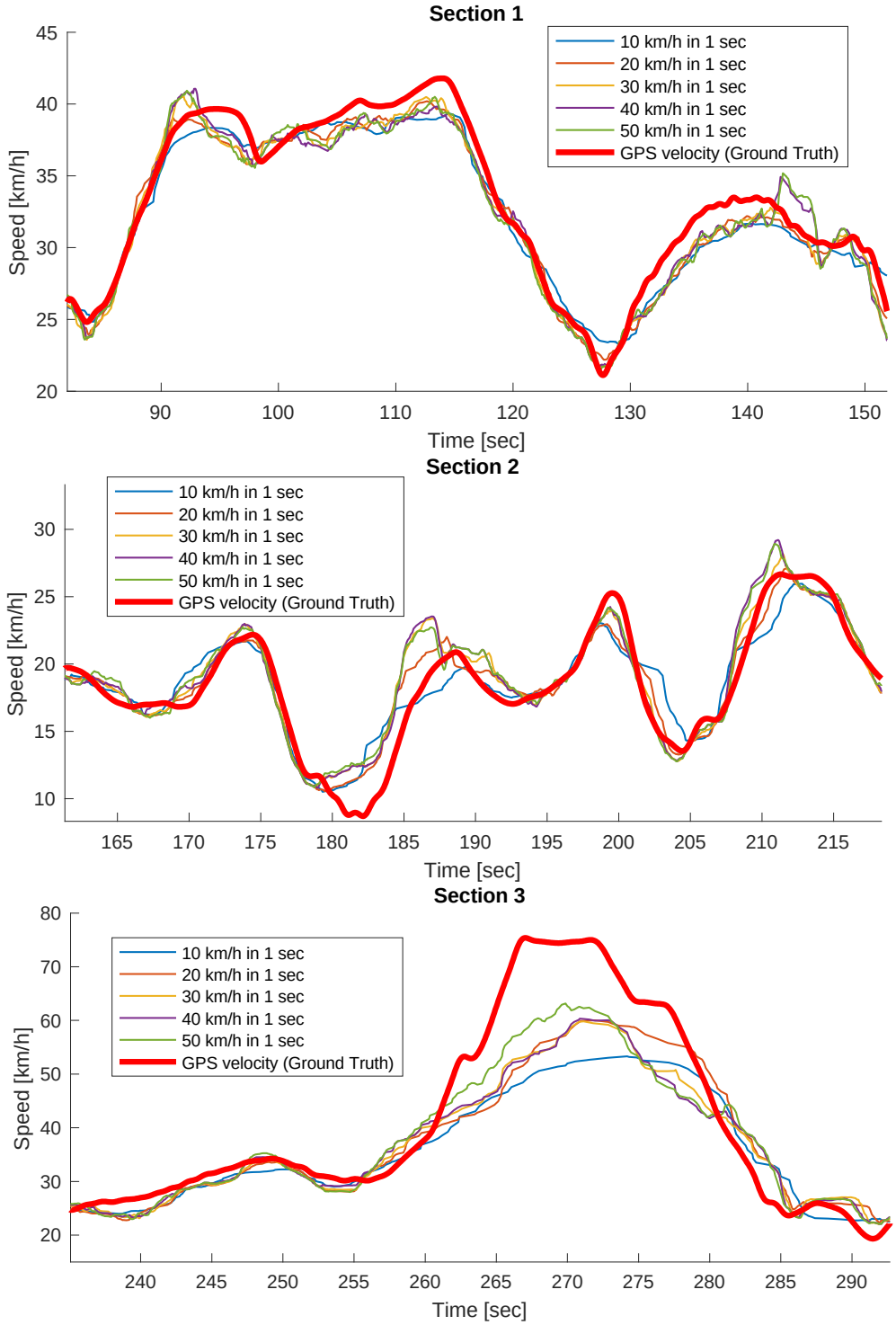

Figure 5-11 Velocity profiles of the three sections for five different acceleration limits. Resolutions: 960x540 pixels and 15 fps.

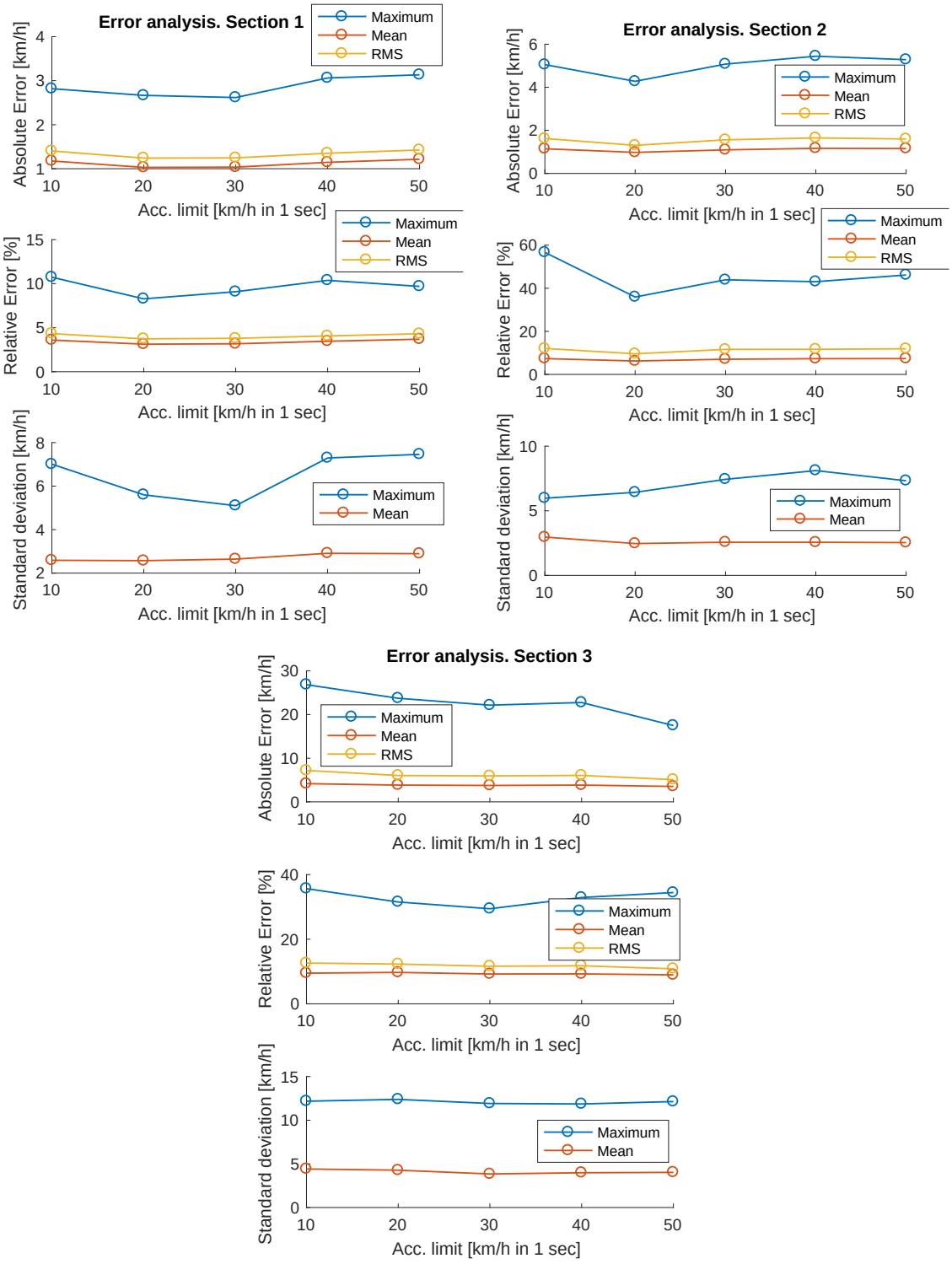

Figure 5-12 Error analysis of the three sections for five different acceleration limits. Resolutions: 960x540 and 15 fps.

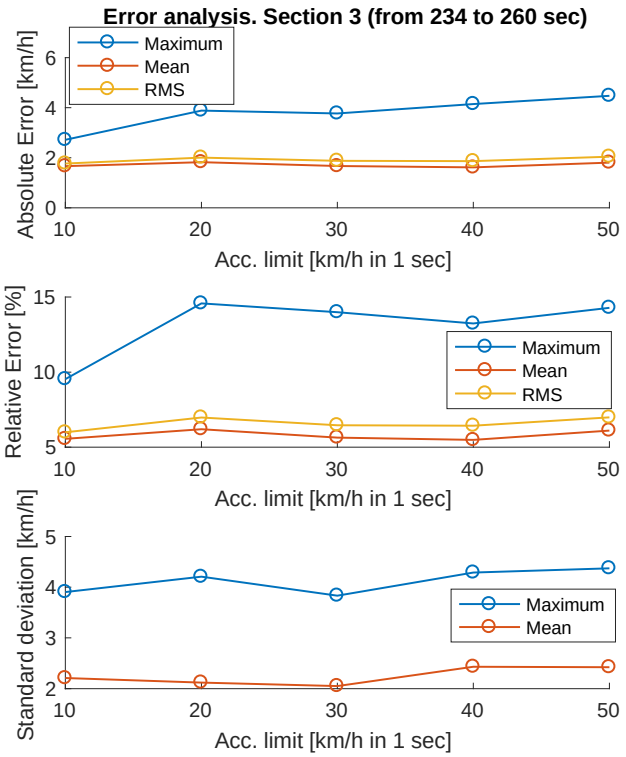

Figure 5-13 Error analysis of section 3 (from 234 to 260 sec) for five different acceleration limits. Resolutions: 960x540 and 15 fps.

Section 3 is another thing. The problem in this case is the maximum errors and standard deviations are located in the high-speed part and are very big. But if the rest of the section is analysed independently, as it is shown in Figure A-3 and Figure 5-13, it can be seen that in general the absolute error is less than 4 km/h for the five acceleration limits with a maximum relative error of 15% before starting the fastest part.

With all of the above, for the next analyses the acceleration limit of 20 km/h/s is utilised. Figure 5-14, Figure 5-15 and Figure 5-16 show the above results but only for this threshold and with their moving standard deviations superimposed. It is noticeable how the ground truth is, for almost all the analysed dataset, inside the boundaries defined by the standard deviation of the estimated velocity. The most outstanding parts where it is outside are in second 182, the place where the car is turning and has some parked cars very close to it (see Figure 5-8), and the high-speed section.

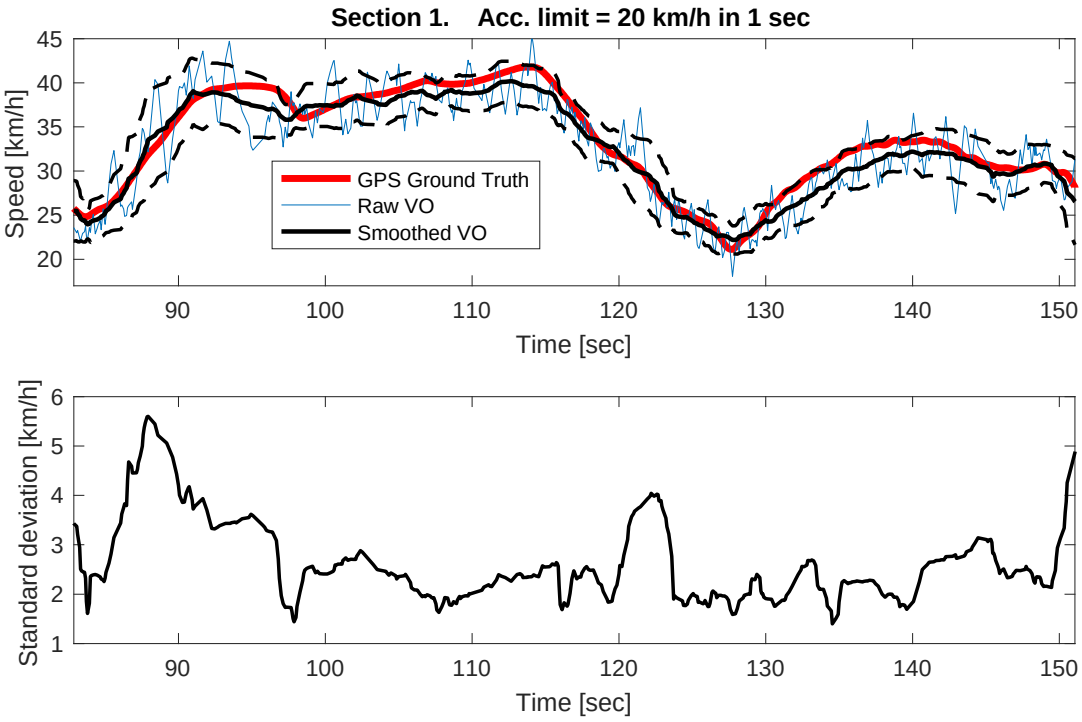

Figure 5-14 Velocity estimation in section 1 from VO using a video of 960x540 pixels and 15 fps and an acceleration limit of 20 km/h/s. Dashed lines correspond to the moving standard deviation limits.

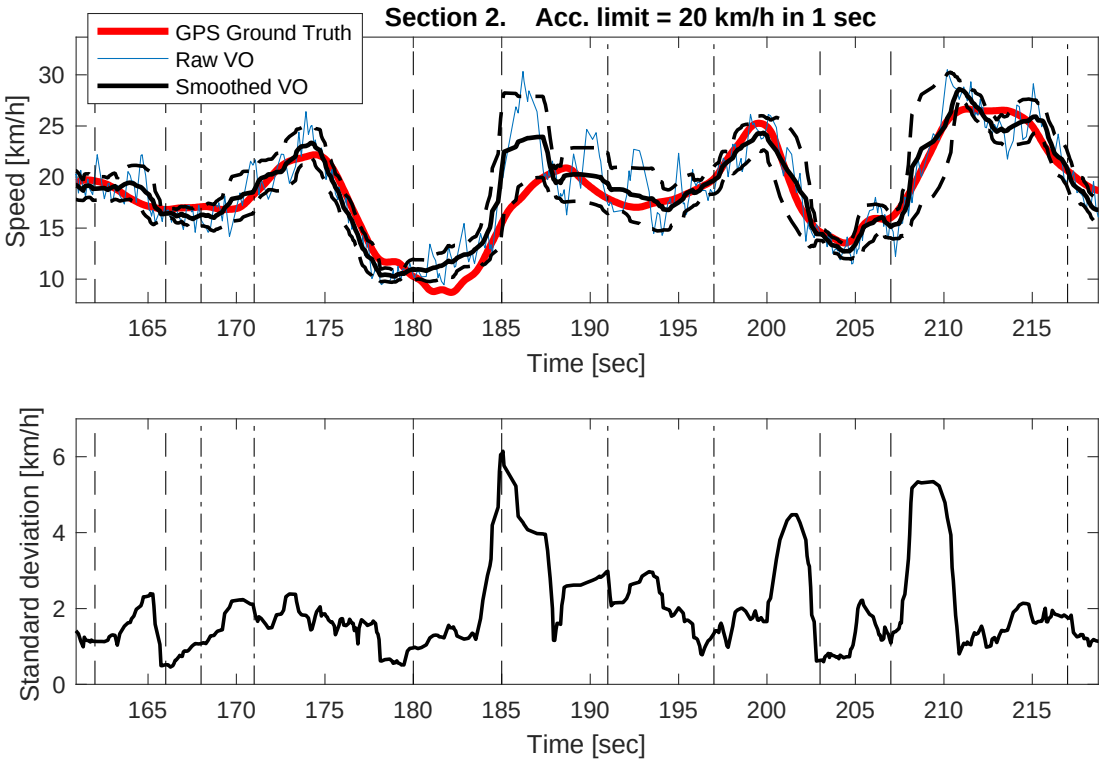

Figure 5-15 Velocity estimation in section 2 from VO using a video of 960x540 pixels and 15 fps and an acceleration limit of 20 km/h/s. Dashed lines correspond

to the moving standard deviation limits. Vertical dashed and dash-dot line pairs indicate each one of the six 90º curves (Note that the last one does not finish inside the plot).

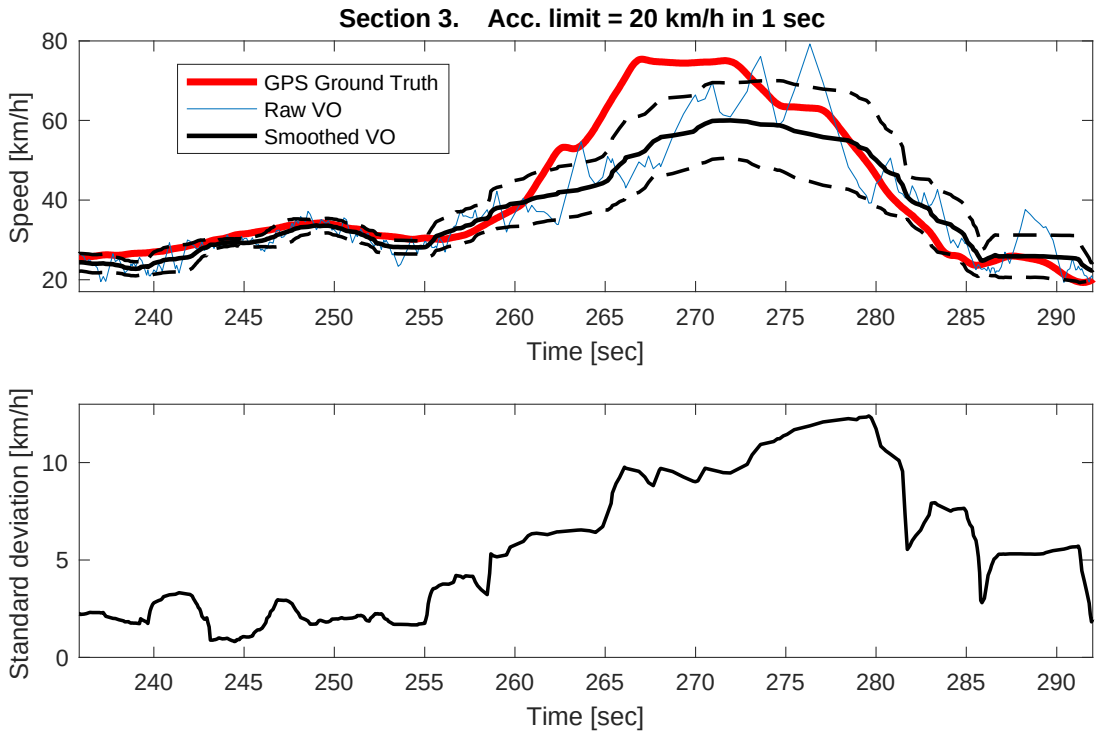

Figure 5-16 Velocity estimation in section 3 from VO using a video of 960x540 pixels and 15 fps and an acceleration limit of 20 km/h/s. Dashed lines correspond to the moving standard deviation limits.

Then, there are few moments where the measurements are not very precise, that is, where the standard deviation reaches big values:

- Section 1 (Figure 5-14): from the second 83 to 95 is where the maximum deviations are located for this sector. It corresponds to a moment in which the car is accelerating and also when it is in the place shown in Figure 5-6 and explained in 5.2.1.1. Then around the second 122 there is another peak of the standard deviation, again in a moment where the car was changing its speed (decelerating in this case).
- Section 2 (Figure 5-15): curves themselves do not cause big standard deviations or errors but the objects that are close to the car while turning. The two biggest standard deviation peaks are in places where the car had other objects close to it: other vehicles in second 185 (see Figure

5-8) and a bush in second 209 (see Figure 5-9), both explained in 5.2.1.2. Both of them, plus the one in 201 sec, also correspond to moments in which the car is changing its speed.

- Section 3 (Figure 5-16): in this case the biggest standard deviations correspond to the moments in which the car is moving at high speeds and at the same time accelerating or decelerating.

5.3 Influence of the frame feature tracking threshold

Taking into account that the maximum number of corners detected by the FAST feature detector is 3500 (using the configuration explained in section 4.2.1.1) the frame threshold must be less or equal than 3500.

An analysis has been made to see the influence of the frame threshold on the results and to select one. To do so, the three sections of the video has been utilised using the next configuration:

- The FAST feature detector configuration explained in section 4.2.1.1 for frame features and the one in section 4.3.1 for road corners.
- A threshold for the road feature tracking of 30 corners (it is explained in section 4.3.1)
- A resolution of 960x540 pixels at 15 fps
- An acceleration limit of 20 km/h in 1 second.
- A moving average filter window of 1 second forward and 1 backward (30-samples size).

Figure 5-17 and Figure 5-18 show the results for the four chosen frame thresholds: 25%, 50%, 75% and 100% of the maximum possible number of corners, that is, 875, 1750, 2625 and 3500 corners respectively. The last one means that at every frame the number of detected corners will be below the threshold and, therefore, the redetection step is executed for each new image.

In Figure 5-17, it can be seen how the four velocity profiles are in general quite similar to the ground truth, especially in section 1. However, there are some points where they differ more:

- Between the seconds 90 and 95 (video section 1), the curve of 3500 corners has a worse performance than the other three, but very small.
- In video section 2, between 193 and 197 seconds, the curve of 3500 corners stands out because it does not follow well the actual velocity.
- As seen before, in video section 3, between 265 and 280 seconds, all of the estimations fail, with a maximum error of around 36 km/h, in the case of 875 corners, and around 20 km/h for the other three thresholds.

Figure 5-18 shows how the frame threshold of 1750 corners has the smallest maximum errors, both absolute and relative (below 3 km/h and 10%), in video section 1. There, the standard deviation is the second smallest. In terms of the average or RMS errors there is no much difference between using one threshold or another, being all of them between 1 and 2 km/h (4 - 6%). However, the one that has a smaller standard deviation corresponds to 2625 corners. In video sections 2 and 3 the maximum errors correspond to the threshold of 875 corners, whereas the other three have similar values (4 km/h in section 2 and 20 km/h in section 3). Finally, because of the big standard deviation and errors of video section 3, compared with the other two, caused by the high-speed part, this section has been shortened to take into account only its first part, until before the high-speed zone. Its error analysis is shown in Figure 5-19, having a performance similar to the other two cases.

Because all of the exposed above, the final choice is the threshold of 1750 corners because the errors, both absolute and relative, are the smallest (especially the maximums) in video section 1 where the maximum standard deviation is the second smallest. In section 2, although the maximum standard deviation is the worst one of the four alternatives, they are quite similar and in terms of errors it is a good option. Finally, in the first part of section 3 the smallest maximum standard deviation also corresponds to 1750 corners and the errors are between the two best ones.

However, it can be concluded that the frame feature tracking threshold does not have a big influence on the results, at least for the tested range, although some small discrepancies may arise at some points.

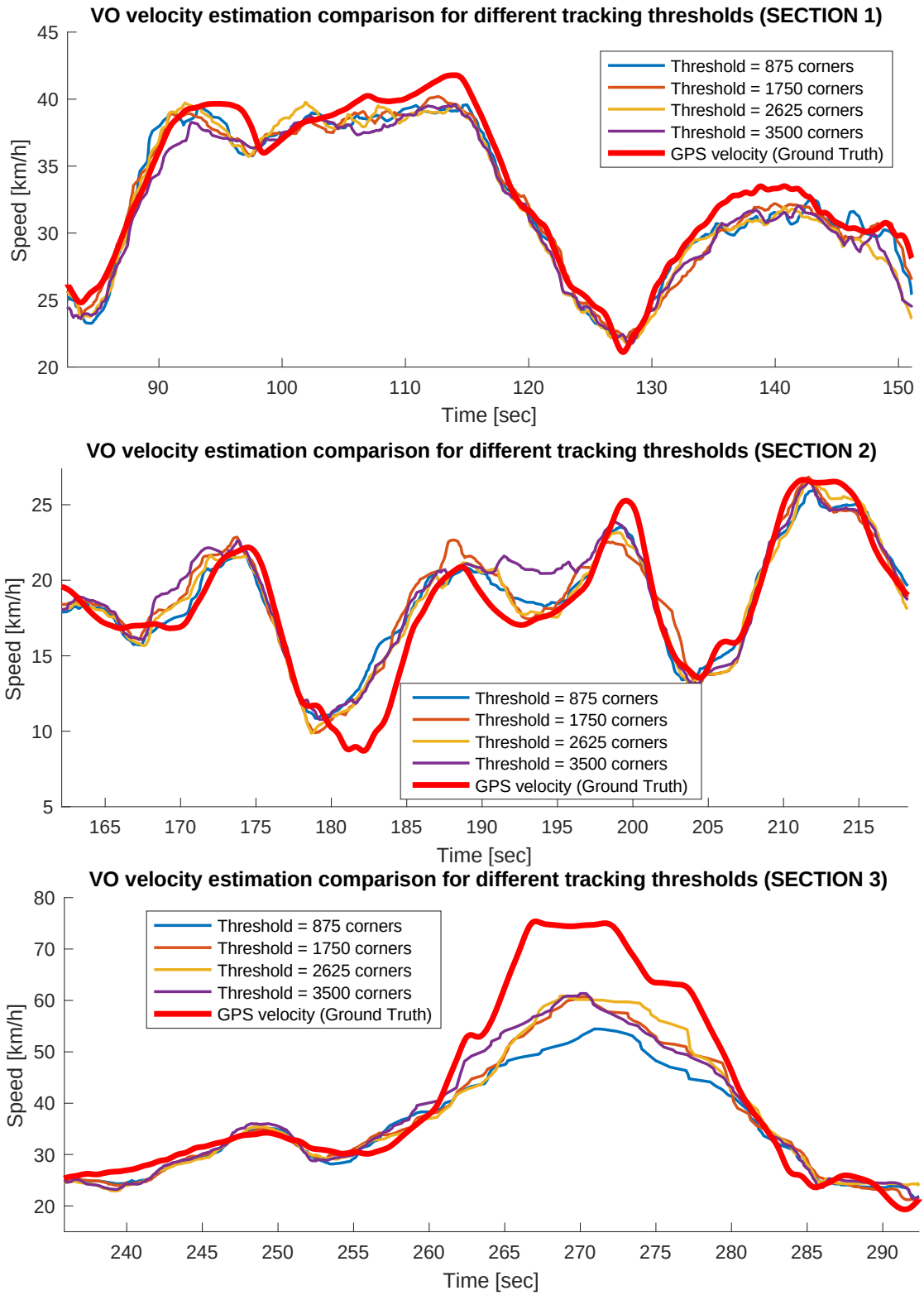

Figure 5-17 Ground truth velocity (in red) compared with the estimations obtained from VO with four different frame tracking thresholds. Resolutions: 960x540 and 15 fps.

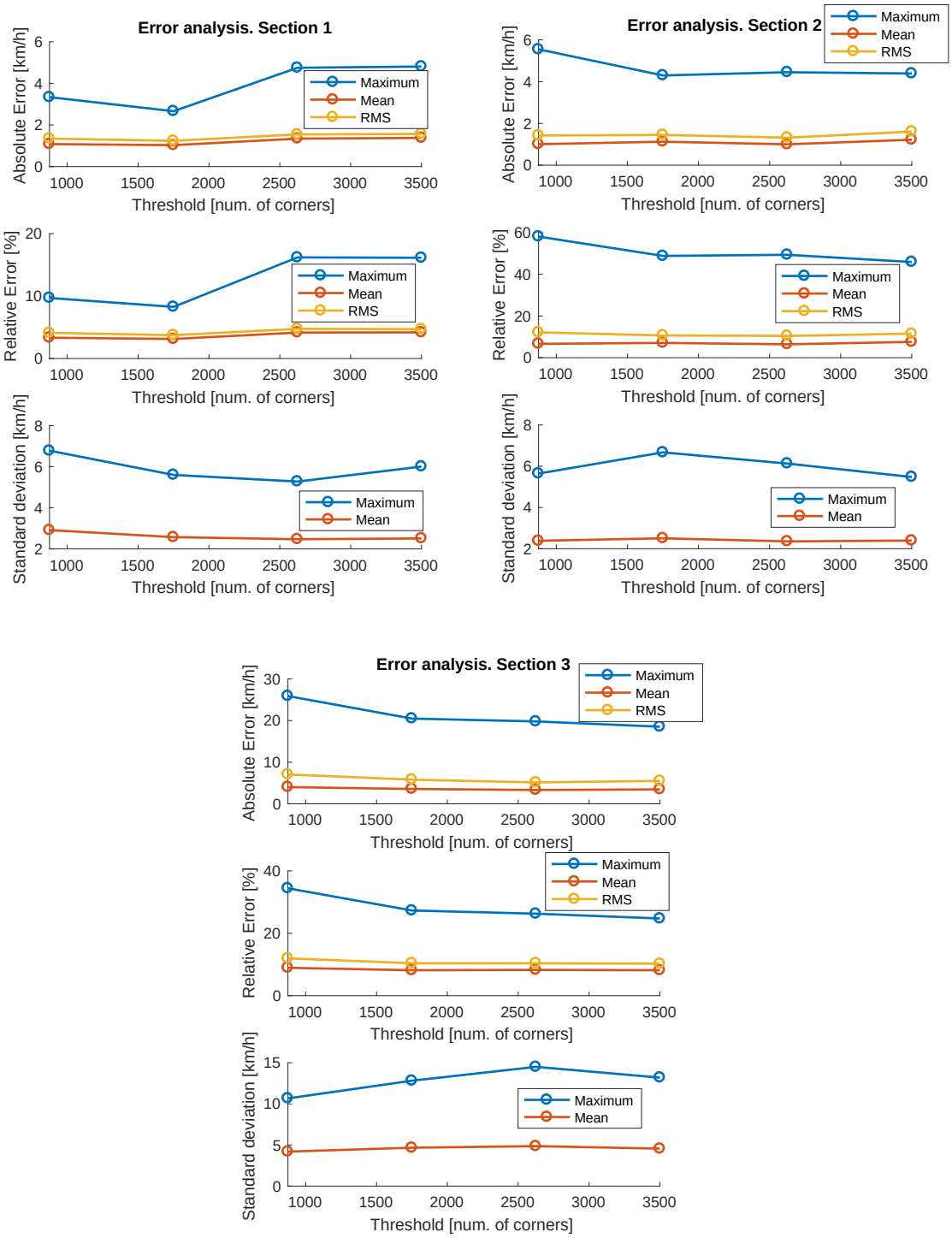

Figure 5-18 Errors as a function of the frame feature tracking threshold (number of corners). Resolutions: 960x540 and 15 fps.

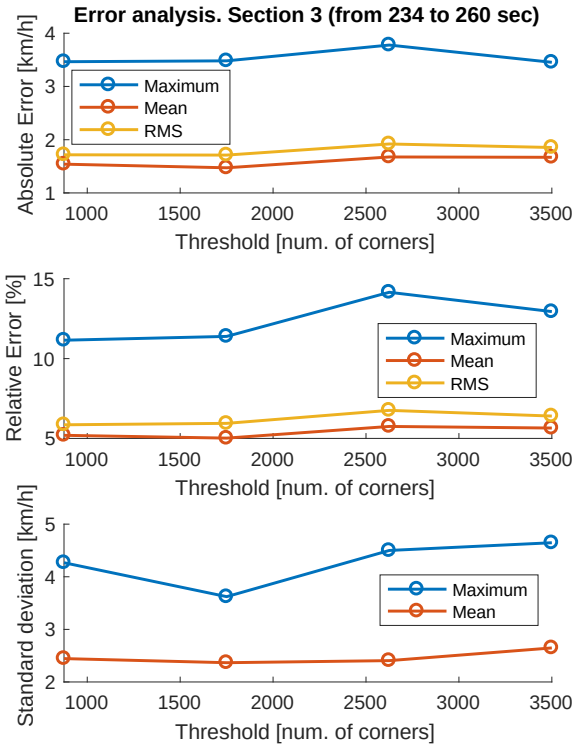

Figure 5-19 Error analysis of section 3 (from 234 to 260 sec) for four different frame feature tracking thresholds. Resolutions: 960x540 and 15 fps.

5.4 Influence of the road feature tracking threshold

Once the frame feature tracking threshold has been chosen, 1750 corners, a similar analysis is done with the road feature tracking threshold to see whether it has a big influence in the final results or not, and to select one for the next analysis (section 5.5).

Once again, the configuration of the algorithm is the same one as in section 5.3 but with the difference that now the frame tracking threshold is fixed and the road tracking threshold is modified. Taking into account that the maximum number of corners detected by the FAST feature detector is 120 (using the configuration explained in section 4.3.1) the road feature threshold must be less or equal than 120.

Figure 5-20 and Figure 5-21 show the results for the four chosen frame thresholds: 25%, 50%, 75% and 100% of the maximum possible detected road corners, that is, 30, 60, 90 and 120 corners respectively. First of all, the big similarity between the results stands out once again, both in the velocity profiles

itself and in the error analysis indicating that this parameter does not affect the final results very much. But a value is needed for further analysis, so the threshold of 60 corners is chosen because it has the smallest errors in sections 1 and 2 with a standard deviation similar to the other options in section 1.

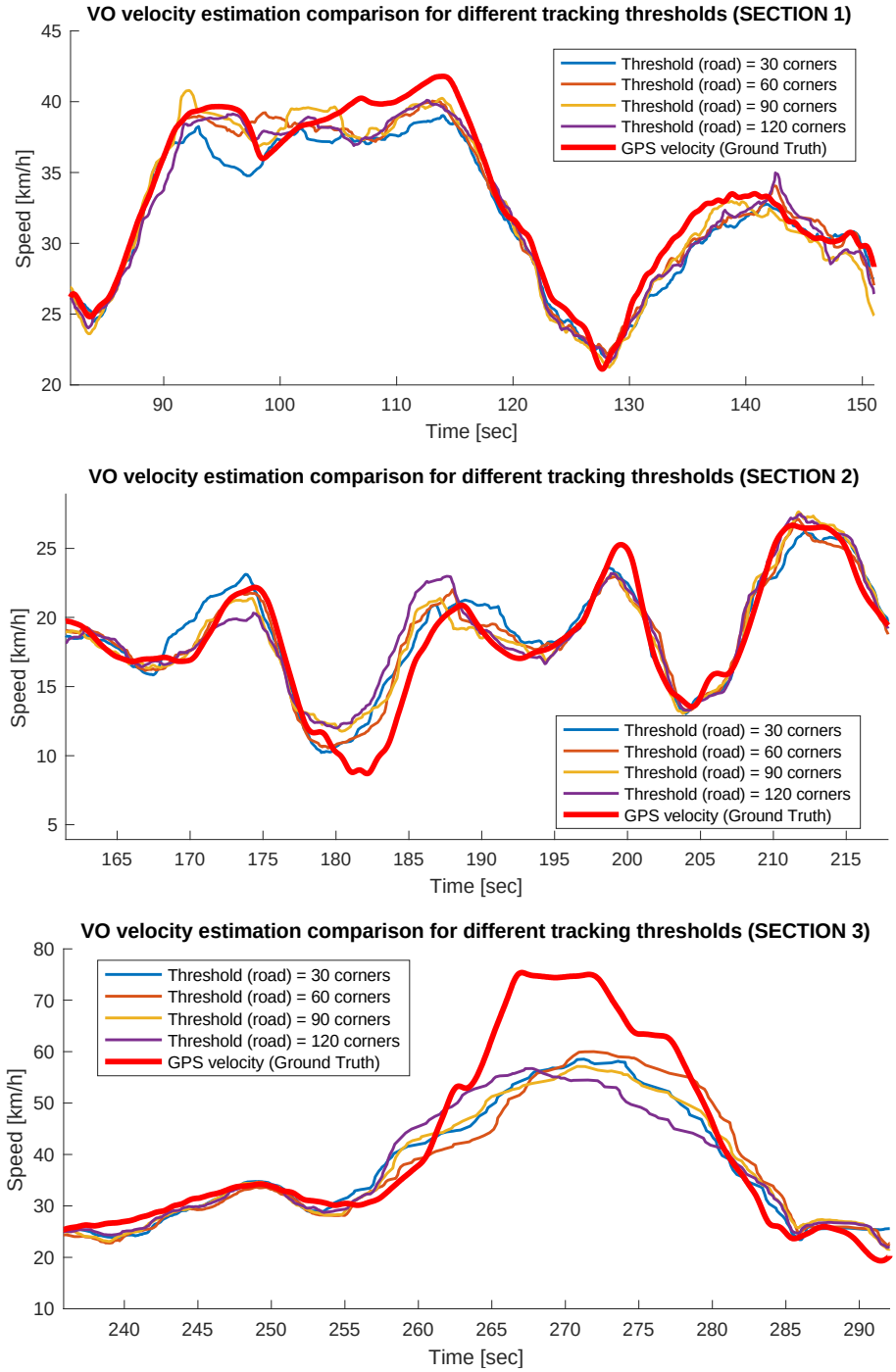

Figure 5-20 Ground truth velocity (in red) compared with the estimations obtained from VO with four different road feature tracking thresholds. Resolutions: 960x540 and 15 fps.

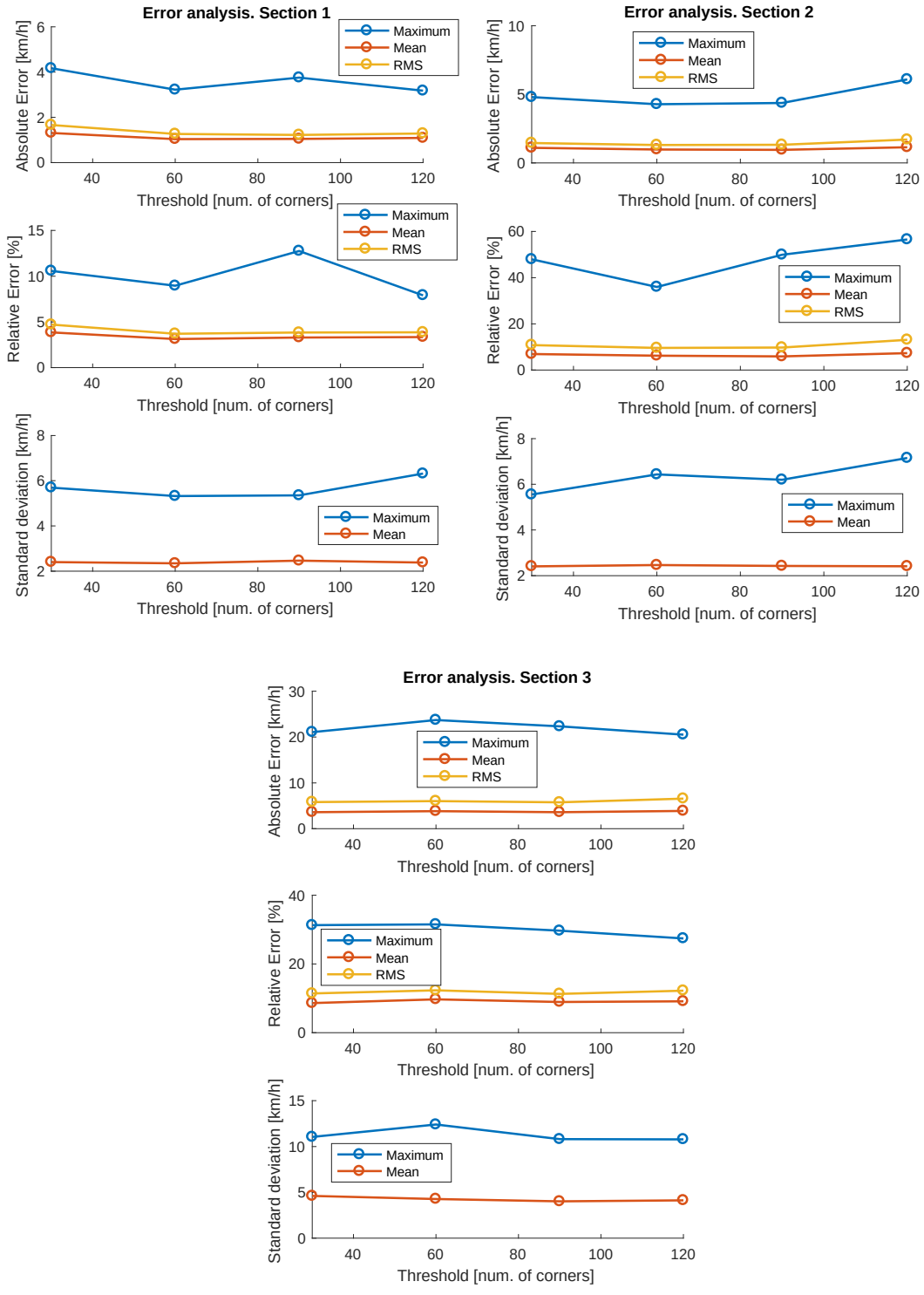

Figure 5-21 Errors as a function of the road feature tracking threshold (number of corners). Resolutions: 960x540 and 15 fps.

5.5 Influence of temporal and spatial resolution

Finally, in this section the algorithm performance is tested for a range of both spatial and temporal resolutions. In particular three different cases of each are studied leading to nine different combinations:

- Spatial resolutions:
    - 1920x1080 pixels (original), from now on FHD (Full High Definition).
    - 960x540 pixels, from now on HD (High Definition).
    - 640x360 pixels, from now on SD (Standard Definition).
- Temporal resolutions:
    - 30 fps (original).
    - 15 fps.
    - 10 fps.

Recall that the previous analyses were done using the intermediate spatial and temporal resolutions.

Below, the analyses are divided into four subsections. The first three correspond to each one of the three frame rates. Inside each one the differences between the spatial resolution are studied. Then, a comparative of all the results is done simultaneously to also see the dissimilarities and points in common between using distinct temporal resolutions.

5.5.1 Temporal resolution: 30 fps

First of all, the original temporal resolution, 30 fps, is studied in the 3 sections of the video, shown in Figure 5-22, Figure 5-23 and Figure 5-24.

In section 1 of the video, Figure 5-22, both FHD and HD are quite similar and follow well the ground truth with the exception of the parts from 96 sec to 116 sec and 140 sec to the end. The former is the one where the road has no road markings and just few shadows so the majority of the detected corners on the road actually correspond to its margins where there are bushes. This introduces errors that may make these results different. The latter is of a place where two

cars are on the other lane. This shows that FHD resolution may be more sensitive to this kind of disturbances. Nonetheless, it is not affected by the other vehicles that the car encountered during this section. The SD, for its part, is more different from 86 sec to around 117 sec, which means that reducing the resolution so much may reduce the effectiveness of the measurements at speeds above 30 km/h. Nevertheless, no error is bigger than 4 km/h or 12% (see Figure B-1), which is a good estimation of the actual speed.

Having a look at the moving standard deviation values (Figure 5-22 bottom), they are very similar independently of the spatial resolution. However, the FHD one has a big value at the end corresponding to the previously described part with cars.

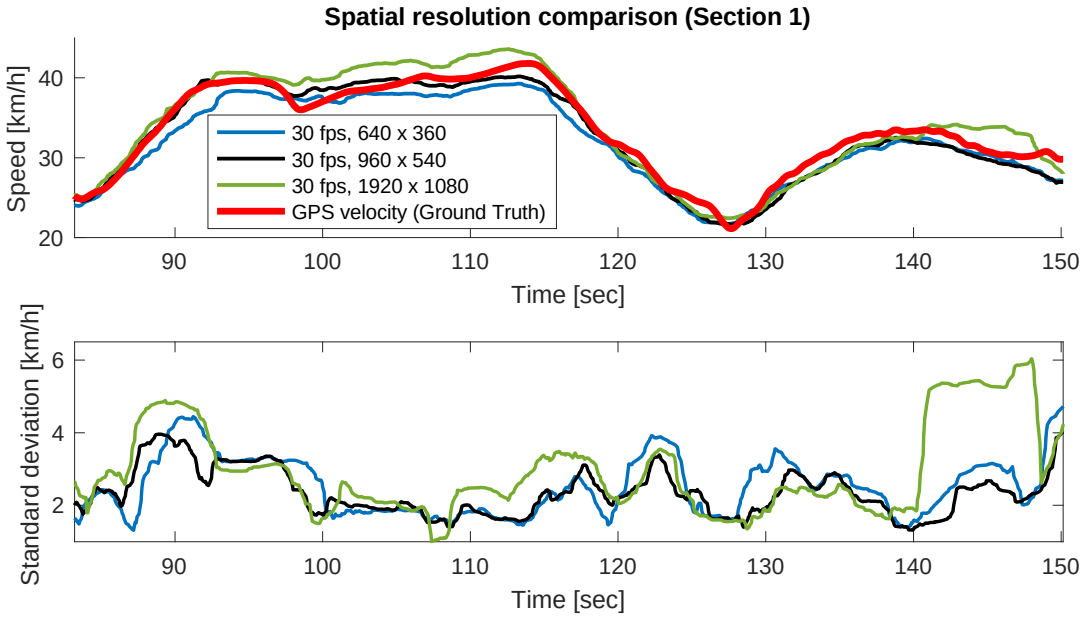

Figure 5-22 Comparative, of the video section 1, of three different spatial resolutions: FHD, HD and SD at 30 fps.

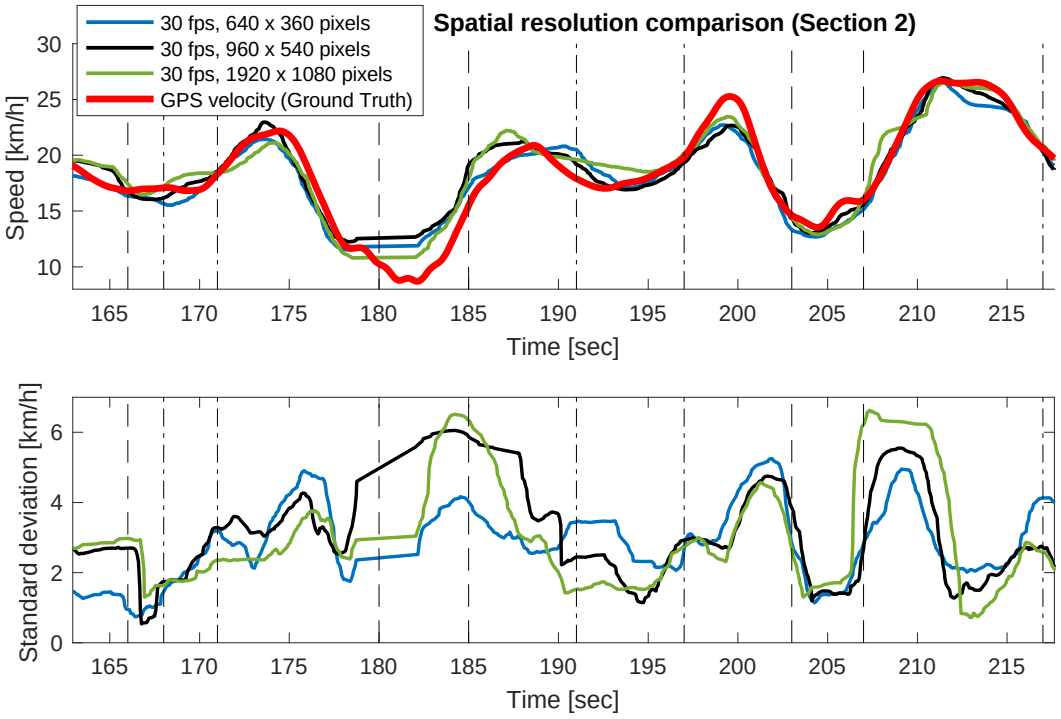

Figure 5-23 Comparative, of the video section 2, of three different spatial resolutions: FHD, HD and SD at 30 fps. Vertical dashed and dash-dot line pairs indicate each one of the six 90º curves (Note that the first and last ones do not start/finish inside the plot).

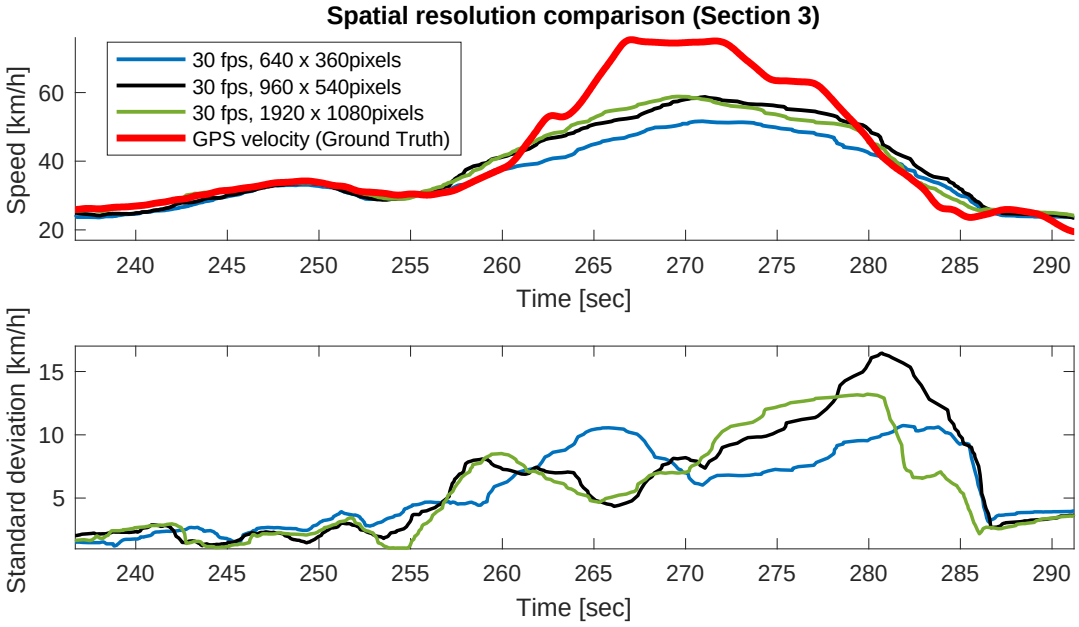

Figure 5-24 Comparative, of the video section 3, of three different spatial resolutions: FHD, HD and SD at 30 fps.

Moving to the next video section, shown in Figure 5-23, results are again very alike, including the standard deviations. But in this occasion, something different happened when measuring the velocity. At the second 178, the car is moving straight and its actual speed drops below 11 km/h so the difference between one frame and the adjacent one is very small (the car moves only 10 cm per frame). In this situation, the algorithm fails in computing a reliable essential matrix and, therefore, a reliable movement of the camera. That is why between 178 sec and 182 sec the velocity was not calculated and there is a straight line in the plot. This part is the one with the biggest errors, both absolute and relative (the latter especially because the slow velocities), being the smallest one when using FHD resolution. This is not only because of the slow velocity but also because in curve 3 there are some vehicles very close to the camera (explained in 5.2.1.2). In general errors are less than 3 km/h, with the exception of that slow part (see Figure B-2).

In the section 3 of the video, once more, the results are very similar before and after the high-speed part, both in terms of velocity and standard deviations (Figure 5-24). Errors, shown in Figure B-3, are below 3 km/h and 10% before the high-speed part. Here, the FHD results stand out because their error is less than 1 km/h from the beginning to second 253, that is, during 17 sec, which is a straight road with static shadows on the ground, that is, more corners can be extracted from the actual road plane.

Then, as explained several times, the velocity estimation greatly fails in the high-speed part, even though the temporal resolution is higher.

5.5.2 Temporal resolution: 15 fps

When using a frame rate of 15 fps the results for the first section, shown in Figure 5-25, are again very similar for the FHD and HD spatial resolutions with the same exceptions commented above for 30 fps. Figure B-4 shows the same plot but with the associated errors. When using the lowest spatial resolution, SD, the errors are, in general, bigger, but less than 5 km/h. For HD and FHD errors are less than 3.8 km/h.

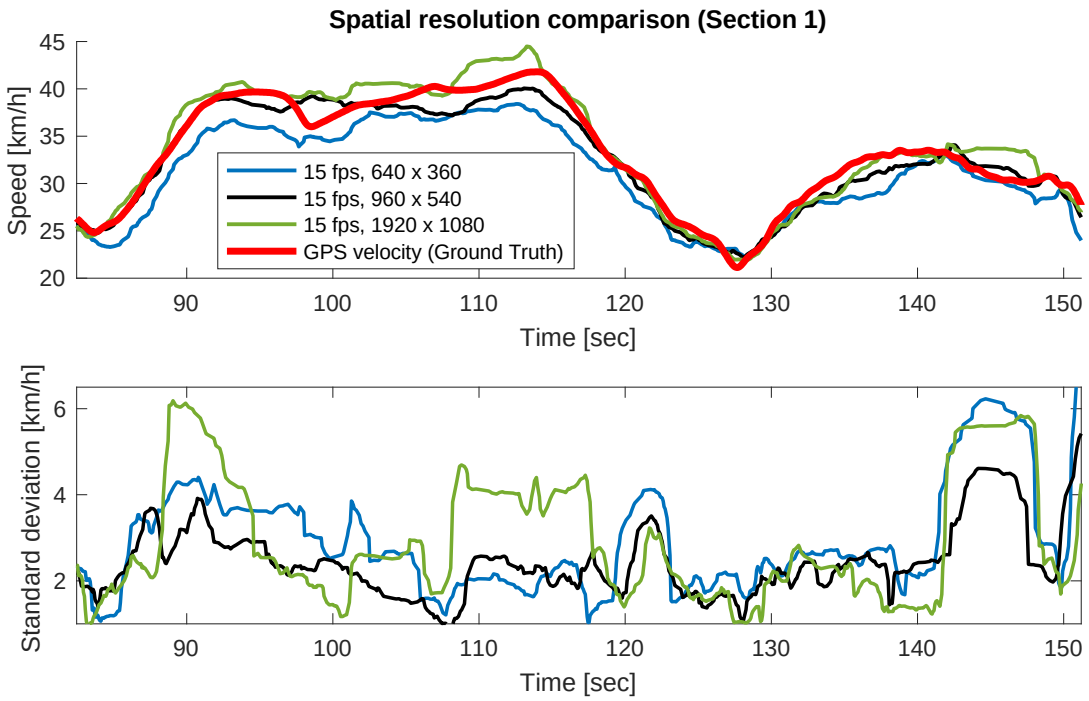

Figure 5-25 Comparative, of the video section 1, of three different spatial resolutions: FHD, HD and SD at 15 fps.

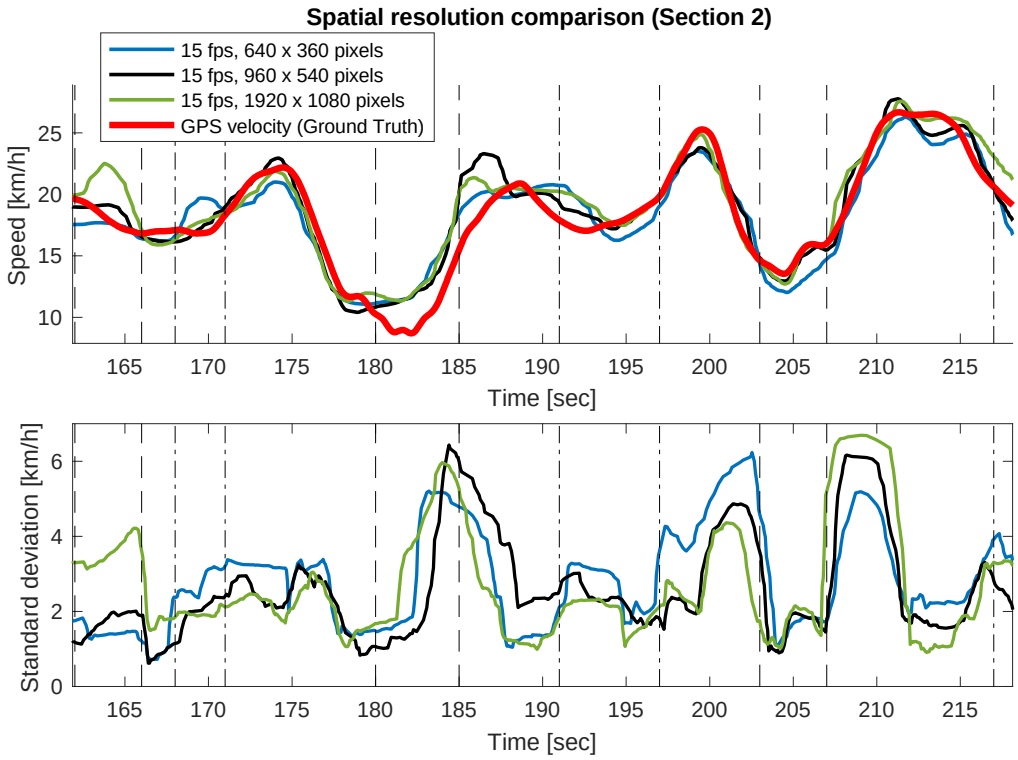

Figure 5-26 Comparative, of the video section 2, of three different spatial resolutions: FHD, HD and SD at 15 fps. Vertical dashed and dash-dot line pairs

indicate each one of the six 90º curves (Note that the first and last ones do not start/finish inside the plot).

Results for section 2, shown in Figure 5-26, look less alike to each other this time in general, especially when the car is turning. At the very beginning the FHD line has a big error (see Figure B-5) corresponding to the vehicle the camera recorded just in front of it in that curve (explained in 5.2.1.2). Then, as always, the measurements fail in the part around 185 sec, corresponding to the third curve with parked cars very close to the camera, also having a big standard deviation. As shown in Figure B-5, errors are generally smaller for FHD and HD spatial resolutions. SD results are worse, with error peaks in the curves. Nonetheless, errors are below 3 km/h with the exception of curves 1 (only for FHD) and 3.

In the section 3 of the video, the estimated velocities are very similar to each other before and after the high-speed part, both in terms of velocity and standard deviations (Figure 5-27). Errors, which are shown in Figure B-6, are again small in that first part for the three resolutions (below 4 km/h) and, once more, FHD resolution stands out for its small error there (less than 2 km/h and 10% during the first 20 sec).

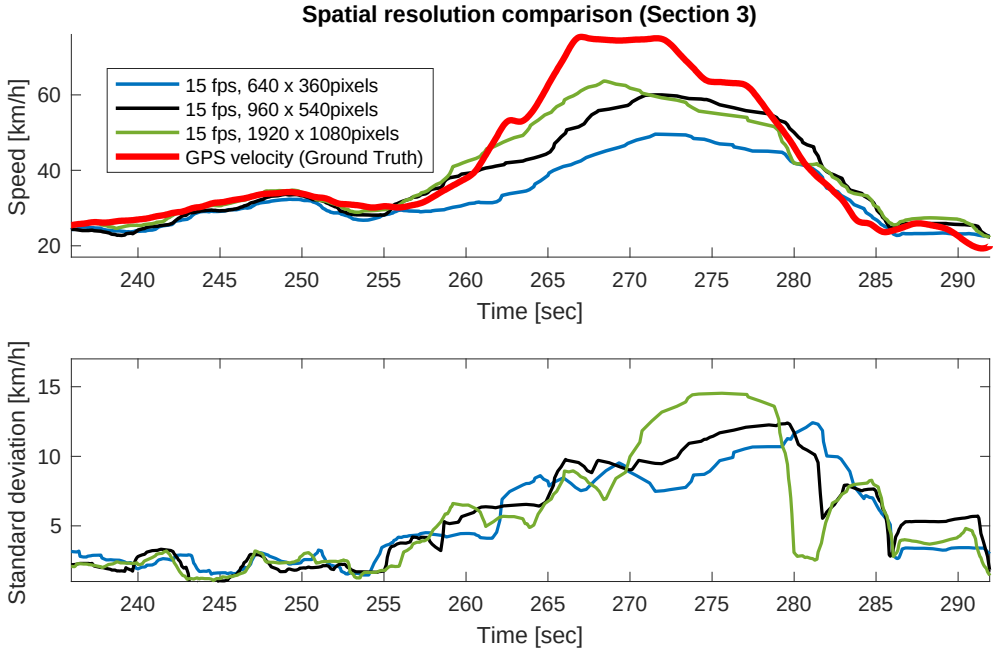

Figure 5-27 Comparative, of the video section 3, of three different spatial resolutions: FHD, HD and SD at 15 fps.

5.5.3 Temporal resolution: 10 fps

Finally, if the original temporal resolution is downsampled to only 10 fps, the next results are obtained.

In section 1, the dissimilarities between the three spatial resolutions become more evident in the part between 96 sec and 116 sec in terms of the estimated speed, although their standard deviations are very similar, specially their trends (see Figure 5-28). It is worth to remind that in this part of the video the car was driven on a road with some cars, without road markings or other kind of road features apart from some few shadows and also with bushes on its margins. The fact that now the camera pose has to be estimated from frames that are less similar than with higher temporal resolutions (at 40 km/h the camera moves 1.10 m compared to the 0.37 m with 30 fps) together with the lack of road features, may make the algorithm fail more, especially with the SD spatial resolution, where the detected features are more likely to be tracked wrongly because they are less salient.

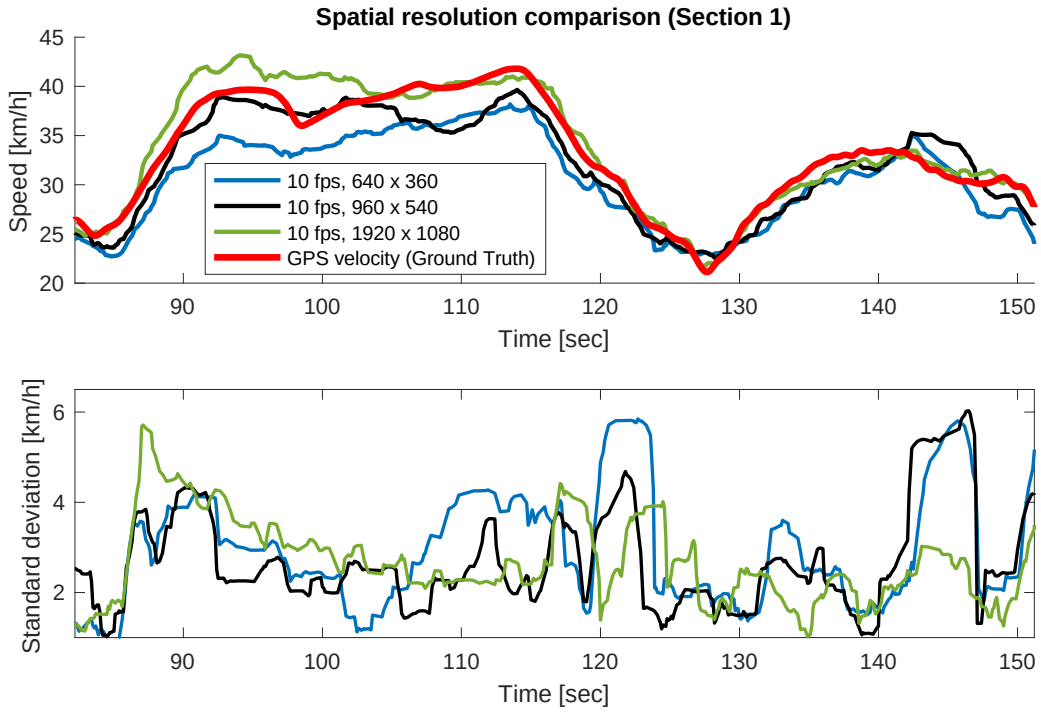

Figure 5-28 Comparative, of the video section 1, of three different spatial resolutions: FHD, HD and SD at 10 fps.

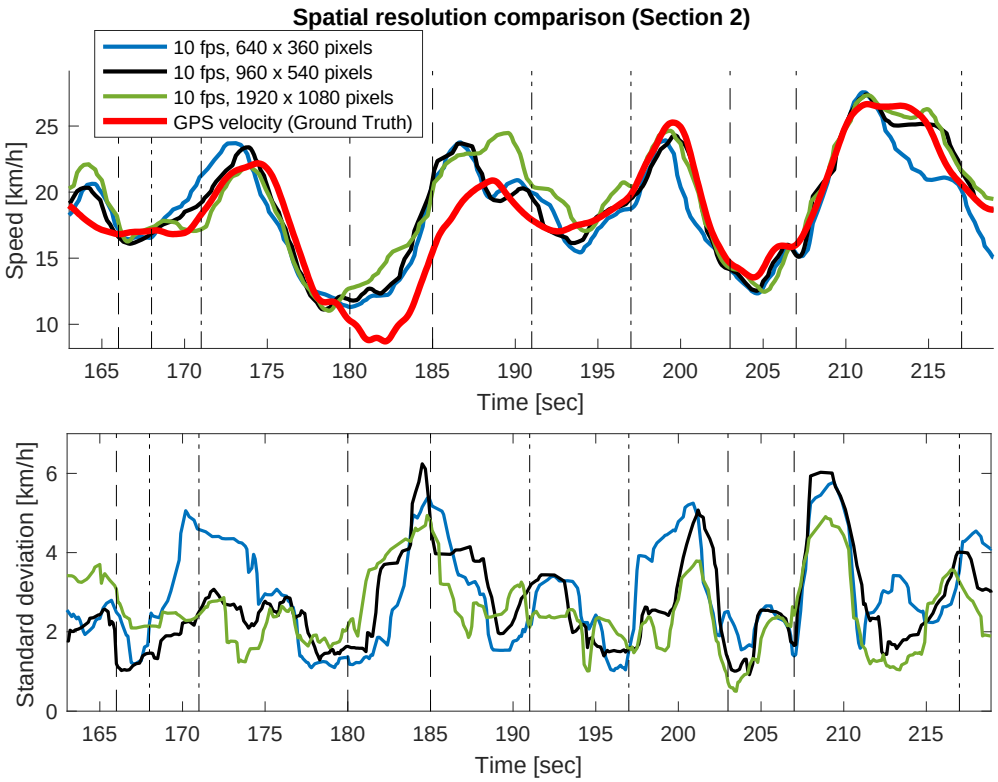

Figure 5-29 Comparative, of the video section 2, of three different spatial resolutions: FHD, HD and SD at 10 fps. Vertical dashed and dash-dot line pairs indicate each one of the six 90º curves (Note that the first and last ones do not start/finish inside the plot).

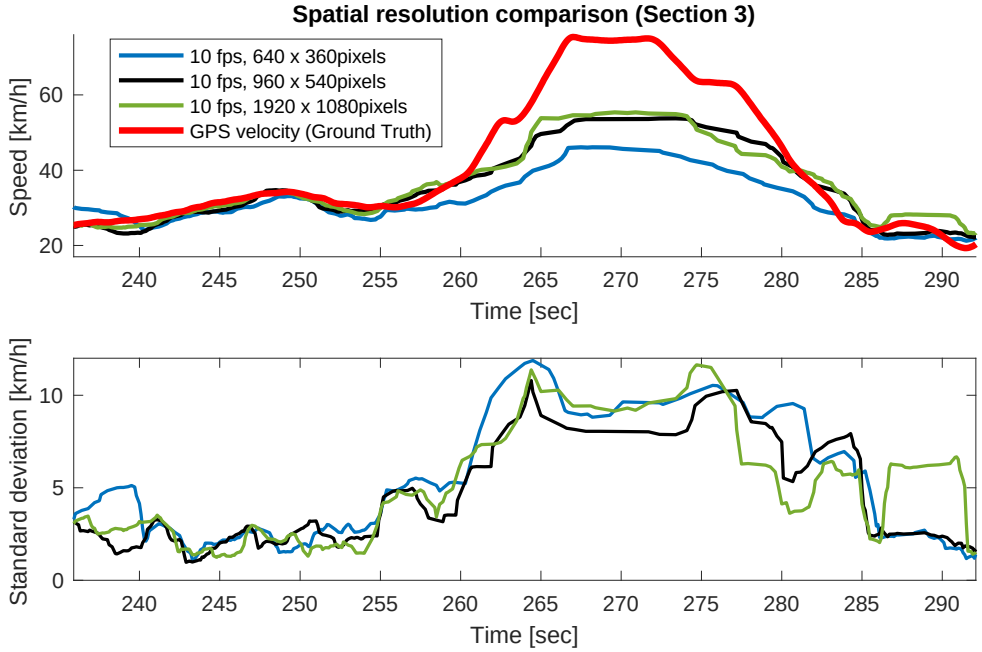

Figure 5-30 Comparative, of the video section 3, of three different spatial resolutions: FHD, HD and SD at 10 fps.

In terms of errors (shown in Figure B-7), they are less than 15% for the whole section (smaller than 6 km/h). The FHD estimation has a good performance from 103 sec to the end with an absolute error always smaller than 2 km/h or, in relative terms, less than 6%. Part of this period of time is on the track described in the above paragraph, which means that using higher spatial resolutions, FHD in this case, can help the algorithm to work better when the temporal resolution is small.

In section 2, shown in Figure 5-29, the SD resolution is the one more affected by the reduction of the frame rate. Apart from the already explained times where all of the spatial resolutions fail, the SD estimations have bigger errors than FHD and HD around 170 sec, 200 sec and 214 sec as shown in Figure B-8.

And, lastly, using a frame rate of 10 fps in the last section of the video does not have a big impact on the results if the high-speed part is omitted (see Figure 5-30 and Figure B-9). Errors are again below 4 km/h from the beginning until second 258 (first 22 sec).

5.5.4 Full comparative

Finally, comparative plots for each section of the video and every spatial/temporal resolution studied are shown from Figure B-10 to Figure B-12, where it is visible how all of them, in general terms, are quite good estimations of the actual velocity with some exceptions:

1. **Parts between 90 sec and 116 sec in section 1** (Figure B-10) **and 255 sec to 280 sec in section 3** (Figure B-12): in these two parts of the video the car was driven on the same road with some cars on the opposite lane (only section 1), without road markings or other kind of road features apart from some few shadows and also with bushes on its margins. All of these factors together have made the results to be so different from each other. The biggest the spatial resolution is the more overestimated is the velocity. And all the way around, the smaller it is the more underestimated is the obtained speed. Additionally, in video section 3, it is the error caused by the high velocities. Then, having a look at the

differences caused by the frame rate, there is not a clear pattern. In the majority of points where these discrepancies occur the 10-fps line is below the 15 fps results and this one, in turn, is below the one of 30 fps. However, this is not always true and, in addition, sometimes the results of the three temporal resolutions are very similar.

2. **Part between 142 sec and 149 sec in section 1** (Figure B-10): in this small part of the video (straight road) two cars are driving on the opposite lane very close one to each other. Both 15 and 30 fps of the FHD spatial resolution and the 10 fps HD case fail here.
3. **Part between 161 sec and 166 sec in section 2** (Figure B-11**)**: the vehicle is turning while, at the same time in front of it on the opposite line, there is another car which covers the road. Working with big temporal resolutions handles well this problem, but when using smaller ones, such as 10 fps or even 15 fps (in the case of FHD), errors increase.
4. **Part between 180 sec and 191 sec in section 2** (Figure B-11**)**: this comprises the curve 3 (the one with vehicles parked very close to the car) and the straight street from that curve until just before the curve 4 (more vehicles are parked on both sides).

It is worth to mention that, in general and specially for higher spatial resolutions, errors are minimum in zones where there were good features on the ground, such as static shadows (from 124 sec to 152 sec in section 1 and from 236 sec to 255 sec in section 3, shown in Figure 5-31) or road markings (from 161 sec to 174 sec and from 200 sec to 217 sec in section 2, Figure 5-32).

The absolute and relative errors (maximum, mean and RMS) are shown in Figure 5-33 and Figure 5-34[5]. It is clear how, in general terms, mean and RMS absolute errors decrease when increasing the spatial or the temporal resolution, and they are below 3 km/h, which means that the results are quite accurate.

[5] Errors of video section 3 has been obtained from the part before the high-speed section to compare only results where the algorithm works.

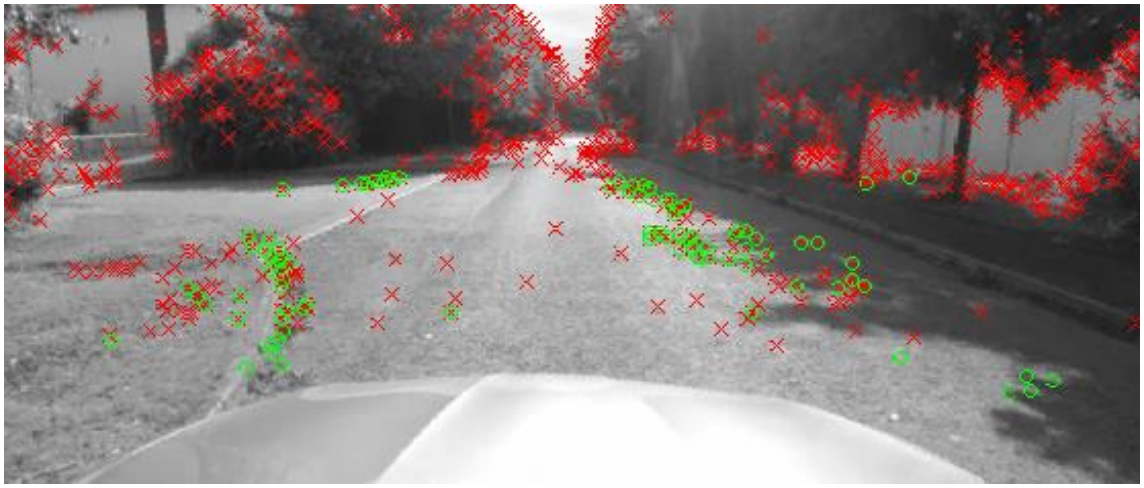

Figure 5-31 Part of a frame corresponding to video section 3 second 245 sec where there are static shadows on the road. Road corners are represented in green.

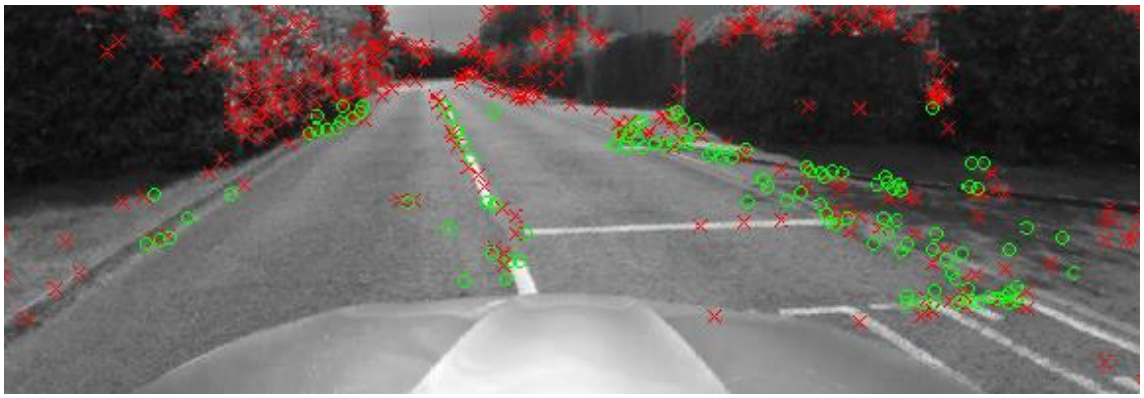

Figure 5-32 Part of a frame corresponding to video section 3 second 245 sec where there are road markings. Road corners are represented in green.

Maximum absolute errors are below 6.30 km/h, but it is worth to explain that they occur when:

- Section 1 (5.78 km/h): when the car is at the point 1 of the above list and for 10 fps and SD resolutions.
- Section 2 (6.30 km/h): when the car is at the point 4 of the above list and for 10 fps and FHD.
- Section 3 (5.37 km/h): omitting the high-speed part, the maximum error corresponds again to 10 fps and SD resolutions but, in this case, it

occurs just at the beginning of this section where the car overtakes a stopped car.

In terms of relative errors, it is similar to the absolute ones, but not as clear as before since their values depend also on the actual speed of the car so, for the same absolute error, they will be bigger for smaller velocities:

- In the case of sections 1 and 3 (without the high-speed part) the maximum mean and RMS relative errors are similar. They are below 8.60%, or below 6% when using the biggest spatial and temporal resolutions. In terms of maximums, they are smaller than 16% in section 1 and 21% in section 3 that can be 8.4% for section 1 using 30 fps and HD resolution and 6% for section 3 with 30 fps and FHD. But again, increasing the spatial and/or the temporal resolutions improves the results.
- In section 2 the maximums are much bigger, from 30% to 70%. The explanation for this is the slow velocities of the car plus the fact that they occur when the car is on the part of the road explained in the point 4 of the list of page 94. If that part is not taken into account, maximum relative errors are around 12% and 25%.

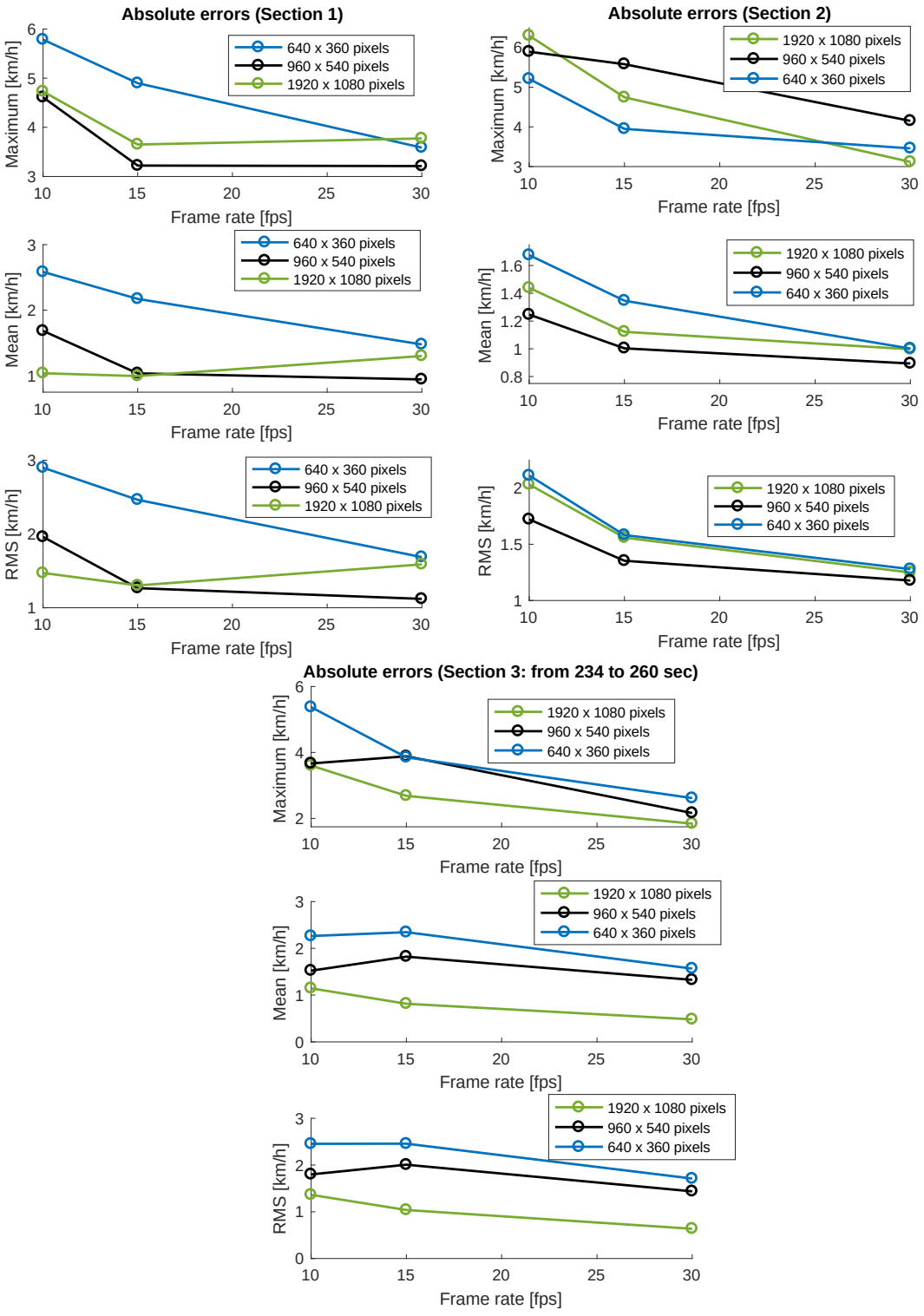

Figure 5-33 Absolute errors (maximum, mean and RMS) of the three sections for the nine spatial and temporal resolution tested combinations.

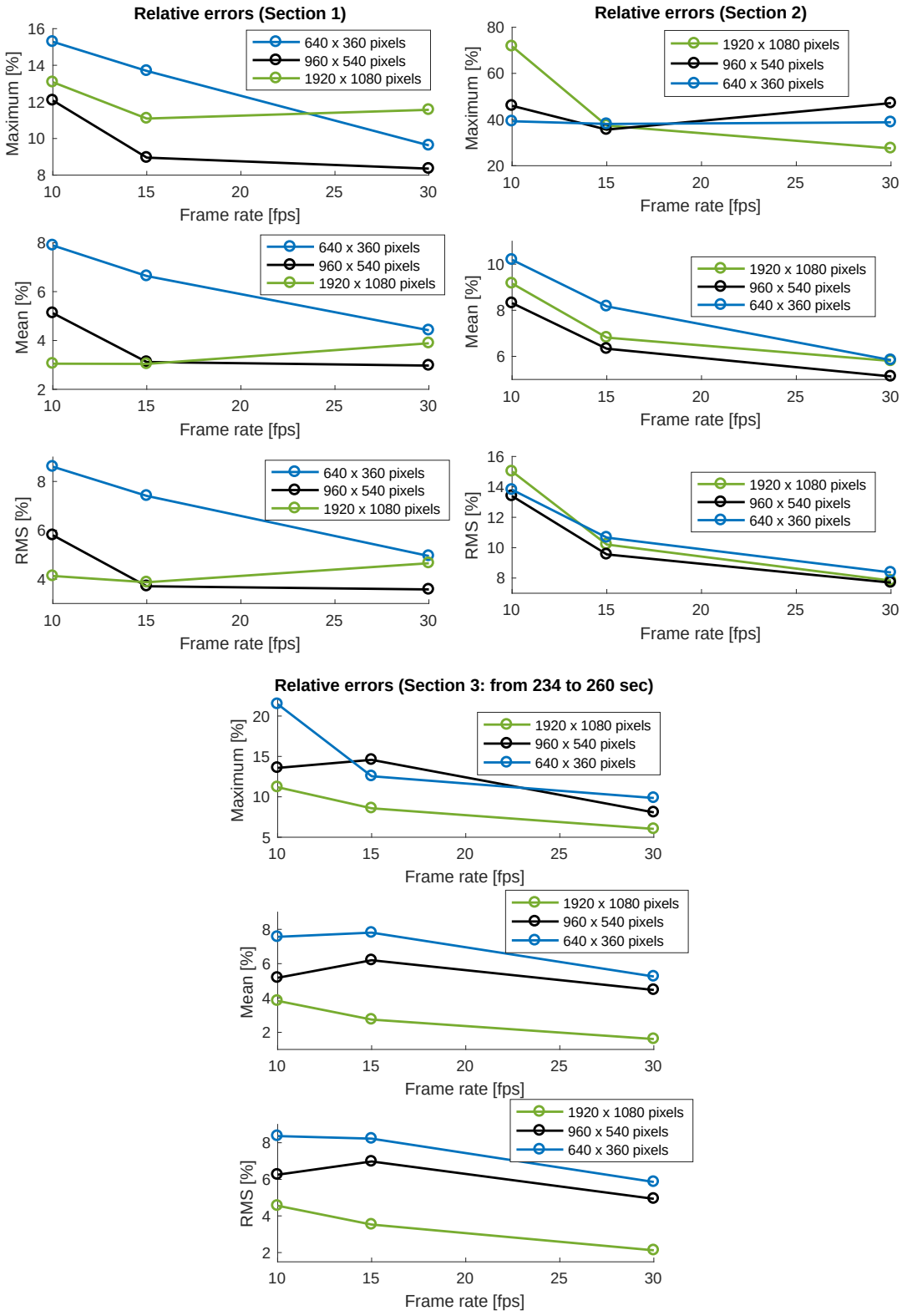

Figure 5-34 Relative errors (maximum, mean and RMS) of the three sections for the nine spatial and temporal resolution tested combinations.

5.6 Conclusions of the analysis

The studied dataset presented some interesting characteristics front the point of view of analysing the performance of the developed VO algorithm in different situations. The video has been divided into three main sections, each of them with some special characteristics: the first one is a quasi-straight road with some traffic on the opposite lane, the second one has six curves and the car moves at low speeds, and the third section is characterised by having a part at higher velocities than the other two.

Once the above is clear, the next step is smoothing the raw output measurements given by the algorithm, probably due to the low quality and contrast of the found corners on the road, which are important to determine the scale factor. So, to obtain a better estimation, the raw velocities are moving average filtered and an acceleration limit is also applied to discard those velocity peaks that are unreal (mainly caused by the presence of other cars). The influence of this last threshold has been demonstrated to be important. A very small value would eliminate too many measurements leading to a curve that miss small real changes of velocity. On the other hand, big acceleration limits do not eliminate those unreal speed peaks. Thus, the final choice was a 20-km/h/s acceleration limit.

Once the acceleration limit was fixed, the evolution of the moving standard deviation along the three studied sections was analysed, finding that at some points the measurements are not very precise, that is, they had big standard deviations. Some of these points are situations in which the car is accelerating or decelerating, but they also take place when there are other objects close to the vehicle, when the road has no road markings and just few shadows so the majority of the detected corners on the road actually correspond to its margins (with vegetation, other vehicles, etc. that increase the estimated height of the plane) and when the car is moving at high speeds.

The influence of both the frame and road feature tracking thresholds have also been studied, showing that they do not have a real influence on the final results. Only small discrepancies were found at some specific points.

Finally, the most interesting study was done: analysing the effect of both the spatial and temporal resolutions on the final estimations. Nine different combinations were tested using three different resolutions of each type. The main finding was that, excepting some specific parts of the footage, all the results are quite similar, especially in terms of the followed trend, with errors of a maximum of 5 km/h. The mentioned exceptions are points where some of these conditions were met:

1. Cars on the opposite lane, roads without road markings or other kind of road features apart from some few shadows and also with bushes on its margins.
2. Objects (mainly other vehicles), placed just in front of the car when turning.
3. Objects (mainly parked vehicles and vegetation) very close to the car and, therefore, to the road margins.
4. High-speed sections.

With all of this, the biggest errors correspond to the point where conditions 1 and 3 were true (section 3), especially when the resolutions are 15 fps and SD: 32.88 km/h (43.73% in relative form). Then, in video section 1, the maximum absolute errors take place when the first condition is true on a part of the track, but this time with errors smaller than 6 km/h (smaller than 15% in relative form).

In general, for the rest of the analysed footage (discarding the high-speed part due to its big errors), absolute errors decrease when increasing the spatial, the temporal resolution or both. Both the mean and RMS absolute errors are below 3 km/h, meaning the results are quite accurate. On the other hand, relative errors also depend on the actual velocity so, for the same absolute error, the smaller the true velocity is the bigger the relative error is. This implies that the trend the absolute errors follow is not always the same in this case. Taking this into account, mean and RMS relative errors are below 8.6%, or even below 6%

if 30 fps are utilised, for sections 1 and 3 with maximums of 16% and 21% respectively, that can be 8.4% for section 1 using 30 fps and HD resolution and 6% for section 3 with 30 fps and FHD. In section 2, relative errors are bigger due to the slow velocities of the car, being around 12% and 25% (if curve 3 is not considered), with means under 10% or even under 6% if a frame rate of 30 fps is utilised.

From all of this, some important findings have been made, mainly related with the range of applicability of the proposed method:

- Algorithm works (small errors):
    - Algorithm works better, i.e. is more accurate, in zones where the road has features: static shadows (as from 124 sec to 152 sec in section 1 and from 236 sec to 255 sec in section 3) or road markings (from 161 sec to 174 sec and from 200 sec to 217 sec in section 2).
    - The proposed algorithm has been checked to perform well when other cars are driving on the opposite lane and the road have enough features (from 124 sec to 152 sec in section 1).
- Algorithm does not work well:
    - Algorithm measurements have low precision, i.e. big standard deviation, when the road does not have enough features and has other objects on its margins (for instance bushes as in Figure 4-10 or Figure 5-9).
- Algorithm does not work:
    - Failure of computing the essential matrix and, therefore, of the camera pose, when the velocities are very small (under 11 km/h for 30 fps).
    - At high speeds (around more than 50 km/h) on roads with not enough features and with other objects on their margins the estimations are not good (big errors and standard deviations).

6 SUMMARY AND CONCLUSIONS

This final chapter includes a summary of the whole report in section 6.1, the obtained conclusions after analysing all the results in section 6.2, and some guidelines for a further study in section 6.3.

6.1 Summary

Following the main objective of this project, measuring the velocity of a ground vehicle using a forward-looking monocular camera, a brief study of the current state of the art was made to see how this problem was approach in previous works. In general, the methodologies that have been already utilised for velocity estimation are based on optical flow. However, these techniques have given results that are noisy or that work only under controlled circumstances and slow speeds, mainly because they rely in in the texture of the captured images, especially the road/ground texture, and they need small motion between frames to correctly compute that flow.

For this reason, a different approach has been utilised in this thesis. A feature-based VO method, in particular a 2D-to-2D approach, has been used to compute the camera pose with respect to the previous one by only using salient features of the image, i.e. corners, instead of all its pixels. However, this methodology has the problem that the computed camera poses have always a translation vector of unit length. That is why, together with the VO algorithm, another step is done simultaneously to obtain a scale factor that allows for obtaining a proper estimation of the movement of the camera.

From the VO itself the relative camera pose is obtained by detecting corner features using the FAST feature detector which are then tracked to the following images using the KLT tracker. Since the number of corners will decrease with new incoming images and, eventually, disappear, a redetection step is done when they fall below a certain threshold. Simultaneously, a road feature detection and tracking is also done to find salient points that are only on the road and its margins and that are utilised to obtain the plane that defines the road.

From the corner correspondences between two consecutive frames the relative camera pose is obtained but always with a translation vector of unit length. Then, knowing the relative position between the two camera poses, the 3D position of the road points is obtained by triangulation and a plane is fitted to them that will represent the *virtual road*. The distance from that plane to the virtual *camera* position, together with the actual camera height, are utilised to obtain a scale factor that allows to compute a proper velocity estimation.

But to do all of the above, a very important part of every computer vision problem, including this one, is calibrating the camera to know the projection of the position of a point in the real 3D world to its position in the image sensor plane in 2D. To do so, the intrinsic camera matrix must be obtained as well as the parameters that describes the distortion produced by the camera lens, which in this case is a wide-angle lens. The former has been obtained from the camera specifications, and the latter has been set by trial and error thanks to a part of the analysed video where the vehicle was in a car park with straight lines on the ground that seem to be curved on the recorded images.

6.2 Conclusions

The available and analysed data provided by RACELOGIC consists in a video with synchronised GPS measurements. This data has been divided into three sectors, each of them with some special characteristics: the first one is a quasi-straight road with some traffic on the opposite lane, the second one has six 90º curves and the car moves at low speeds, and the third section is characterised by having a part at higher velocities than the other two.

After running the developed algorithm over those three video sections, the raw output was quite noisy, probably due to the low quality and contrast of the found corners on the road, which are important to determine the scale factor. So, to obtain a better estimation, the raw velocities are moving average filtered and an acceleration limit is also applied to discard some velocity peaks that are unreal (mainly caused by the presence of other cars) and obtain a smoothed velocity profile.

The acceleration limit is important since its value affects the final results. A very small value would eliminate too many measurements leading to a curve that miss small real changes of velocity. On the other hand, big acceleration limits do not eliminate those unreal speed peaks.

The influence of both the frame and road feature tracking thresholds for redetection have also been studied, showing that they do not have a real influence on the final results. Only small discrepancies were found at some specific points.

Finally, nine different combinations of spatial and temporal resolutions were tested (30, 15 and 10 fps and 1920x1080, 960x540 and 640x360 pixels). Results are very similar to each other with maximum errors of 5 km/h, with the exception of some parts of the footage where they are bigger:

1. Cars on the opposite lane, roads without road markings or other kind of road features apart from some few shadows and also with vegetation on its margins.
2. Objects (mainly other vehicles), placed just in front of the car when turning.
3. Objects (mainly parked vehicles and vegetation) very close to the car and, therefore, to the road margins.
4. High-speed sections.

In general, discarding the high-speed part due to its big errors, absolute errors decrease when increasing the spatial, the temporal resolution or both. Both the mean and RMS absolute errors are below 3 km/h, meaning the results are quite accurate. On the other hand, relative errors also depend on the actual velocity so, for the same absolute error, the smaller the true velocity is the bigger the relative error is. Taking this into account, mean and RMS relative errors are below 8.6%, or even below 6% if 30 fps are utilised, for sections 1 and 3, with maximums of 16% and 21% respectively, that can be 8.4% for section 1 using 30 fps and HD resolution and 6% for section 3 with 30 fps and FHD. In section 2, relative errors are bigger due to the slow velocities of the car, reaching

maximums around 12% and 25% (if curve 3 is not considered), whit means under 10% or even under 6% if 30 fps are utilised.

From all of this, some important findings have been made, mainly related with the range of applicability of the proposed method:

- Algorithm works (small errors):
    - Algorithm works better, i.e. is more accurate, in zones where the road has features such as static shadows or road markings.
    - The proposed algorithm has been checked to perform well when other cars are driving on the opposite lane and the road have enough features.
- Algorithm does not work well:
    - Algorithm measurements have low precision, i.e. big standard deviation, when the road does not have enough features and has other objects on its margins (for instance, vegetation).
- Algorithm does not work:
    - Failure of computing the essential matrix and, therefore, of the camera pose, when the velocities are very small (under 11 km/h for 30 fps).
    - At high speeds (around more than 50 km/h), on roads with not enough features and with other objects on their margins the estimations are not good (big errors and standard deviations).

6.3 Further study

The analyses done in this thesis can be further extended by:

- Calibrating the camera with one of the available calibration tools and checking the performance of the algorithm to see if the results can be improved just by doing this step.
- Running the code on a video recorded on a road with road markings and also at higher speeds to check whether or not the results are affected by the high speeds if a road with enough features is available.

- Checking the performance of the algorithm on other conditions such as with rain, low light, more traffic, etc.
- Using other corner detectors.
- Studying the influence of the images texture, especially the one of the road, on the results.

Additionally, some improvements or changes can be made to the algorithm itself:

- Change the algorithm to work at low speeds (for instance, by just switching to a smaller frame rate when the speed is below a threshold).
- Matching features between frames instead of tracking them could help to obtain better results when the velocities are high.
- Detect cars to avoid detecting points on them.
- Compute road plane, or even the scale factor, in a different way.

REFERENCES

[1] PHOTOMETRICS, *Keep the Noise Down! Low Noise: An Integral Part of High-Performance CCD (HCCD) Camera Systems,* PHOTOMETRICS, 2010.

[2] S. W. Smith, “Chapter 25: Special Imaging Techniques,” in *The Scientist and Engineer's Guide to Digital Signal Processing*, 202, pp. 423-450.

[3] F. Porikli, “Achieving real-time object detection and tracking under extreme conditions,” *Journal of Real-Time Image Processing,* vol. 1, no. 1, pp. 33-40, 2006.

[4] W. Burger, “Zhang’s Camera Calibration Algorithm: In-Depth Tutorial and Implementation,” Hagenberg, 2016.

[5] MathWorks, “Object for storing camera parameters - MATLAB,” MathWorks, 2017. [Online]. Available: https://uk.mathworks.com/help/vision/ref/cameraparameters-class.html. [Accessed June 2017].

[6] D. Scaramuzza and F. Fraundorfer, “Visual Odometry Part I: The First 30 Years and Fundamentals,” *IEEE Robotics & Automation Magazine,* pp. 80-92, December 2011.

[7] D. Nister, O. Naroditsky and J. Bergen, “Visual Odometry,” in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’04)*, 2004.

[8] H. Longuet-Higgins, “A computer algorithm for reconstructing a scene from two projections,” *Nature,* vol. Vol. 293, no. 10, pp. 133-135, 1981.

[9] C. Harris and J. Pike, “3D positional integration from image sequences,” in *Proceedings Alvey Vision Conference*, 1988.

[10] H. P. Moravec, *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover,* Stanford, CA: Stanford University, 1980.

[11] S. Lacroix, A. Mallet and R. Chatila, “Rover self localization in planetary-like environments,” in *5th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Noordwijk, The Netherlands, 1999.

[12] J.-P. Tardif, Y. Pavlidis and K. Daniilidis, “Monocular Visual Odometry in Urban

Environments Using an Omnidirectional Camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, 2008.

[13] D. Nister, "An Efficient Solution to the Five-Point Relative Pose Problem," in *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 2004.

[14] M. J. Milford and G. F. Wyeth, "Single Camera Vision-Only SLAM on a Suburban Road Network," in *IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, 2008.

[15] R. Goecke, A. Asthana, N. Pettersson and L. Petersson, "Visual Vehicle Egomotion Estimation using the Fourier-Mellin Transform," in *IEEE Intelligent Vehicles Symposium*, Istanbul, Turkey, 2007.

[16] S. Roy, "Ground Vehicle Velocity Estimation Based on Fusion of Monocular Visual Odometry and IMU Data," MSc Thesis, Cranfield University, 2016.

[17] M. Bevilacqua, A. Tsourdos and A. Starr, "Egomotion estimation for monocular camera visual odometer," in *Instrumentation and Measurement Technology Conference Proceedings (I2MTC)*, Taipei, Taiwan, 2016.

[18] X. Song, L. D. Seneviratne, K. Althoefer, Z. Song and Y. H. Zweiri, "Visual Odometry for Velocity Estimation of UGVs," in *Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation*, Harbin, China, 2007.

[19] F. Fraundorfer and D. Scaramuzza, "Visual Odometry Part II: Matching, Robustness, Optimization, and Applications," *IEEE Robotics & Automation Magazine,* pp. 78-90, June 2012.

[20] RACELOGIC Ltd., *VBOX Video HD2 - RLVBVDHD2-1/RLVBVDHD2-2/ RLVBVDHD2DSP,* Datasheet, 2017.

[21] D. Ray, "7 – Camera Calibration database – argus / 3D for the people," Argus project, 11 March 2016. [Online]. Available: http://argus.web.unc.edu/camera-calibration-database/. [Accessed 6 July 2017].

[22] A. Singh, "Avi Singh's blog. Monocular Visual Odometry using OpenCV," 8 June 2015. [Online]. Available: https://avisingh599.github.io/vision/monocular-vo/. [Accessed 21 June 2017].

[23] A. Singh, “Avi Singh's blog. Visual Odmetry from scratch - A tutorial for beginners,” 25 March 2015. [Online]. Available: https://avisingh599.github.io/vision/visual-odometry-full/. [Accessed 21 June 2017].

[24] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox and N. Roy, “Visual odometry and mapping for autonomous flight using an RGB-D camera,” *Robotics Research,* vol. 100, pp. 235-252, 2017.

[25] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *Computer Vision – ECCV 2006*, Berlin, Heidelberg, 2006.

[26] MathWorks, “Detect corners using FAST algorithm and return cornerPoints object - MATLAB detectFASTFeatures,” MathWorks, 2017. [Online]. Available: https://uk.mathworks.com/help/vision/ref/detectfastfeatures.html. [Accessed 10 June 2017].

[27] A. Schmidt, M. Kraft and M. Fularz, “On Tracking and Matching in Vision Based Navigation,” in *Image Analysis and Recognition. ICIAR 2014*, 2014.

[28] MathWorks, “Track points in video using Kanade-Lucas-Tomasi (KLT) algorithm - MATLAB,” MathWorks, 2017. [Online]. Available: https://uk.mathworks.com/help/vision/ref/vision.pointtracker-class.html. [Accessed 1 July 2017].

[29] MathWorks, “3-D locations of undistorted matching points in stereo images - MATLAB triangulate,” MathWorks, 2017. [Online]. Available: http://uk.mathworks.com/help/vision/ref/triangulate.html#outputarg_reprojectionErrors. [Accessed 15 July 2017].

APPENDICES

Appendix A Acceleration limit analysis

This appendix contains some plots regarding the analysis done in section 5.2.2 *Acceleration limit* that are quite big to be in the main structure of the report.

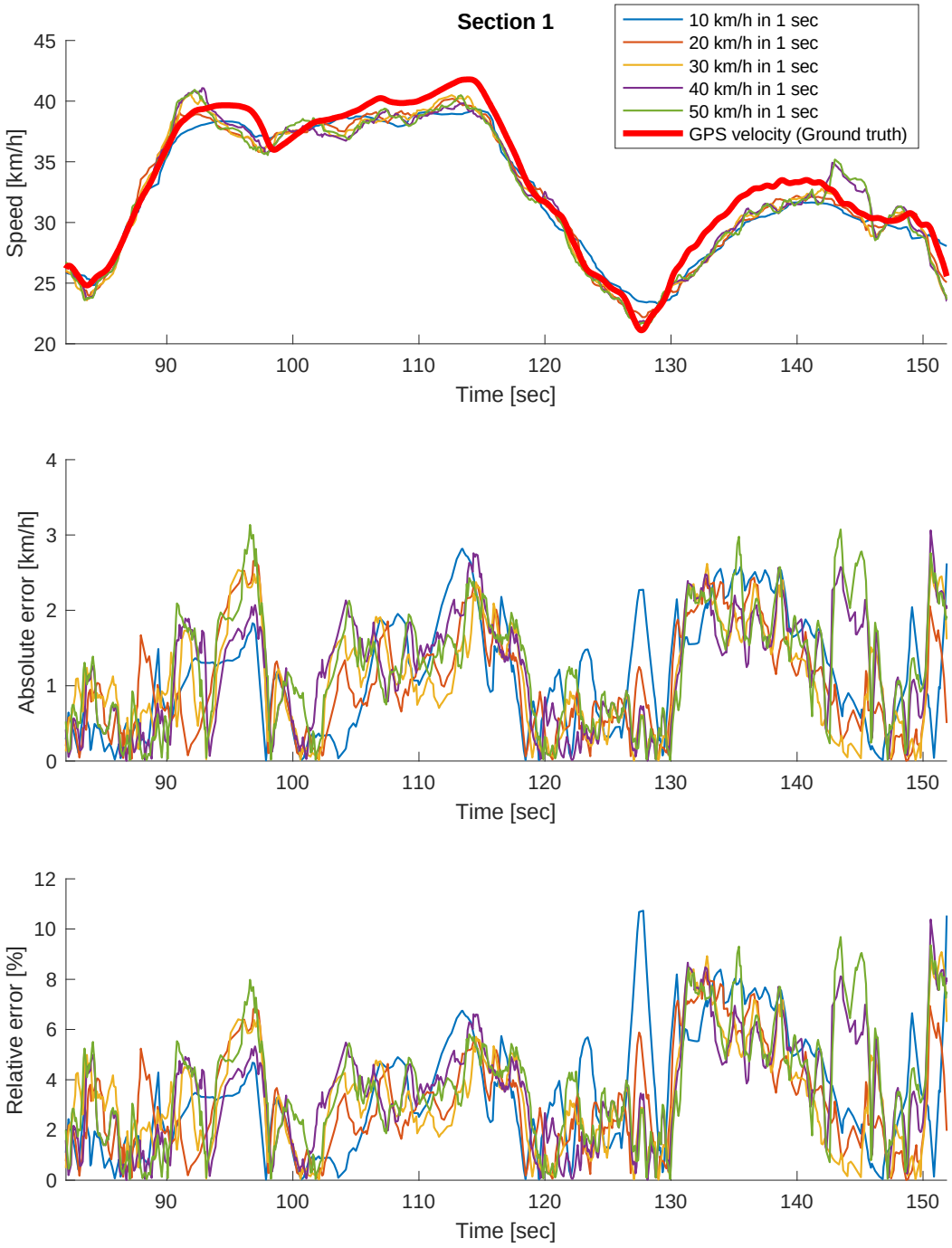

Figure A-1 Velocity profile of section 1 with the corresponding absolute and relative errors for five different acceleration limits. For more details go to page 72

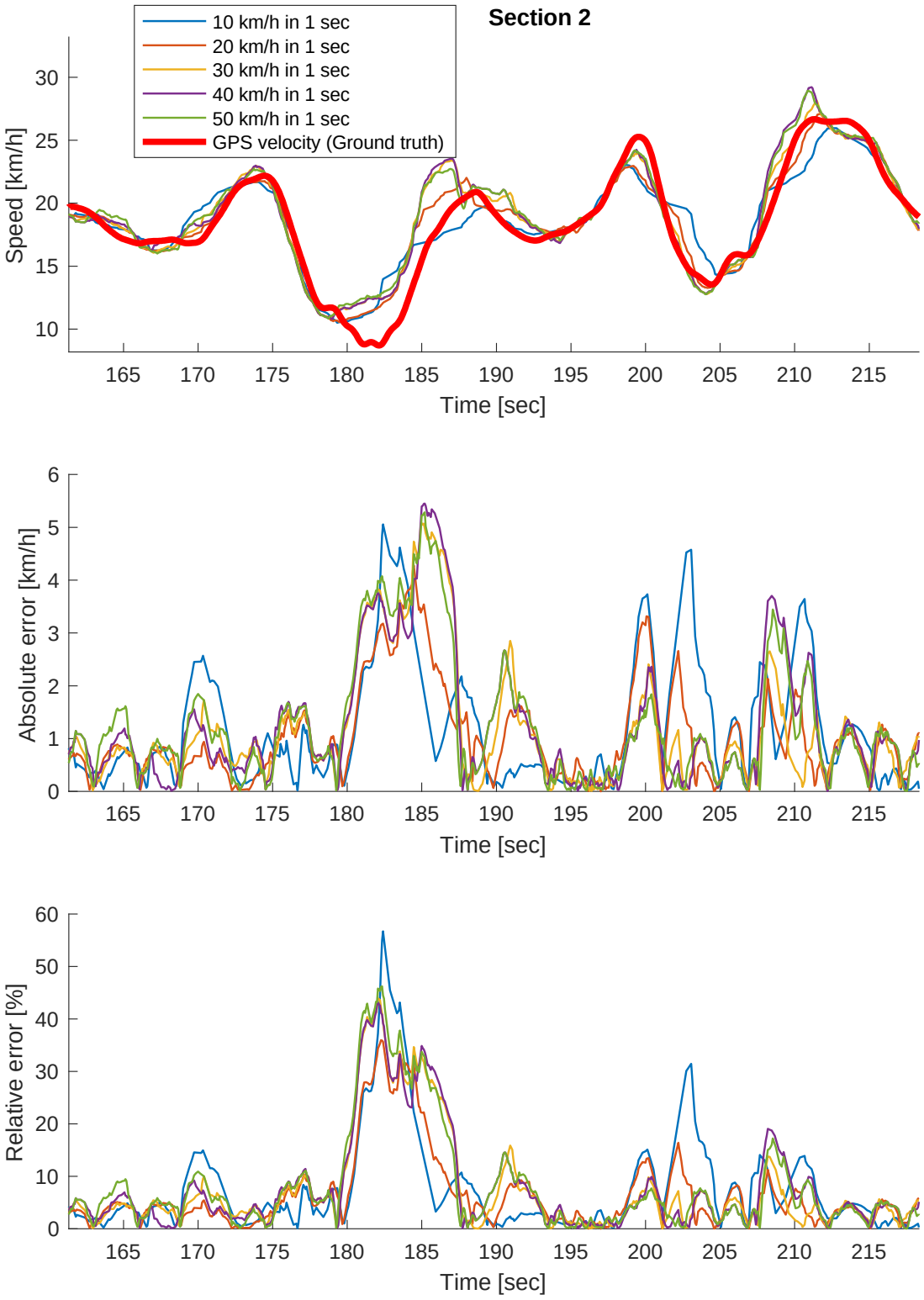

Figure A-2 Velocity profile of section 2 with the corresponding absolute and relative errors for five different acceleration limits. For more details go to page 72

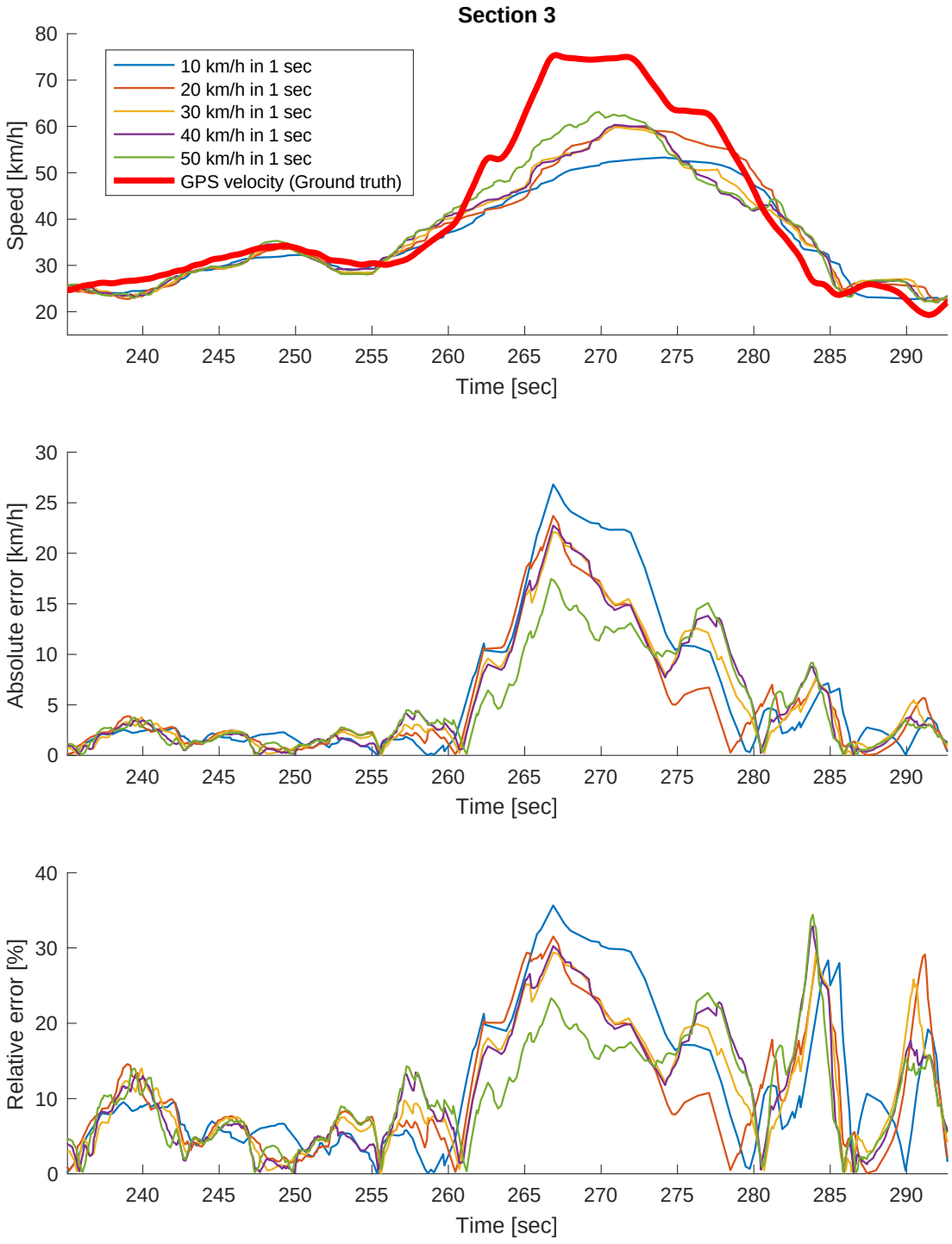

Figure A-3 Velocity profile of section 3 with the corresponding absolute and relative errors for five different acceleration limits. For more details go to page 75

Appendix B Spatial and temporal resolution analysis

This appendix contains some plots regarding the analysis done in section 5.5 *Influence of temporal and spatial resolution* that are quite big to be in the main structure of the report.

B.1 30 fps

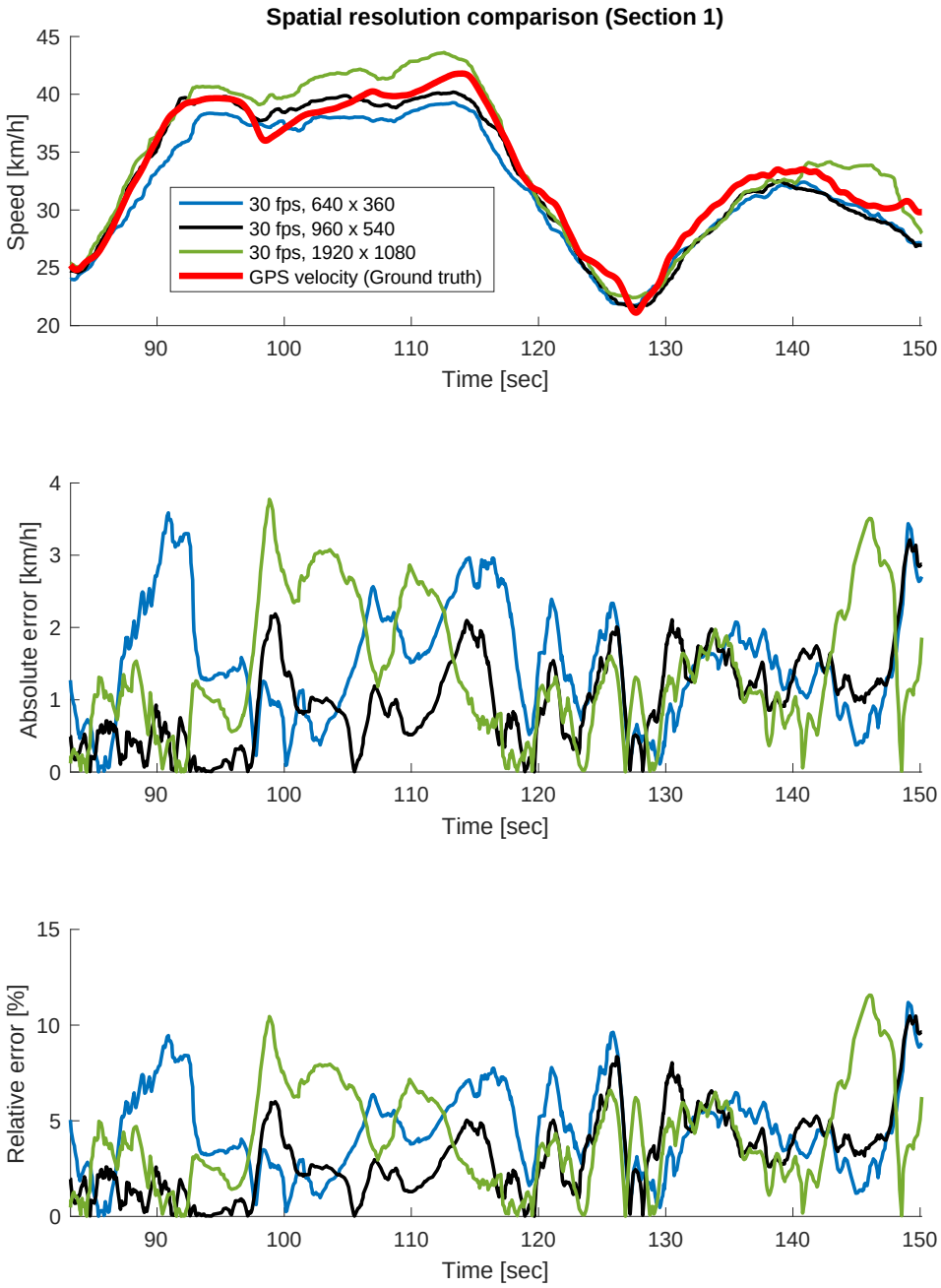

Figure B-1 Velocity profile of section 1 with the corresponding absolute and relative errors for the three analysed spatial resolutions at 30 fps. More details on page 86.

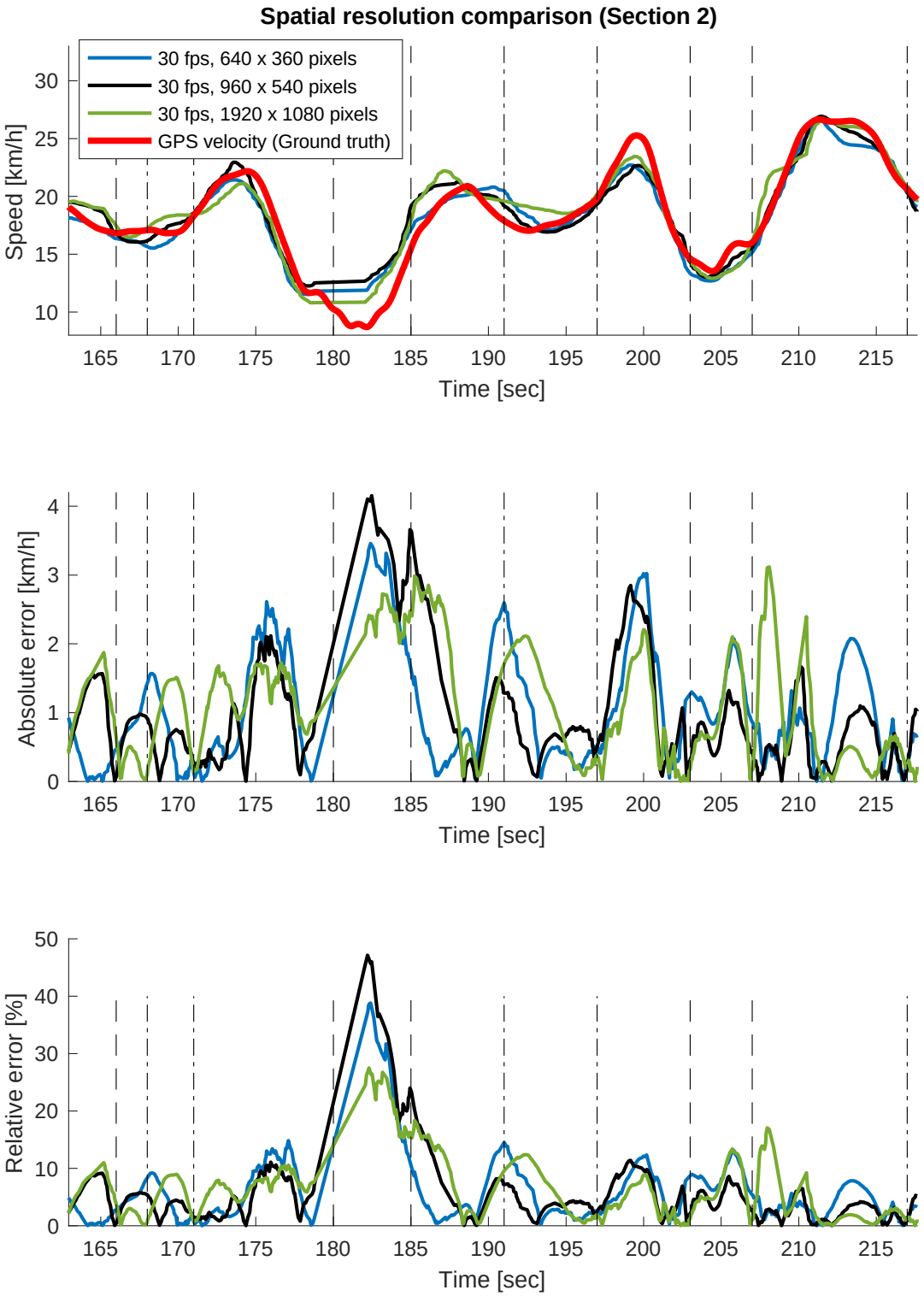

Figure B-2 Velocity profile of section 2 with the corresponding absolute and relative errors for the three analysed spatial resolutions at 30 fps. Vertical dashed and dash-dot line pairs indicate each one of the six 90º curves (Note that the first and last ones do not start/finish inside the plot). More details on page 88.

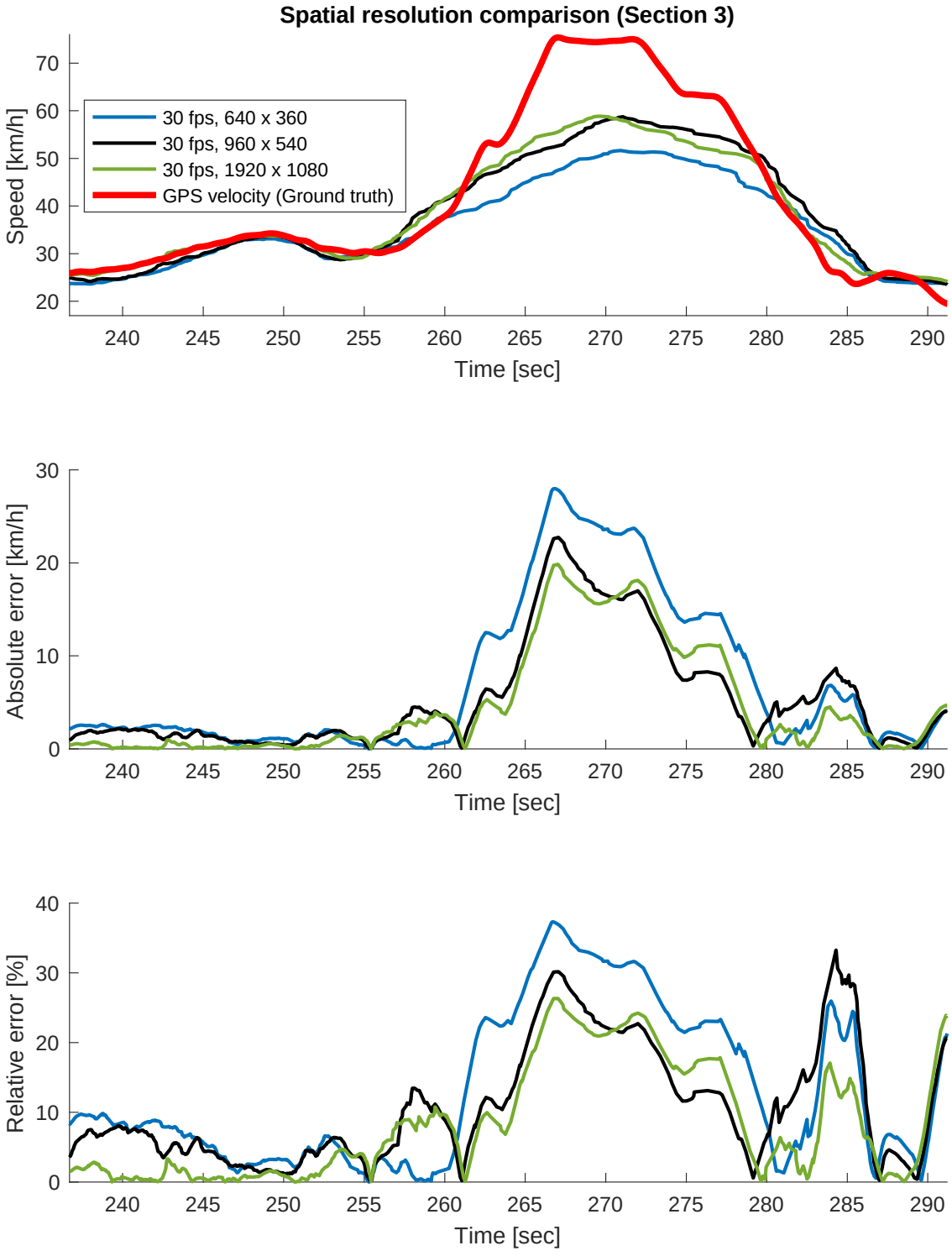

Figure B-3 Velocity profile of section 3 with the corresponding absolute and relative errors for the three analysed spatial resolutions at 30 fps. More details on page 88.

B.2 15 fps

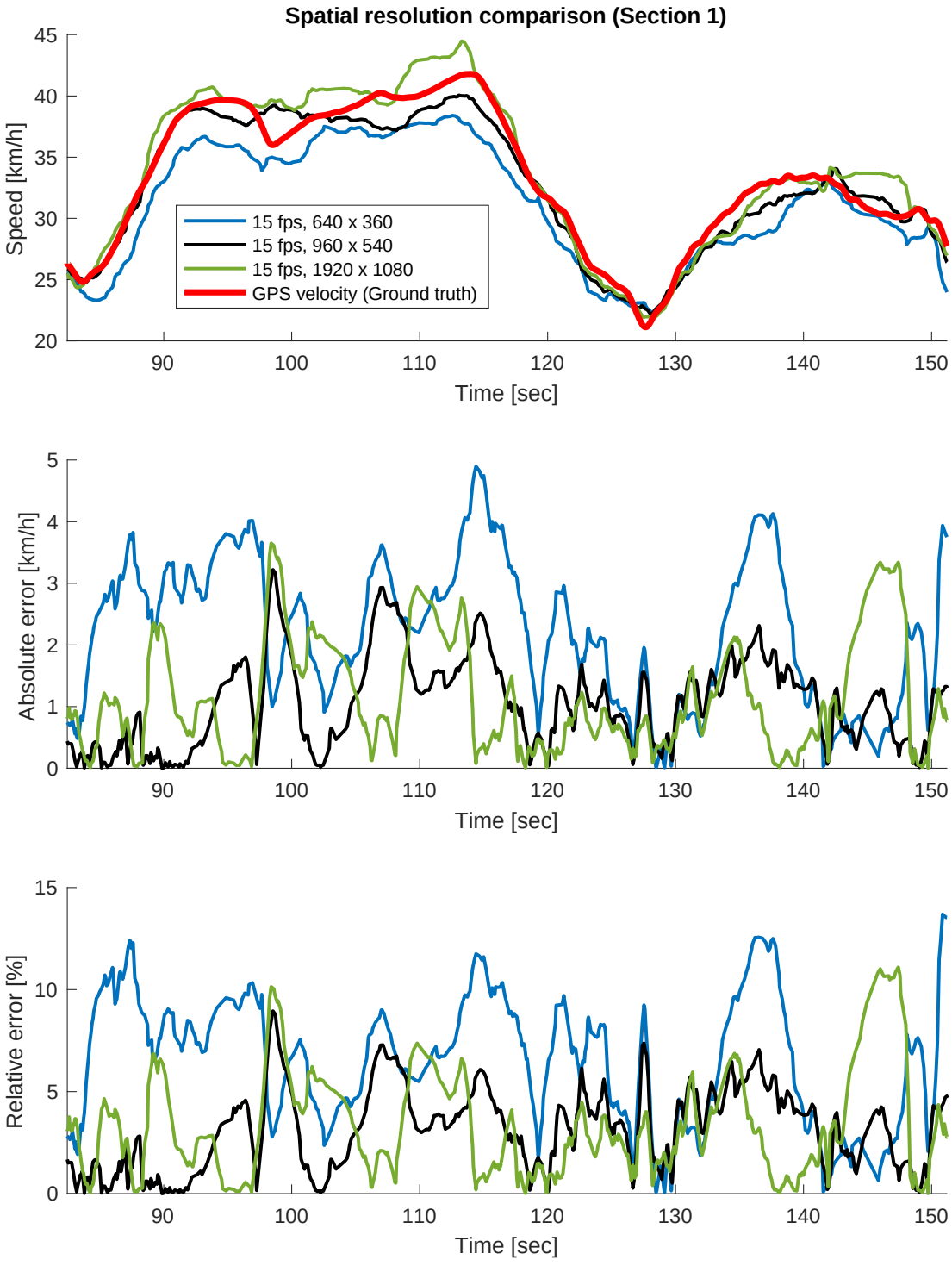

Figure B-4 Velocity profile of section 1 with the corresponding absolute and relative errors for the three analysed spatial resolutions at 15 fps. More details on page 88.

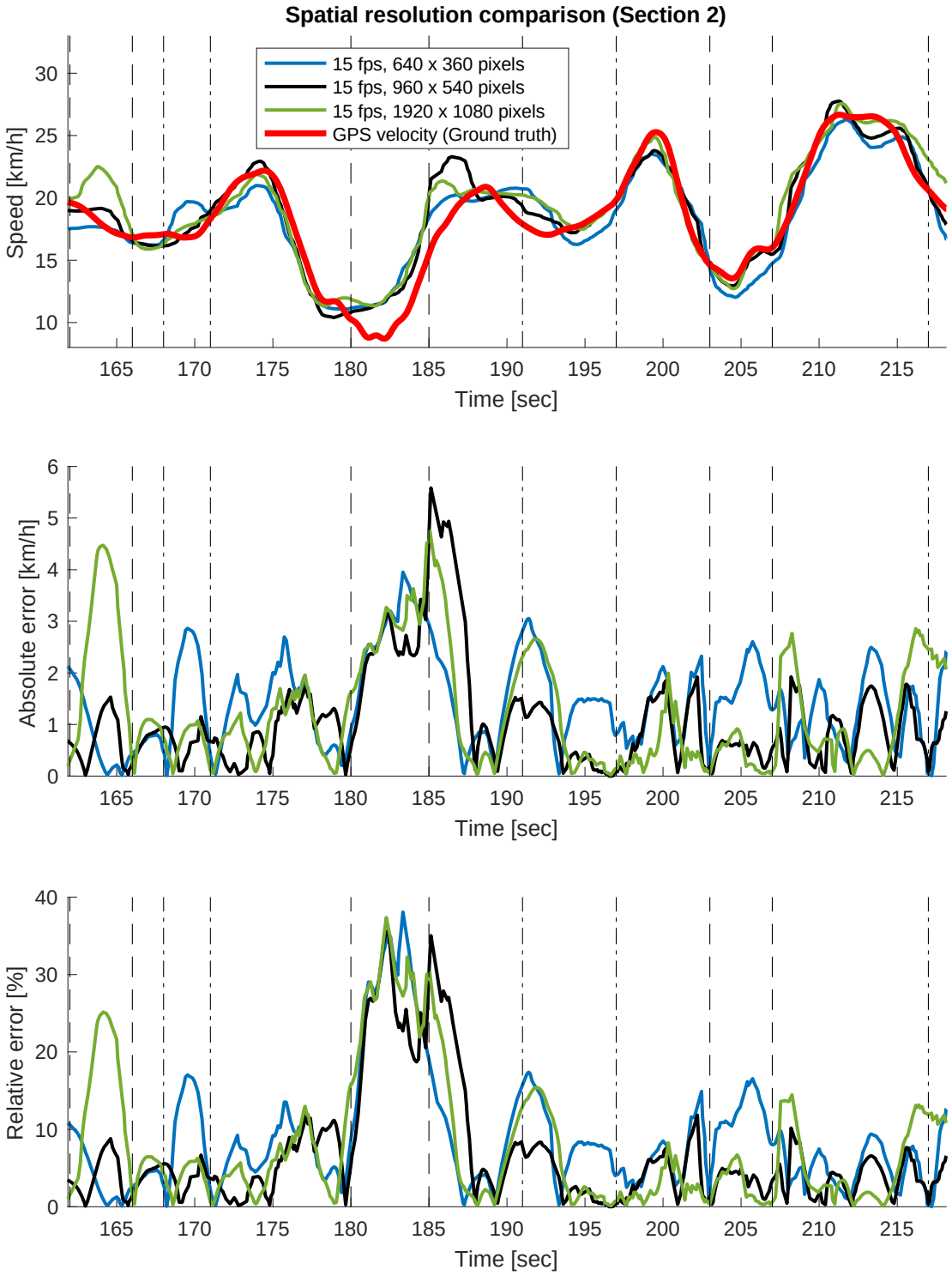

Figure B-5 Velocity profile of section 2 with the corresponding absolute and relative errors for the three analysed spatial resolutions at 15 fps. Vertical dashed and dash-dot line pairs indicate each one of the six 90º curves (Note that the last one does not finish inside the plot). More details on page 90.

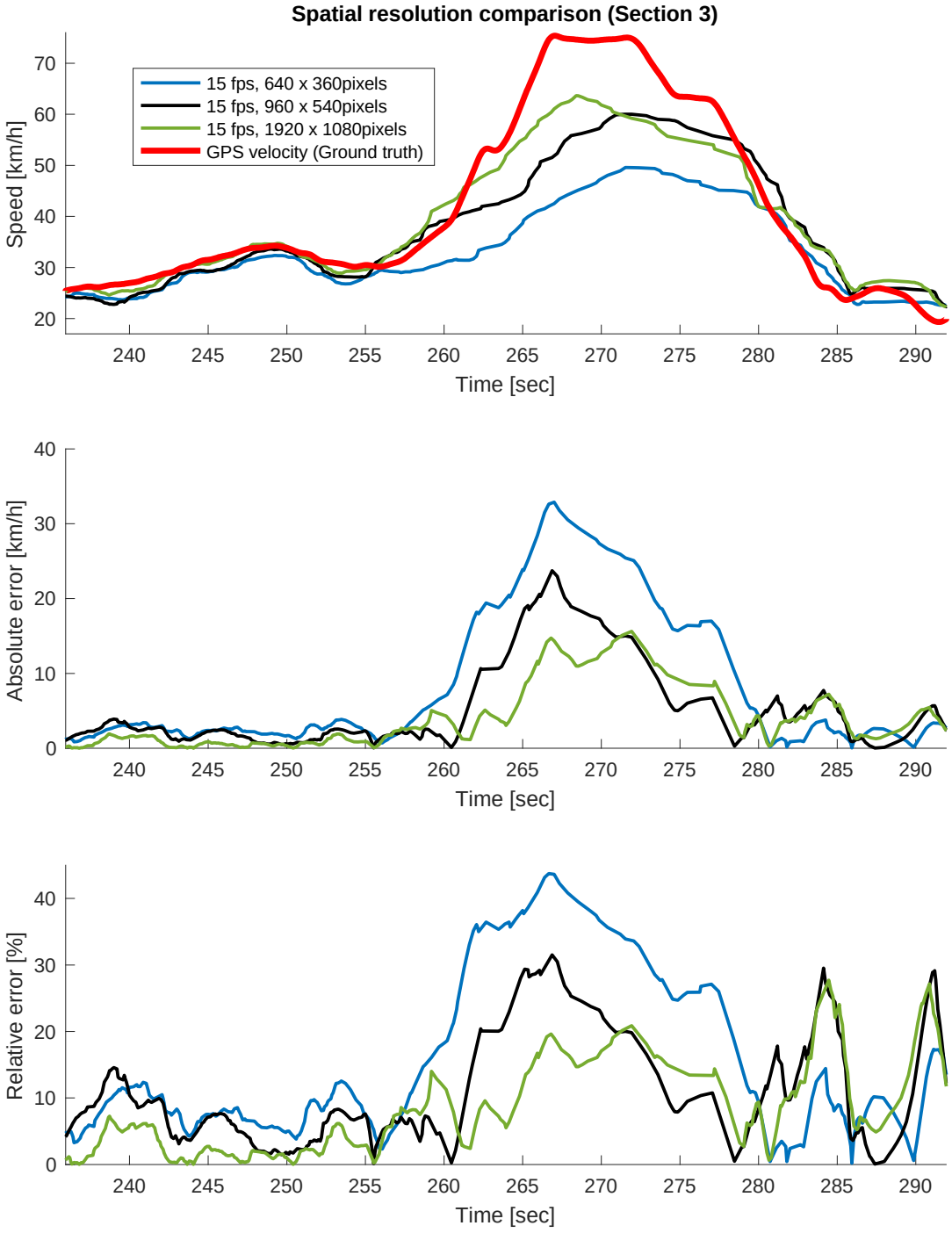

Figure B-6 Velocity profile of section 3 with the corresponding absolute and relative errors for the three analysed spatial resolutions at 15 fps. More details on page 90.

B.3 10 fps

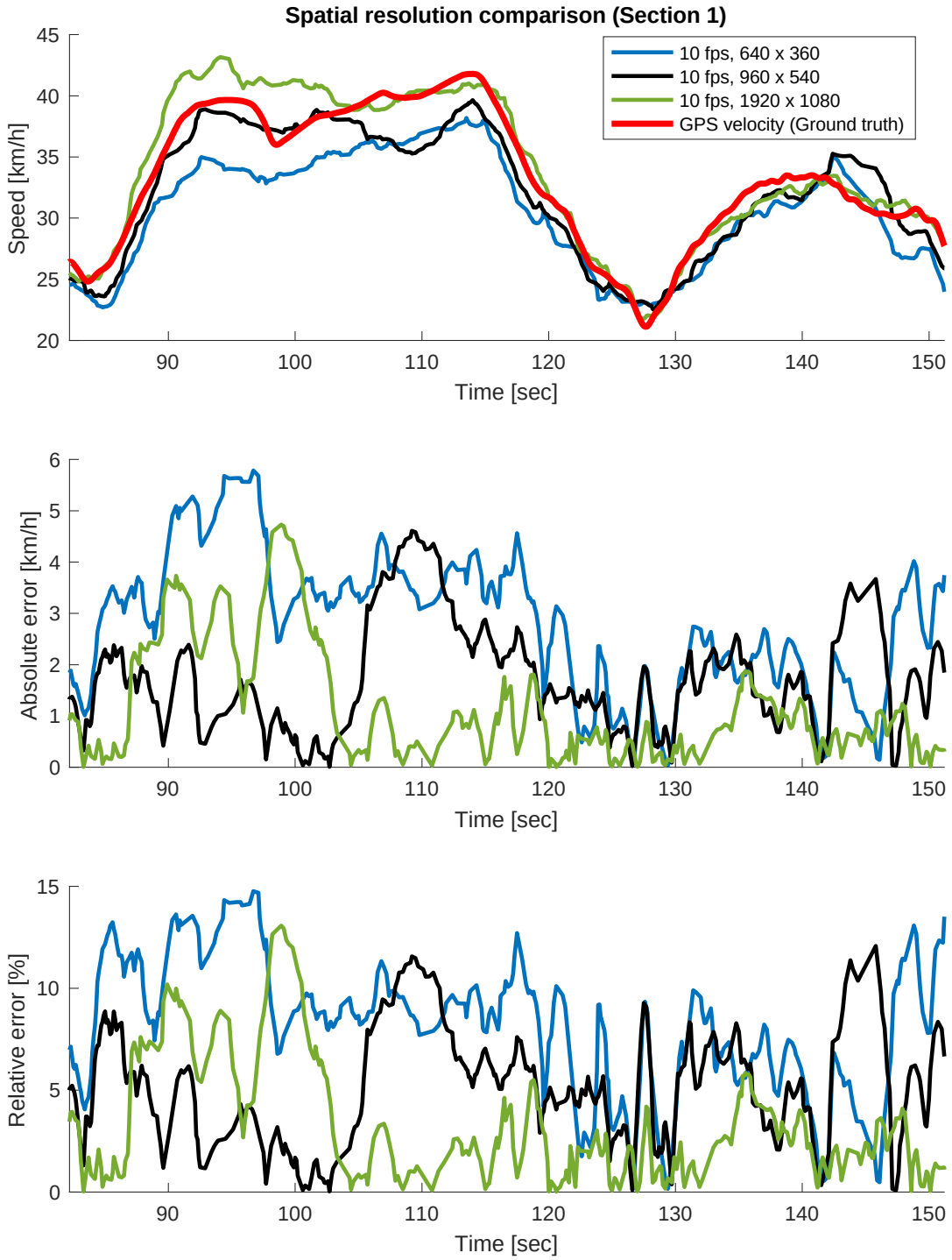

Figure B-7 Velocity profile of section 1 with the corresponding absolute and relative errors for the three analysed spatial resolutions at 10 fps. More details on page 93.

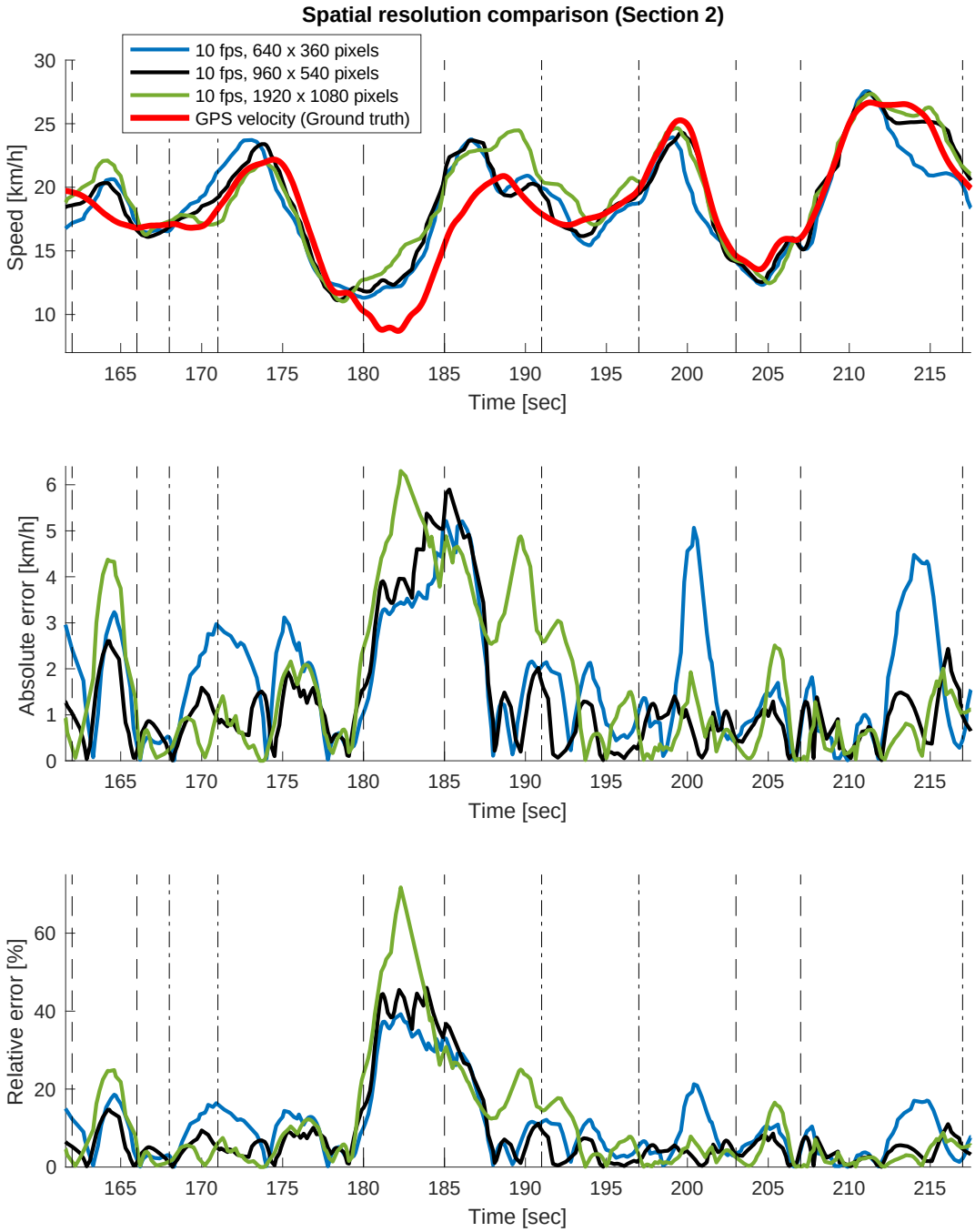

Figure B-8 Velocity profile of section 2 with the corresponding absolute and relative errors for the three analysed spatial resolutions at 10 fps. Vertical dashed and dash-dot line pairs indicate each one of the six 90º curves (Note that the last one does not finish inside the plot). More details on page 93.

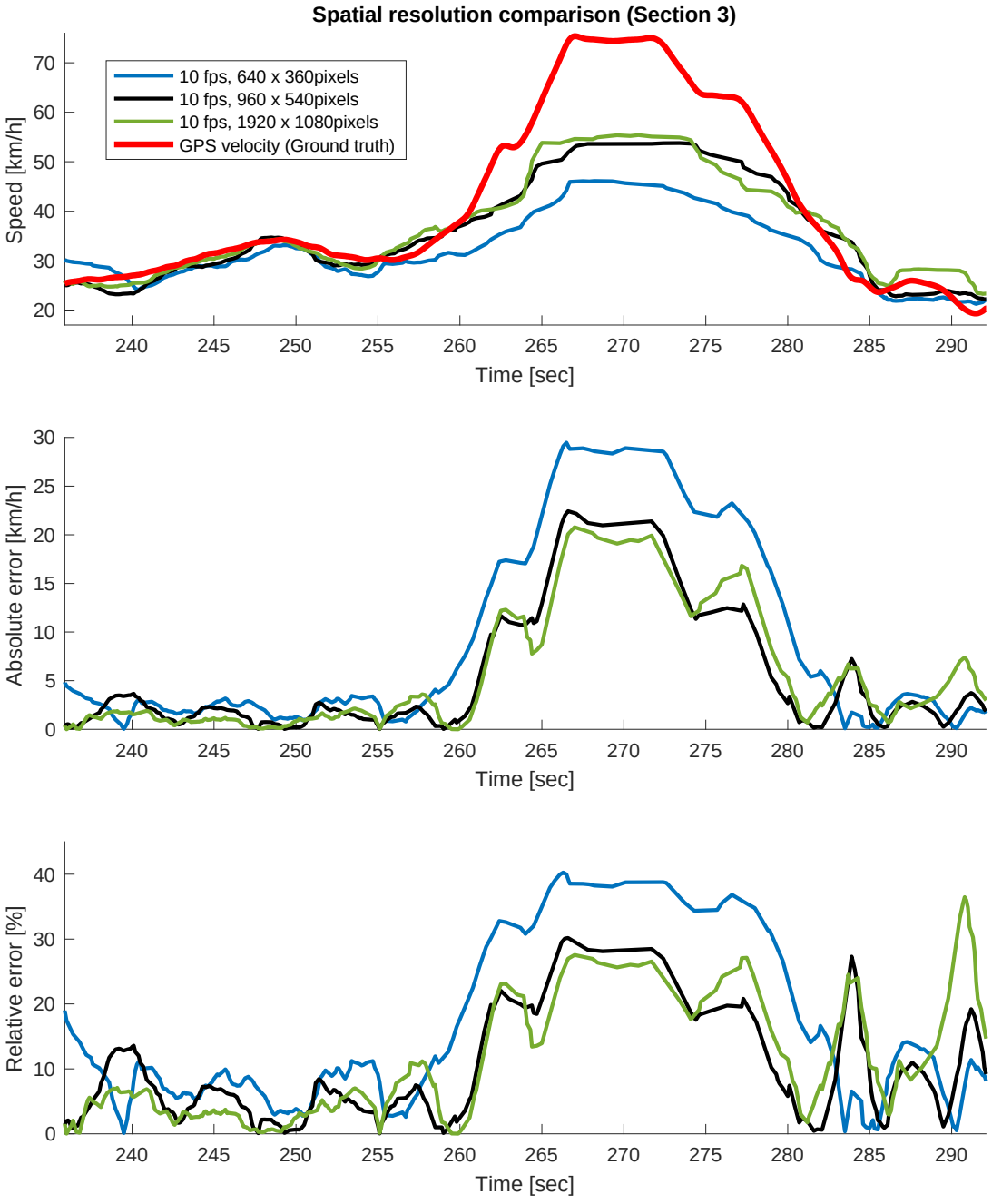

Figure B-9 Velocity profile of section 3 with the corresponding absolute and relative errors for the three analysed spatial resolutions at 10 fps.

B.4 All spatial and temporal resolutions

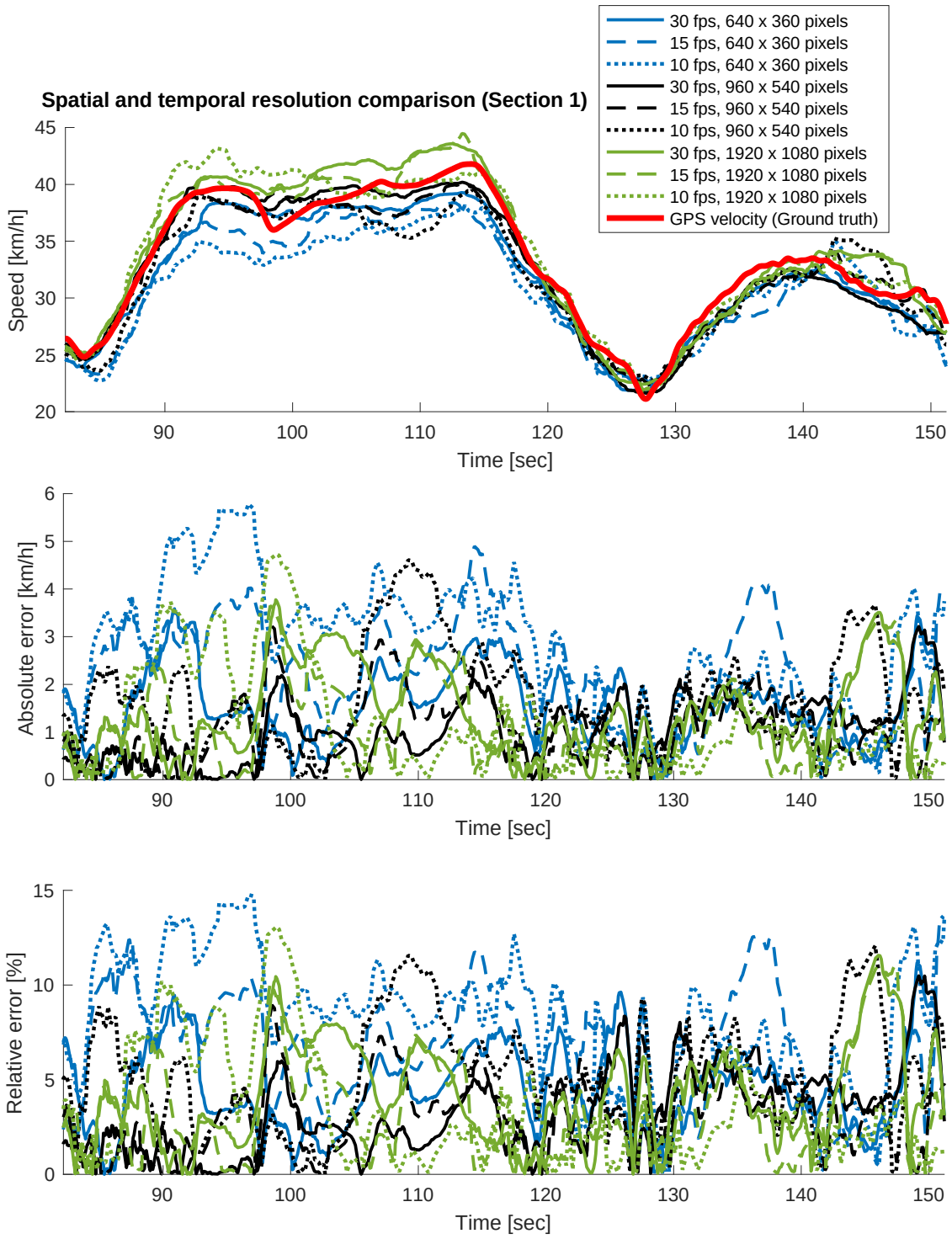

Figure B-10 Velocity profile of section 1 with the corresponding absolute and relative errors for all the analysed spatial and temporal resolutions. More details on page 93.

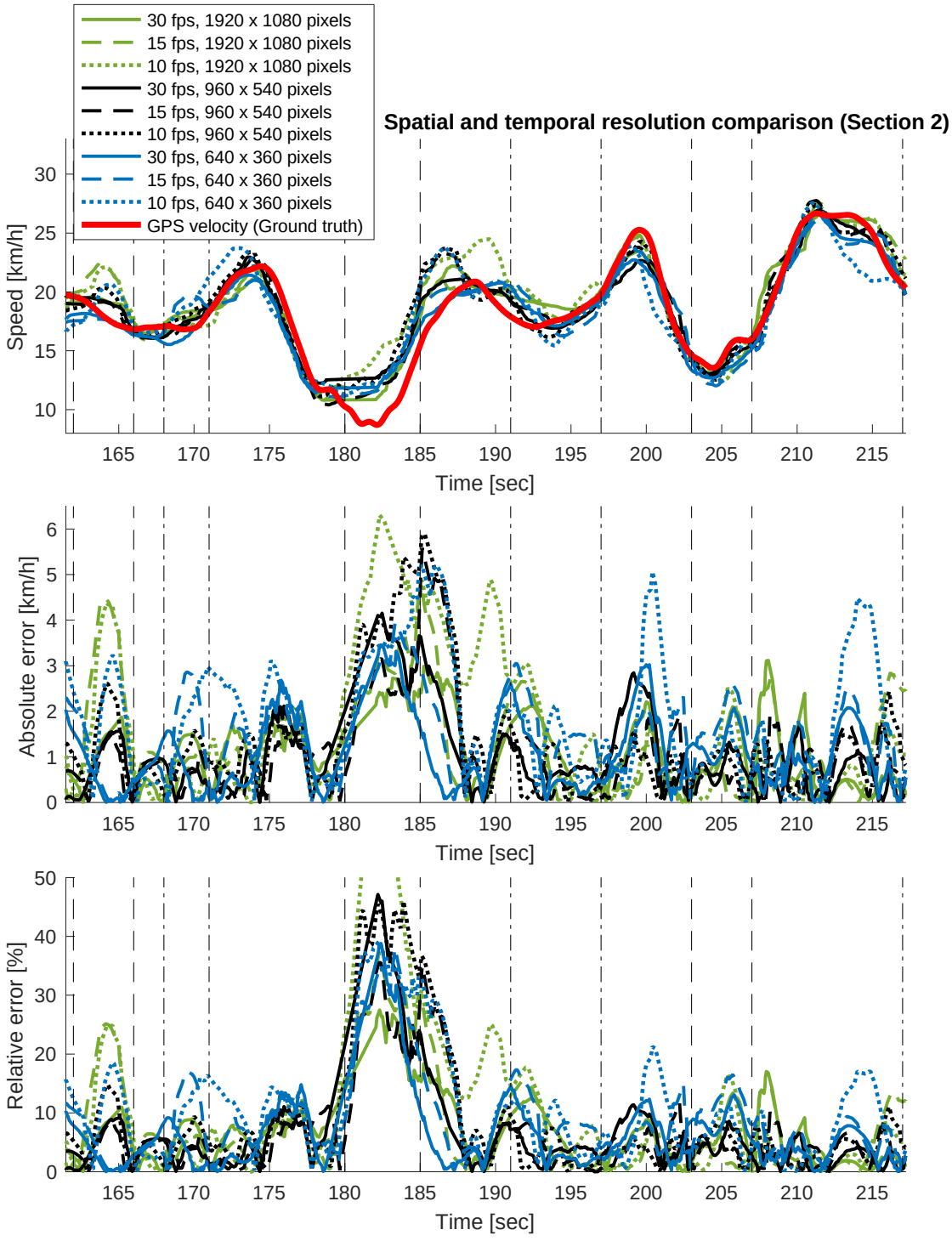

Figure B-11 Velocity profile of section 2 with the corresponding absolute and relative errors for all the analysed spatial and temporal resolutions. Vertical dashed and dash-dot line pairs indicate each one of the six 90º curves (Note that the last one does not finish inside the plot). More details on page 94.

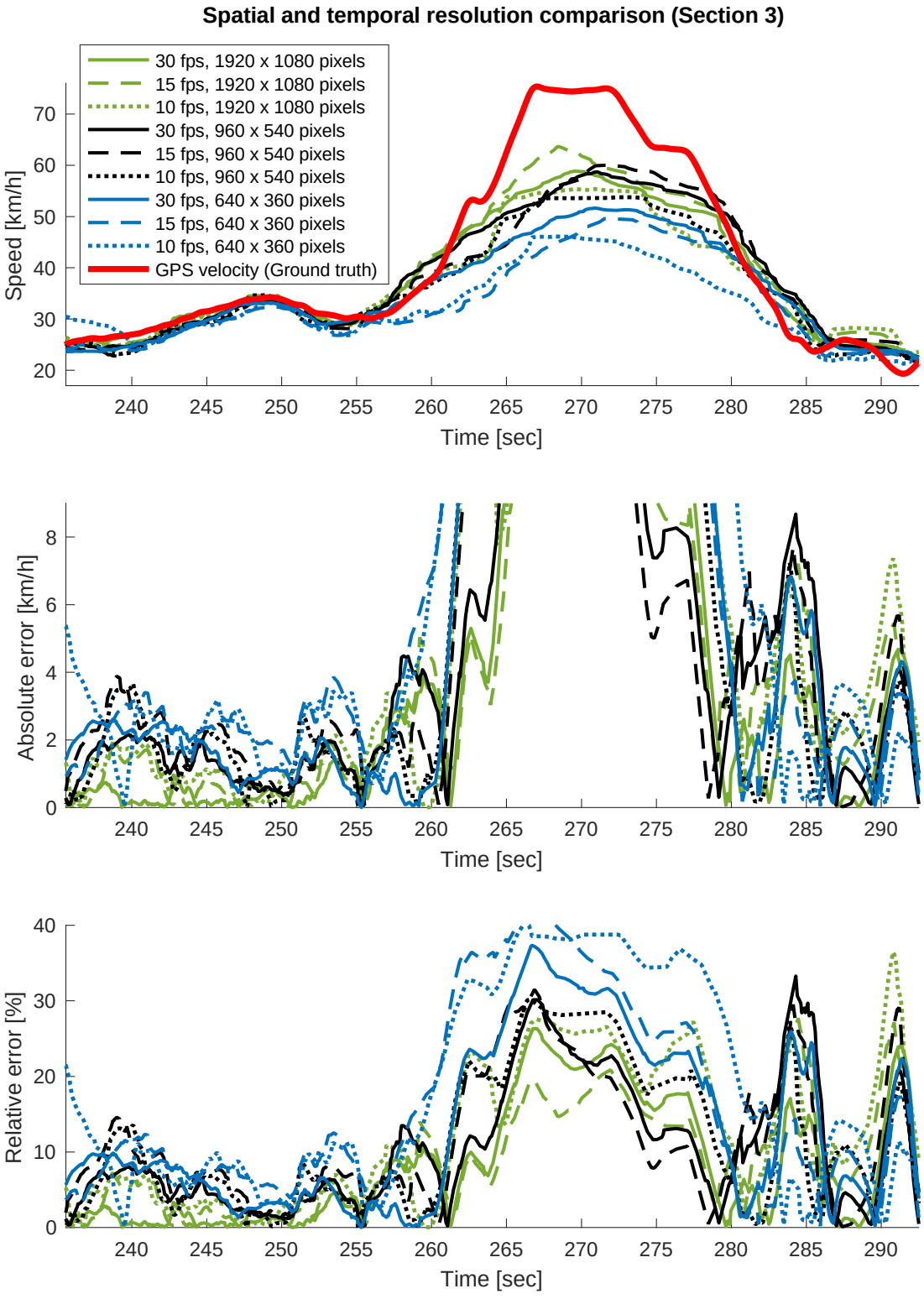

Figure B-12 Velocity profile of section 3 with the corresponding absolute and relative errors for all the analysed spatial and temporal resolutions (this two plots are partially shown to see better the parts where the algorithm works well). More details on page 93.

Appendix C Software

In this appendix the structure of the developed MATLAB scrips, as well as their individual purposes, are explained.

There is a total of 11 scripts/functions that can be divided into two main group:

- Scripts to read the raw data of the GPS and camera, and save the interesting information into a *.mat* file to be accessed quickly when it is required in other scripts/functions:
  - **cam_parameters.m** (function).
  - **filtering_data.m** (function).
  - **GPS_reading_save.m**
  - **GPS_resampling.m**
  - **read_save_video.m**
  - **timevector.m** (function).
- Scripts to estimate the velocity:
  - **main_VO.m**
  - **bucketFeatures.m** (function).
  - **bucketFeatures_road.m** (function).
  - **SCALE_FACTOR.m** (function).
  - **track.m** (function).

Now, each of them is briefly explained.

C.1 Scripts to read and save the raw data

From these code files, the ones that generates the output (*.mat* file) that will directly be read by the velocity estimation scripts are **GPS_reading_save.m** (or **GPS_resampling.m** if the data is resampled and low-pass filtered) and **read_save_video.m**.

C.1.1 GPS_reading_save.m

It is a code to create time series of GPS data time-synchronized with the video. It reads the raw data from a *.xlsx* document and exports the latitude, longitude, height, velocity and time to a *.mat* file called *GPS_data_sync.mat*.

This script uses the custom function **timevector.m**, which creates a time vector and time series of data using video time as reference.

Then, if the data needs to be resampled and low-pass filtered, the **GPS_resampling.m** script is utilised. It needs the *GPS_data_sync.mat* file, and what it does is:

1. Transform latitude and longitude into Cartesian coordinates.
2. Low-pass filter the raw measurements (both position and velocity) to eliminate higher frequencies with the custom function **filtering_data.m**.
3. Resample the filtered measurements (to 30 Hz in this thesis, since the video data was recorded at that frequency).
4. Save the resampled velocity (position is not saved but is used in this script to make some plots about the trajectory of the vehicle).

C.1.2 read_save_video.m

It is a code to read a video file and save its frames and other specific information in a *.mat* file. Apart from that, before saving the *.mat*, file other tasks are also done, explained in *4.1 Pre-processing steps*:

1. Convert each frame to greyscale.
2. Remove lens distortion from each frame.
3. Rotate each frame.
4. Reduce size of the greyscale frames (if necessary), that is, reduce the spatial resolution of the video.

This script uses the created function **cam_parameters.m** to define the camera parameters and the lens distortion coefficients to then create a *cameraParameters* object that will be need not only in **read_save_video.m**, but also in **main_VO.m**

Apart from the frames, other parameters are also exported that will be needed in **main_VO.m**:

- Duration of the exported video frames.
- Starting time of the video from which the frames are exported.

- Scale of reduction of the spatial resolution.
- Frame rate.
- Camera parameters.

C.2 Scripts to estimate the velocity

From this scripts/functions the main one is **main_VO.m**. This is the core of the velocity estimation (explained in *4 ALGORITHM OVERVIEW*). It needs as inputs the *.mat* files created with the scripts of C.1: *GPS_data_sync.mat* and the video frames with their information about the video.

Then, the rest of functions are utilised to do different tasks described below.

C.2.1 bucketFeatures.m

Function to detect corner features on the top-half part of the image using the *bucketing* technique to ensures a uniform distribution of features over the image (described in *4.2.1 Feature detection: FAST algorithm for corner detection*).

C.2.2 bucketFeatures_road.m

Function to detect corner features only on the road using the FAST feature detector (described in *4.3.1 Road feature detection, tracking and triangulation*).

C.2.3 SCALE_FACTOR.m

Function to estimate the scale factor of the camera pose estimation (described in *4.3 Computing the scale factor*):

1. Triangulate detected/tracked corners from two adjacent views.
2. Fit a plane to the resultant 3D points.
3. Compute distance from that plane to camera position.
4. Obtain scale factor.

C.2.4 track.m

Function to track corner features from previous frame to current one using the KLT tracker.