Computer vision approaches for vehicle sideslip angle estimation

1st Leonardo Serena
Department of Industrial Engineering
University of Padova
Padua, Italy
leonardo.serena@unipd.it

2nd Basilio Lenzo
Department of Industrial Engineering
University of Padova
Padua, Italy
basilio.lenzo@unipd.it

3rd Mattia Bruschetta
Department of Information Engineering
University of Padova
Padua, Italy
mattia.bruschetta@unipd.it

4th Ricardo de Castro
Department of Mechanical Engineering
University of California at Merced
Merced, CA, USA
rpintodecastro@ucmerced.edu

***Abstract*—Vehicle sideslip angle, defined as the angle between the longitudinal axis of a vehicle and its velocity vector, is a crucial parameter in vehicle dynamics. Unfortunately vehicle sideslip angle is very hard to access directly, therefore a variety of estimation methods have been developed so far. Such estimation methods are essentially based on model-based approaches or neural networks. This paper looks at the problem from a fresh angle, by investigating possible solutions to measure vehicle sideslip angle via computer vision techniques, harnessing recent improvements in computer vision algorithms. Preliminary experiments on a radio-controlled scaled vehicle show promising results using the "phase correlation" algorithm.**

***Index Terms*—Sideslip, Vehicle dynamics, Computer vision**

I. Introduction

Vehicle safety is a crucial aspect in the development of advanced driver assistance systems. To implement and deploy effective vehicle safety systems, including lateral stability controls, accurate knowledge of relevant vehicle motion parameters is required. A critical role is played by vehicle sideslip angle, β, i.e. the angle between vehicle longitudinal axis and velocity vector at the centre of mass. The literature generally agrees on the idea that knowledge of β can lead to significantly improve vehicle safety [1], [2].

Unfortunately, β is not easily accessible: velocity measurements from vehicle onboard sensors (IMUs) are not reliable enough, GPS methods cannot guarantee a sufficient high-frequency update and may be unreliable in many situations due to signal unavailability. Sideslip angle sensors are commercially available, but with too large cost to be implemented on a passenger car. A possible solution is a state estimator, either observer-based or neural network-based [3], [4]. By combining measurements from onboard sensors such as longitudinal and lateral acceleration, angular velocities, and wheel velocities, it is possible to obtain an estimate of β. Suitable model-based observers may even be further improved with GPS measurements where available. This approach requires a fairly complex model of the vehicle which often represents a too-high computational burden for the task. Artificial neural network approaches require fewer resources and allow to fairly easily relate available sensor measurements to β, without vehicle models. On the other hand, a typical drawback is the training procedure, which requires a high amount of data and must be repeated in case of system changes - not always feasible.

Recent works that employ image correlation techniques showed that it is possible to measure the velocity vector of the vehicle with a camera. For example, [5] used a camera pointing to the tyre to compute longitudinal slip ratio by comparing the displacement of the tyre with respect to the road in subsequent images. In a similar way [6] used a high-performance camera pointed at the road surface, computing the velocity vector by comparing subsequent frames of the footage: i) by employing Shi-Tomasi feature selection [7], particles of the road surface are detected; ii) Lucas-Kanade [8] optical flow algorithm is then used to track the features in the next frame and compute the displacement vector, related to β. To obtain even more accurate measurements of the displacement, visual odometry correction techniques were assessed. A real-time implementation of a similar solution was presented in [9], where the mentioned visual odometry corrections on the displacement vector were not implemented to save computational resources, obtaining a less accurate result. In [10] a Remote Controlled (RC) scaled model of a vehicle is used to test an image correlation technique to measure sideslip angle. The camera is positioned on the vehicle front end, pointing to the road surface. The images are projected into the road surface plane, where red markers are placed in order to be detected and tracked with the aforementioned algorithms. The measurements are combined with a Kalman filter and compared with measurements from another camera fixed above the testing surface and used as ground truth.

The discussed preliminary results look promising and are worth further investigation: other computer vision techniques can be applied and their operational and technical limits should be evaluated. For example [11] used a phase correlation tech-

nique and a down-facing camera to reconstruct the trajectory - not the speed profile - of a Rover. In [12] a phase correlation solution has been proposed for the same purpose on a passenger car, but the solution has been tested only in laboratory tests. A similar solution may be assessed to compute the velocity vector instead. On such basis, the objective of this paper is to implement an accurate sideslip sensor with a down-facing camera using image correlation techniques. First, algorithm candidates, including sparse optical flow and cross-correlation techniques, are evaluated. The implementation exploits the OpenCV library [13]. Tests are then carried out on an remote-controlled (RC) scaled vehicle: footage is recorded and subsequently analyzed and processed.

II. Motion estimation

The first objective is to compute the optical flow, i.e. the apparent motion present in a stream of images when the camera is moving. By evaluating the displacement of the camera from two subsequent images it is possible to measure the velocity vector of the vehicle. There are two main methods to estimate the optical flow of a moving camera: dense and sparse optical flow. Dense optical flow methods take into account the transition of all the pixels in the images, therefore they are the most accurate methods. Yet, high accuracy reflects in high computational cost. Sparse optical flow methods aim to track only a set of specific features in the images instead of the whole image, therefore the computational cost is far lower - at the expense of accuracy. In this context the real time capabilities of the solution are more valuable than high accuracy, so sparse methods employed in [9] are initially inspected in paragraphs 1 and 2 below. An interesting solution based on [12] that takes into account the whole image is presented in paragraph 3.

1) Shi-Tomasi feature selection: By exploiting the presence of textured surfaces on the road, it is possible to employ the Shi-Tomasi algorithm to detect features in images. The aim is to detect corners in an image. Corners are intended as interesting regions of an image, i.e. that are easy to track from frame to frame. Specifically, a corner is a region that produces large differences in the image when shifted in the x and y directions, differently from planar surfaces (where changes cannot be detected) and edges (where changes can only be detected in one direction). By considering a window W around a pixel (x,y), for a displacement $(\Delta x, \Delta y)$ the algorithm aims to maximize the auto-correlation function:

$$E(x,y) = \sum_{(x_i,y_i)\in W} [I(x_i,y_i) - I(x_i+\Delta x, y_i+\Delta y)]^2 \quad (1)$$

where $I(x_i, y_i)$ represents the pixel intensity. The argument can be approximated via Taylor series expansion:

$$I(x_i+\Delta x, y_i+\Delta y) = I(x_i,y_i) + I_x\Delta x + I_y\Delta y \quad (2)$$

where I_x and I_y are partial derivatives on respectively the x and y direction. This yields:

$$E \approx \begin{bmatrix}\Delta x & \Delta y\end{bmatrix} \begin{bmatrix}\sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_y I_x & \sum_W I_y^2\end{bmatrix} \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix} \quad (3)$$

where the involved matrix is the auto-correlation matrix $A_a \in \mathbb{R}^2$. The Shi-Tomasi algorithm evaluates the eigenvalues λ_1, λ_2 of A_a, that can be associated with the x and y rate of change in the image: a large λ_1 indicates differences in the x direction around the selected pixel, and an a high level of λ_2 indicates differences in the y direction around the selected pixel. If both eigenvalues are nearly zero the selected region is associated to a plain surface; if only one eigenvalue is nearly zero the selected region is associated to an edge. Therefore a pixel is selected as a corner if the score function

$$R = \min(\lambda_1, \lambda_2) \quad (4)$$

exceeds a predefined positive threshold.

2) Lucas-Kanade feature tracking: Lucas-Kanade (LK) algorithm is a sparse optical flow method that aims to track selected features among two subsequent images (frames), computing the velocity vector $v = [\frac{\delta x}{\delta t}, \frac{\delta y}{\delta t}]$, where δt is the interval of time between two subsequent frames. Given that the velocity vector is evaluated in a discrete time system, the interval between two subsequent frames is equal to $\delta t = 1$, therefore, to compute the actual velocity of a moving object, the spatial displacement $[\delta x, \delta y]$ must be divided by the sampling time. Since LK is a sparse method, first a set of features is selected in a first image and then tracked in the next one. The algorithm makes three main assumptions: the displacement between two subsequent frames is not large; the brightness in two frames is approximately constant; the pixels around the selected feature move all in the same direction with the feature itself. While the first assumption can be overcome with other Computer Vision tools, the two last assumptions are a key part of the implementation. The assumption of constant brightness is formalized as:

$$I(x,y,t) = I(x+\delta x, y+\delta y, t+\delta t) \quad (5)$$

which can be linearized by approximating via Taylor series expansion:

$$I(x+\delta x, y+\delta y, t+\delta t) \approx I(x,y,t) + I_x\delta x + I_y\delta y + I_t\delta t \quad (6)$$

where I_x and I_y and I_t are partial derivatives respectively with respect to x, y, and time t. This leads to the optical flow equation:

$$0 \approx I_x\frac{\delta x}{\delta t} + I_y\frac{\delta y}{\delta t} + I_t \quad (7)$$

that is an undetermined equation in the two variables $[\delta x, \delta y]$ recalling the fact that δt is known. Lucas-Kanade algorithm solves this problem by evaluating a window of n pixels around the feature: one of the assumptions is that the n pixels around the selected point move all in the same direction. Therefore it is possible to solve the optical flow equation for all the pixels in a selected window W of n pixels. By taking:

$$A = \begin{bmatrix} I_x(1) & I_y(1) \\ I_x(2) & I_y(2) \\ . & . \\ . & . \\ I_x(n) & I_y(n)\end{bmatrix}, b = \begin{bmatrix} -I_t(1) \\ -I_t(2) \\ . \\ . \\ -I_t(n)\end{bmatrix} \quad (8)$$

then the displacement vector $v = [\frac{\delta x}{\delta t}, \frac{\delta y}{\delta t}]$ is computed as the least square solution of:

$$A^T A v = A^T b \tag{9}$$

that results in:

$$\begin{bmatrix} \sum_W I_x^2 & \sum_W I_x I_y \\ \sum_W I_y I_x & \sum_W I_y^2 \end{bmatrix} v = - \begin{bmatrix} \sum_W I_x I_t \\ \sum_W I_x I_t \end{bmatrix} \tag{10}$$

that can be solved by inverting $A^T A$. It should be noted that when the same window is considered, $A^T A$ corresponds to the auto-correlation matrix of the Shi-Tomasi method, A_a. So, $A^T A$ in eq. 9 is invertible and well-conditioned by properly selecting the eigenvalues' threshold (eq. 4), i.e. by selecting the features to be tracked with the Shi-Tomasi method. In the OpenCV implementation, pixels near the center are weighted with a window function in order to increase their impact with respect to pixels in the edges.

3) Phase correlation (PC): This method exploits the shift property of the Fourier transform to compute the displacement between two images. Recall the shifting property of the Fourier transform: a shift in the time domain correspond to a phase shift in the frequency domain. Therefore by evaluating the phase shift between two subsequent images it is possible to obtain their spatial displacement. The idea is that by applying the Fourier transform to a first frame $Im_t(x, y)$ at time t:

$$\mathcal{F}\{Im_t(x, y)\} = G_t(p, q) \tag{11}$$

and applying the same transform to the subsequent image $Im_{t+1}(x+\delta x, y+\delta y)$ where $v = [\delta x, \delta y]$ is the displacement vector:

$$\begin{aligned} \mathcal{F}\{Im_{t+1}(x + \delta x, y + \delta y)\} &= G_{t+1}(p, q) \\ &= G_t e^{2\pi i(p\delta x + q\delta y)} \end{aligned} \tag{12}$$

where p and q in the Fourier domain correspond to x and y in the time domain. The phase shift between images can be evaluated by computing the normalized cross-power-spectrum:

$$\begin{aligned} R &= \frac{G_t \cdot G_{t+1}^*}{|G_t \cdot G_{t+1}^*|} \\ &= \frac{G_t \cdot G_t^* e^{2\pi i(p\delta x + q\delta y)}}{|G_t \cdot G_t^*|} \\ &= e^{2\pi i(p\delta x + q\delta y)} \end{aligned} \tag{13}$$

where $\cdot$ is the element-wise multiplication and $*$ is the complex conjugate operator. From (13) it is possible to obtain $v = [\delta x, \delta y]$ by inverse transforming R. In fact the inverse transform of the exponential function result in an impulse located in the displacement position: the location of the maximum value of $\mathcal{F}^{-1}(R)$ corresponds to the velocity vector. In the proposed scenario every pixel carries information about the velocity vector of the vehicle, therefore this method allows a very accurate estimate of the displacement. This method is also naturally robust to white noise. Since every pixel of the image is evaluated, this method is more computationally expensive with respect to the ones presented in the above paragraphs 1 and 2, however, well-known algorithms may be exploited to speed up the Fourier transformation process.

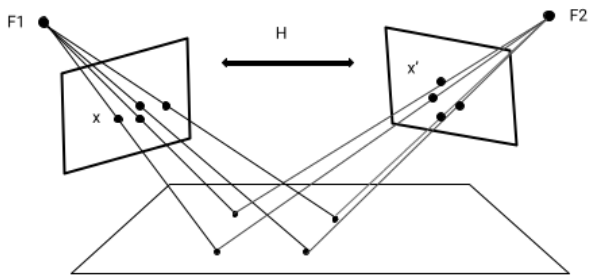

Fig. 1. Homography representation: the same camera is pointing to a group of co-planar features from two different points (F1, F2), x and x' are the projection coordinates of the features

III. Sideslip angle computation

As pointed out in [6] [14], estimating the velocity vector of the vehicle with a computer vision approach can lead to erroneous measurements in case of significant angular velocities of the vehicle body. This problem can be solved by camera pose estimation, that is the determination of the camera position and orientation with respect to an object (in this case the detected features). This is a known problem in computer vision and can be solved by computing the *fundamental* matrix, which completely describes the geometric relationship between tracked points in two different images and the camera. Since the detected features lay on a surface, the camera pose estimation problem can be simplified with a homography transformation (Fig. 1). The homography relates features located in the same planar surface and the camera by mapping their respective projection coordinates from one frame to the other, i.e. $x' = h(x)$. The homography function is embedded in the homography matrix $H \in \mathbb{R}^{3\times3}$ with 8 degrees of freedom, since one entry represents a scaling factor that can be neglected. In order to estimate H at least 4 points are needed (2 sets of coordinates to estimate 8 entries). For each tracked point it is [15]:

$$x' = Hx \tag{14}$$

which can be rewritten as:

$$x' \times Hx = 0 \tag{15}$$

Equation (15) is reformulated into:

$$Ah = 0 \tag{16}$$

where h represents the 9 entries of the homography matrix and is in the null space of matrix A, therefore it can be obtained via singular value decomposition (SVD). In order to obtain a robust solution, the Random Sample Consensus (RANSAC) algorithm is used [16]. RANSAC is an iterative method used to estimate a model (the matrix H) rejecting outliers. The algorithm randomly samples data points to produce a model. The model is then evaluated over the whole dataset, labeling data points that do not fit the model as outliers, while the remaining are labeled as inliers. If the number of inliers

satisfies a selected threshold, the model is recomputed taking into account only inliers, to improve accuracy.

In this context, the camera motion is constrained to a planar motion and the road surface can be reduced to a plane, therefore the subsequent frame motion can be approximated by an affine transformation (rotation, translation and scaling). The camera pose estimation can be reconstructed by an affine matrix $M \in \mathbb{R}^{2\times 3}$ instead of a homography matrix. This simplifies the pose estimation since only 3 points are needed to compute the matrix, leading to a less complex model to be evaluated by the RANSAC algorithm in order to obtain a valid solution. This approximation takes into account vertical movements - given that estimating the Affine matrix includes a scaling factor - as well as rotation on the z-axis, but neglects the image deformation possibly caused by vehicle rolling or pitching. The error introduced by this approximation can be reduced by means of Kalman filtering.

For example, a Kalman Filter based on a kinematic model of the vehicle can be used. The choice of the kinematic model is motivated by the fact that accurate knowledge of the parameters of the dynamic model is not needed [3]. The main equations are:

$$a_x = \dot{v_x} - v_y r \tag{17}$$

$$a_y = \dot{v_y} + v_x r \tag{18}$$

where a_x and a_y are the vehicle longitudinal and lateral acceleration, respectively, v_x and v_y are the longitudinal and lateral components of the vehicle speed at the centre of mass, r is the yaw rate. The state of the discretized system is:

$$x(k) = [v_x(k)\ v_y(k)]^T \tag{19}$$

with input:

$$u(k) = [a_x(k)\ a_y(k)]^T \tag{20}$$

The discrete state and input matrices are:

$$A = \begin{bmatrix} 0 & r\delta t \\ -r\delta t & 0 \end{bmatrix} B = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \delta t \tag{21}$$

and the output variable y is described by:

$$y = Hx = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x \tag{22}$$

where r is assumed to not vary too much between samples (spaced by δt). The actual y is reconstructed by the camera sensor: in the proposed configuration the displacement vector extracted by either the affine matrix or the phase correlation peak represents the scaled velocity vector of the vehicle.

IV. Preliminary validation

A first set of tests have been conducted on non-vehicle-related footage. The footage was taken with a smartphone camera equipped with a Sony IMX363 sensor (12 MP 1/2.55" sensor, f/1.7 aperture lens), recording a normal walk. The camera pointed downwards to a feature-rich pavement. Videos can be accessed here: https://bit.ly/walk240223, where the displayed angle - specified in degrees with respect to the longitudinal axis of the smartphone - was computed with the Lucas-Kanade method with the Affine transformation correction. The reference system used to compute β is attached to the camera, with the x-axis pointing to the direction of movement along the longitudinal axis of the smartphone and the z-axis pointing upwards. The y-axis direction comes from the right-hand rule. The angle is then computed as $\beta = \arctan(\frac{\delta y}{\delta x})$. In Fig. 2 the proposed algorithms are compared: in the LK algorithm β is computed as the mean of all the angles of the tracked features; the same features are processed with RANSAC algorithm to compute the Affine transformation correction; eventually the PC algorithm has been evaluated. The algorithms yield comparable results. Further investigations are carried out with an RC car, using other sensor information: in a first implementation, the estimated longitudinal velocity will be compared with readings from wheel encoders.

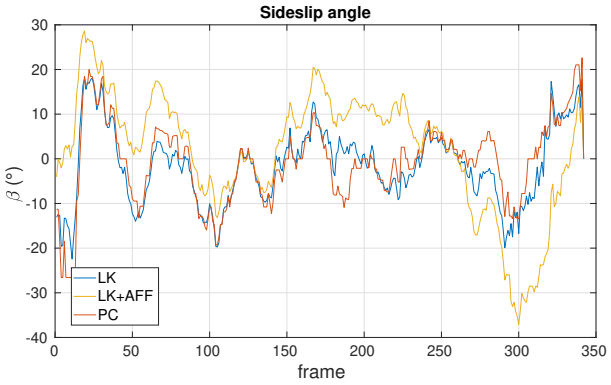

Fig. 2. Measured sideslip angle from walk video: the proposed algorithms are compared (Lucas-Kanade, Lucas-Kanade with Affine transformation, Phase Correlation)

V. Results

In this section the performance of the algorithms is assessed by comparing the longitudinal velocity computed with computer vision algorithms with the longitudinal velocity returned by onboard vehicle sensors. Tests have been performed on a customized 1:12 scale RC vehicle [17]. The RC car is powered by 4 electric motors with a maximum speed around 18 km/h, controlled by a Teensy 4.1 micro-controller. It features Hall sensors in each wheel to measure wheel angular velocity, and a 9 Degrees-of-freedom Inertial Measurement Unit (IMU). Within this work, the available vehicle has been equipped with a Raspberry PI 3B along with a PI Camera V2 to record footage and analyze it offline. The camera has been mounted on a custom support set on the rear axle of the vehicle so that the camera was pointing downwards as in Fig. 3. The Raspberry PI has been set to start the recording when the micro-controller started logging signals from the vehicle's sensors.

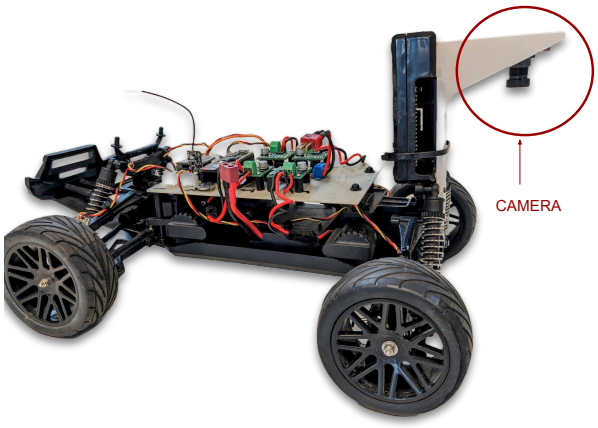

Fig. 3. RC car with custom camera support

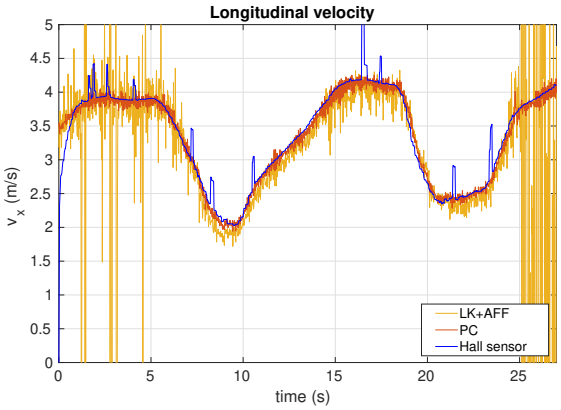

Fig. 4. Camera and Hall sensor measurements of the longitudinal velocity

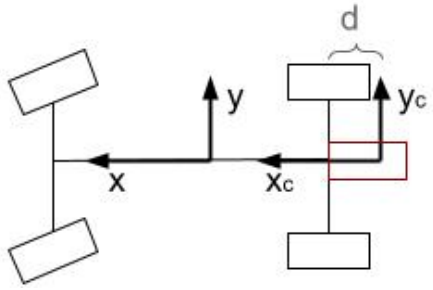

Fig. 5. Vehicle and camera reference frames

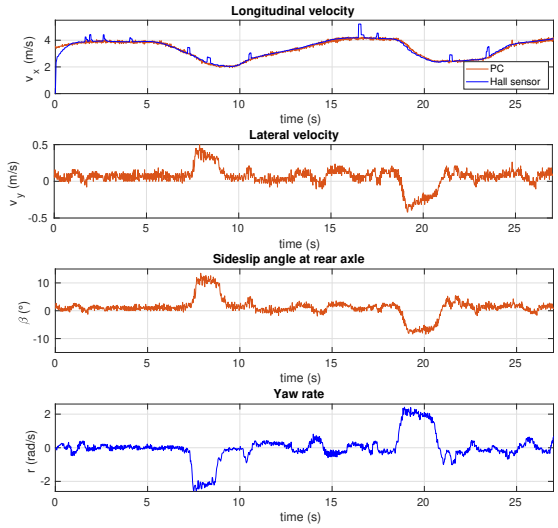

Fig. 6. First test: camera (orange) and on-board sensors (hall sensors and gyroscope, blue) measurements

A. Longitudinal dynamics

The longitudinal velocity of the vehicle is computed from the Hall sensors measurements with:

$$v_x^{HALL} = \frac{1}{4}\sum_{i=1}^{4} \omega_i \cdot R \tag{23}$$

where ω_i is the angular velocity of the i^{th} wheel and R is the wheel radius. This measurement is compared to the longitudinal velocity computed with the proposed algorithms (PC and LK). The computed displacement in pixel has been mapped to real world coordinates by exploting the PI Camera Field of View (FoV) dimensions in real world coordinates and knowing the frequency of acquisition (150 FPS). To correctly represent results, both data sources have been sampled at 50 Hz.

In Fig. 4 the measurements of the longitudinal velocity from the wheel Hall sensors and the proposed computer vision algorithms are compared. The velocity measurement from the Hall sensors shows some peaks due to the sensor not properly detecting the magnet present in the wheel. The computer vision solutions shows very high correlation with the wheel sensor measurement. The Lucas-Kanade results show inconsistent results, i.e., the LK method does not provide a reliable measurement for the whole duration of the test. This is likely due to the very low quality of the videos captured by the PI camera, making it hard to track specific details present in the asphalt surface. On the other hand the Phase Correlation algorithm shows high robustness, providing consistent results among different environments and light conditions.

B. Lateral dynamics

In this paragraph the lateral velocity of the vehicle is assessed. Only the Phase Correlation algorithm is herein considered. The lateral velocity at the rear axle of the vehicle, v_y^{RR}, is obtained by computing:

$$v_y^{RR} = v_y^{CAM} + d \cdot r \tag{24}$$

where v_y^{CAM} is the velocity measured by the camera and d is the distance between camera and rear axle, i.e. 4 cm in our case.

In Fig. 6 and Fig. 7 the longitudinal and lateral velocities obtained by the PC algorithm are compared to other on-board sensors. The sideslip angle at the rear axle is computed using (24). The lateral velocity v_y^{RR} is expected to be zero

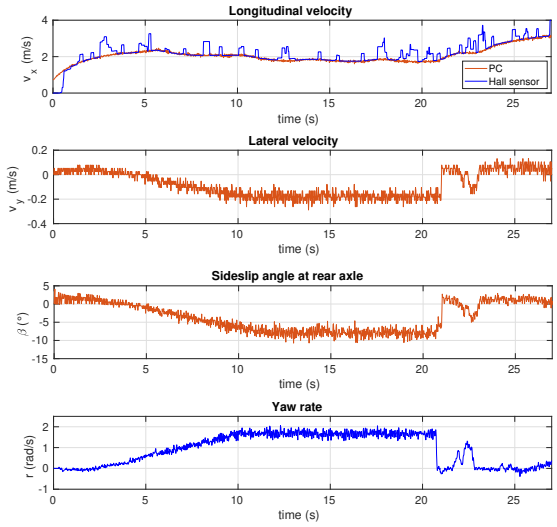

Fig. 7. Second test: camera (orange) and on-board sensors (hall sensors and gyroscope, blue) measurements

(i.e., no slip angle at the rear axle) except when the vehicle is performing a turning maneuver [2]. The lateral velocity measurement shows a small bias that is motivated by the presence of a slight tilt in the camera housing. The promising results on the longitudinal dynamics, along with these preliminary results for the vehicle lateral dynamics, suggest that the proposed computer vision approach is a promising solution to the problem of measuring sideslip angle.

VI. Conclusions

In this work, new methods to measure the sideslip angle with computer vision techniques have been proposed, analyzed and tested. The algorithms have been tested employing a scaled vehicle customized with a down-facing camera: the camera measurements are compared to the on board sensors of the vehicle. The LK algorithm showed some limitations since it requires high quality images in order to provide reliable results. The PC algorithm returned consistent results in different scenarios, showing high robustness to noise with respect to the LK solution. The camera measurements provided an accurate longitudinal velocity measurement, consistent with the sensors available on the vehicle. Future work will focus on the validation of the lateral velocity measurement on a real vehicle and the real time implementation of the proposed algorithm.

VII. Acknowledgements

This work was supported in part by the Italian Ministry of Foreign Affairs and International Cooperation, grant number PGR01170.

References

[1] J. Liu, Z. Wang, L. Zhang, and P. Walker, "Sideslip angle estimation of ground vehicles: a comparative study," *IET Control Theory & Applications*, vol. 14, no. 20, pp. 3490–3505, 2020.

[2] B. Lenzo, "Torque vectoring control for enhancing vehicle safety and energy efficiency," in *Vehicle Dynamics: Fundamentals and Ultimate Trends*. Springer, 2022, pp. 193–233.

[3] D. Chindamo, B. Lenzo, and M. Gadola, "On the vehicle sideslip angle estimation: A literature review of methods, models, and innovations," vol. 8, no. 3, p. 355. [Online]. Available: http://www.mdpi.com/2076-3417/8/3/355

[4] M. Viehweger, C. Vaseur, S. van Aalst, M. Acosta, E. Regolin, A. Alatorre, W. Desmet, F. Naets, V. Ivanov, A. Ferrara *et al.*, "Vehicle state and tyre force estimation: demonstrations and guidelines," *Vehicle system dynamics*, vol. 59, no. 5, pp. 675–702, 2021.

[5] T. R. Botha and P. S. Els, "Tire longitudinal slip-ratio measurement using a camera," in *Volume 3: 16th Int. Conf. on Advanced Vehicle Technologies*. ASME, p. V003T01A043. [Online]. Available: https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings/IDETC-CIE2014/46346/Buffalo,%20New%20York,%20USA/257281

[6] ——, "Digital image correlation techniques for measuring tyre-road interface parameters: Part 1 – side-slip angle measurement on rough terrain," vol. 61, pp. 87–100. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0022489815000488

[7] J. Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600.

[8] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (IJCAI)," vol. 81.

[9] D. K. Johnson, T. R. Botha, and P. S. Els, "Real-time side-slip angle measurements using digital image correlation," vol. 81, pp. 35–42. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S002248981730263X

[10] C. Kuyt and M. Como, "Mixed kinematics and camera based vehicle dynamic sideslip estimation for an RC scaled model," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, pp. 1216–1221. [Online]. Available: https://ieeexplore.ieee.org/document/8511487/

[11] T. Kazik and A. H. Goktogan, "Visual odometry based on the fourier-mellin transform for a rover using a monocular ground-facing camera," in *2011 IEEE International Conference on Mechatronics*. IEEE, pp. 469–474. [Online]. Available: http://ieeexplore.ieee.org/document/5971331/

[12] M. Birem, R. Kleihorst, and N. El-Ghouti, "Visual odometry based on the fourier transform using a monocular ground-facing camera," vol. 14, no. 3, pp. 637–646. [Online]. Available: http://link.springer.com/10.1007/s11554-017-0706-3

[13] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of SW Tools*, 2000.

[14] T. R. Botha and P. Schalk Els, "Digital image correlation techniques for measuring tyre–road interface parameters: Part 2 – longitudinal tyre slip ratio measurement," vol. 61, pp. 101–112. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0022489815000683

[15] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. USA: Cambridge University Press, 2003.

[16] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, p. 381–395, jun 1981. [Online]. Available: https://doi.org/10.1145/358669.358692

[17] S. Lovato, G. Righetti, A. Canton, B. Lenzo, and M. Massaro, "Development of a remote-controlled scaled multi-actuated vehicle," in *Proceedings of I4SDG Workshop 2023*, V. Petuya, G. Quaglia, T. Parikyan, and G. Carbone, Eds. Cham: Springer Nature Switzerland, 2023, pp. 270–277.